ACTIONSCRIPT® 3.0
Geliştirici Kılavuzu

Yasal bildirimler

Yasal bildirimler için bkz. http://help.adobe.com/tr_TR/legalnotices/index.html.

Son güncelleme: 7/10/2015

İçindekiler

Bölüm 1: Tarih ve saatlerle çalışma
Takvim tarih ve saatlerini yönetme 1
Zaman aralıklarını denetleme 4
Tarih ve saat örneği: Basit analog saat 6

Bölüm 2: Dizelerle çalışma
Dizelerin temelleri 10
Dizeler oluşturma 11
length özelliği 12
Dizelerdeki karakterlerle çalışma 12
Dizeleri harmanlama 13
Diğer nesnelerin dize halinde temsilini alma 14
Dizeleri bitiştirme 14
Dizelerdeki alt dizeleri ve desenleri bulma 15
Dizeleri büyük harfe veya küçük harfe dönüştürme 19
Dize örneği: ASCII sanatı 19

Bölüm 3: Dizilerle çalışma
Dizilerin temelleri 25
Dizinlenmiş diziler 26
İlişkilendirilebilir diziler 37
Çok boyutlu diziler 41
Dizileri klonlama 42
Array sınıfını genişletme 43
Diziler örneği: PlayList 48

Bölüm 4: Hataları işleme
Hata işlemenin temelleri 52
Hata türleri 53
ActionScript 3.0'da hata işleme 55
Flash çalışma zamanlarının hata ayıklayıcı sürümleri 57
Bir uygulamada eşzamanlı hataları işleme 58
Özel hata sınıfları oluşturma 62
Hata olaylarını ve durumunu yanıtlama 63
Error sınıflarını karşılaştırma 66
Hataları işleme örneği: CustomErrors uygulaması 70

Bölüm 5: Normal ifadeler kullanma
Normal ifadelerin temelleri 76
Normal ifade sözdizimi 77
Dizelerle normal ifadeleri kullanma yöntemleri 90
Normal ifade örneği: Wiki ayrıştırıcısı 91

Bölüm 6: XML ile çalışma

XML temelleri 96
XML işlemeye yönelik E4X yaklaşımı 99
XML nesneleri 100
XMLList nesneleri 103
XML değişkenlerini başlatma 104
XML nesnelerini birleştirme ve dönüştürme 105
XML yapılarında geçiş yapma 107
XML ad alanlarını kullanma 111
XML tür dönüştürmesi 112
Harici XML belgelerini okuma 113
ActionScript'te XML örneği: internetten RSS verilerini yükleme 114

Bölüm 7: Yerel JSON işlevini kullanma

JSON API'sine genel bakış 118
Özel JSON davranışını tanımlama 119

Bölüm 8: Olayları işleme

Olay işleme temelleri 126
ActionScript 3.0 olay işleme, önceki sürümlerden nasıl farklılık gösterir 128
Olay akışı 130
Olay nesneleri 132
Olay dinleyicileri 135
Olay işleme örneği: Çalar Saat 142

Bölüm 9: Uygulama etki alanlarıyla çalışma

Bölüm 10: Görüntü programlama

Görüntü programlama temelleri 153
Çekirdek görüntüleme sınıfları 156
Görüntüleme listesi yaklaşımının avantajları 158
Görüntüleme nesneleriyle çalışma 160
Görüntüleme nesnelerini işleme 176
Nesnelere animasyon uygulama 196
Sahne Alanı yönlendirmesi 198
Görüntüleme içeriğini dinamik olarak yükleme 201
Görüntüleme nesnesi örneği: SpriteArranger 207

Bölüm 11: Geometriyle çalışma

Geometri temelleri 213
Point nesnelerini kullanma 214
Rectangle nesnelerini kullanma 216
Matrix nesnelerini kullanma 219
Geometri örneği: Görüntüleme nesnesine matris dönüştürmesi uygulama 220

Bölüm 12: Çizim API'sini kullanma

API çizmeyle ilgili temel bilgiler 225
Grafikler sınıfı 226

Çizgi ve eğriler çizme . . . 226
Yerleşik yöntemleri kullanarak şekiller çizme . . . 229
Degrade çizgiler ve dolgular oluşturma . . . 230
Çizim yöntemleriyle Math sınıfını kullanma . . . 234
Çizim API'si ile animasyon uygulama . . . 235
API çizme örneği: Algoritmik Görsel Oluşturucu . . . 236
Çizim API'sinin ileri düzey kullanımı . . . 238

Bölüm 13: Bitmap'lerle çalışma
Bitmap'lerle çalışmanın temelleri . . . 245
Bitmap ve BitmapData sınıfları . . . 247
Piksellerde değişiklik yapma . . . 248
Bitmap verilerini kopyalama . . . 251
Bitmap verilerini sıkıştırma . . . 252
Gürültü işlevleri ile doku oluşturma . . . 253
Bitmap'leri kaydırma . . . 255
Mipmap oluşturma avantajından yararlanma . . . 256
Bitmap örneği: Animasyonlu dönen ay . . . 257
Bitmap görüntülerin senkronize olmayan kod çözme işlemi . . . 267

Bölüm 14: Görüntüleme nesnelerine filtre uygulama
Görüntüleme nesnelerine filtre uygulama temelleri . . . 270
Filtreler oluşturma ve uygulama . . . 271
Kullanılabilir görüntü filtreleri . . . 278
Görüntüleme nesnelerine filtre uygulama örneği: Filter Workbench . . . 294

Bölüm 15: Pixel Bender gölgelendiricileriyle çalışma
Pixel Bender gölgelendiricilerinin temelleri . . . 302
Gölgelendirici yükleme veya gömme . . . 304
Gölgelendirici meta verilerine erişme . . . 305
Gölgelendirici girdisi ve parametre değerlerini belirtme . . . 307
Gölgelendirici kullanma . . . 312

Bölüm 16: Film klipleriyle çalışma
Film kliplerinin temelleri . . . 324
MovieClip nesneleriyle çalışma . . . 325
Film klibi oynatımını denetleme . . . 325
ActionScript ile MovieClip nesneleri oluşturma . . . 328
Harici bir SWF dosyasını yükleme . . . 331
Film klibi örneği: RuntimeAssetsExplorer . . . 333

Bölüm 17: Hareket aralarıyla çalışma
Hareket Arası Temelleri . . . 337
Ara hareket komut dosyalarını kopyalama Flash uygulamasında . . . 338
Ara hareket komut dosyalarını birleştirme . . . 339
Animasyonu açıklama . . . 340
Filtreler ekleme . . . 342
Ara hareketi görüntüleme nesneleriyle ilişkilendirme . . . 344

Bölüm 18: Ters kinematikle çalışma

Ters Kinematik Temelleri 345
TK Armatürlerine Animasyon Uygulamaya Genel Bakış 346
TK armatürü hakkında bilgi alma 348
IK Taşıyıcısını Başlatma ve IK Taşıyıcısının Hareketini Sınırlama 348
Bir IK Armatürünü hareket ettirme 349
Yayları Kullanma 350
TK Olaylarını Kullanma 350

Bölüm 19: Üç boyutlu (3B) çalışma

3B görüntüleme nesnelerinin temelleri 352
Flash Player ve AIR çalışma zamanında 3B görüntüleme nesnelerini anlama 353
3B görüntüleme nesneleri oluşturma ve taşıma 355
3B nesneleri 2B görünümde yansıtma 357
Örnek: Perspektif projeksiyonu 359
Karmaşık 3B dönüştürmeler gerçekleştirme 362
3B efektler için üçgenleri kullanma 365

Bölüm 20: Metinle Çalışmanın temelleri

Bölüm 21: TextField sınıfını kullanma

Metin görüntüleme 375
Metni seçme ve işleme 379
Metin girdisi yakalama 381
Metin girdisini kısıtlama 382
Metin biçimlendirme 383
Gelişmiş metin oluşturma 387
Statik metinle çalışma 389
TextField Örneği: Gazete stili metin biçimlendirme 390

Bölüm 22: Flash Metin Motoru'nu kullanma

Metin oluşturma ve görüntüleme 400
FTE'de Olay İşleme 404
Metin biçimlendirme 408
Fontlarla çalışma 412
Metni denetleme 415
Flash Text Engine örneği: Mizanpaj 421

Bölüm 23: Metin Mizanpajı Çerçevesini Kullanma

Metin Mizanpajı Çerçevesi'ne Genel Bakış 430
Metin Mizanpajı Çerçevesini Kullanma 431
TLF ile metin yapılandırma 436
TLF ile metin biçimlendirme 440
TLF ile metin içe aktarma ve dışa aktarma 441
TLF ile metin kaplarını yönetme 442
TLF ile metin seçimini, düzenlemeyi ve geri almayı etkinleştirme 443
TLF ile olay işleme 443
Metin içindeki görüntüleri konumlandırma 444

Bölüm 24: Sesle çalışma

Sesle çalışmanın temelleri ... 445
Ses mimarisini anlama ... 446
Harici ses dosyalarını yükleme ... 447
Gömülü seslerle çalışma ... 450
Akış ses dosyalarıyla çalışma ... 451
Dinamik olarak oluşturulmuş sesle çalışma ... 452
Sesleri çalma ... 454
Sesler yüklenirken ve çalınırken dikkate alınması gereken güvenlikle ilgili noktalar ... 458
Ses düzeyini denetleme ve yatay kaydırma ... 459
Ses meta verisiyle çalışma ... 460
Ham ses verilerine erişme ... 461
Ses girdisini yakalama ... 465
Ses örneği: Podcast Oynatıcısı ... 471

Bölüm 25: Videoyla çalışma

Video temelleri ... 478
Video biçimlerini anlama ... 479
Video sınıfını anlama ... 482
Video dosyalarını yükleme ... 483
Video oynatımını denetleme ... 484
Tam ekran modunda video oynatma ... 486
Video dosyalarını akışa alma ... 490
İşaret noktalarını anlama ... 490
Meta veri ve işaret noktaları için geri çağrı yöntemleri yazma ... 491
İşaret noktalarını ve meta verileri kullanma ... 496
NetStream etkinliğini izleme ... 506
Video dosyaları için gelişmiş başlıklar ... 509
Video örneği: Video Jukebox ... 511
Donanım hızlandırmalı gösterim için StageVideo sınıfını kullanma ... 517

Bölüm 26: Kameralar ile çalışma

Camera sınıfını anlama ... 526
Ekranda kamera içeriğini görüntüleme ... 527
Kamera uygulamanızı tasarlama ... 527
Bir kullanıcının kamerasına bağlanma ... 528
Kameraların takılı olduğunu doğrulama ... 528
Kamera erişimi için izinleri algılama ... 529
Kameranın video kalitesini en üst düzeye çıkarma ... 531
Kamera durumunu izleme ... 532

Bölüm 27: Dijital haklar yönetimini kullanma

Korumalı içerik iş akışını anlama ... 535
NetStream sınıfının DRM ilişkili üyeleri ve olayları ... 541
DRMStatusEvent sınıfını kullanma ... 543
DRMAuthenticateEvent sınıfını kullanma ... 544
DRMErrorEvent sınıfını kullanma ... 547

DRMManager sınıfını kullanma 548
DRMContentData sınıfını kullanma 549
Adobe Access desteği için Flash Player'ı güncelleme 550
Bant Dışı Lisanslar 551
Etki alanı desteği 553
Etki alanı desteğini kullanarak şifrelenmiş içeriği yürütme 553
Lisans önizlemesi 553
İçeriği sunma 554
Open Source Media Framework (Açık Kaynak Medya Çerçevesi) 554

Bölüm 28: AIR'ye PDF içeriği ekleme
PDF Özelliklerini Saptama 557
PDF içeriği yükleme 558
Komut dosyası PDF içeriği 558
AIR'de PDF içeriği için bilinen sınırlamalar 560

Bölüm 29: Kullanıcı etkileşiminin temelleri
Kullanıcı girdisini yakalama 562
Odak yönetme 563
Girdi türlerini keşfetme 564

Bölüm 30: Klavye girdisi
Klavye girdisini yakalama 566
IME sınıfını kullanma 568
Sanal klavyeler 573

Bölüm 31: Fare girdisi
Fare girişi yakalama 581
Fare girdisi örneği: WordSearch 583

Bölüm 32: Dokunmatik, çoklu dokunmatik ve hareket girdisi
Dokunma girdisinin temelleri 588
Dokunma desteği keşfi 590
Touch olayını işleme 591
Dokun-sürükle 594
Gesture olayını işleme 595
Sorun Giderme 599

Bölüm 33: Kopyalama ve yapıştırma
Kopyala ve yapıştır işleminin temelleri 602
Sistem panosundan okuma ve sistem panosuna yazma 603
AIR'de HTML kopyalama ve yapıştırma 603
Pano veri biçimleri 605

Bölüm 34: Akselorometre girdisi
Akselerometre desteğini kontrol etme 611
Akselerometre değişikliklerini saptama 611

Bölüm 35: AIR uygulamasında taşıma ve bırakma
AIR'de sürükle ve bırak temelleri 614
Dışarı sürükleme hareketini destekleme 616
İçeri sürükleme hareketini destekleme 618
HTML'de sürükle ve bırak 621
Bir HTML öğesinden veri sürüklemek 624
Bir HTML öğesine veri sürükleme 625
Örnek: Varsayılan HTML içeri sürükleme davranışını geçersiz kılma 626
Uygulama dışı HTML sanal alanlarındaki dosya bırakmaları işleme 628
Dosya uzantılarını bırakma 629

Bölüm 36: Menülerle çalışma
Temel menü işlemleri 638
Yerel menüler oluşturma (AIR) 645
HTML'deki (AIR) içerik menüleri hakkında 647
Açılır pencere yerel menülerini görüntüleme (AIR) 648
Menü olaylarını işleme 649
Yerel menü örneği: Pencere ve uygulama menüsü (AIR) 650

Bölüm 37: AIR'de görev çubuğu simgeleri
Görev çubuğu simgeleri hakkında 654
Dock simgeleri 655
Sistem Tepsisi simgeleri 655
Pencere görev çubuğu simge ve düğmeleri 657

Bölüm 38: Dosya sistemiyle çalışma
FileReference sınıfını kullanma 659
AIR dosya sistemi API'sini kullanma 673

Bölüm 39: Yerel verileri saklama
Paylaşılan nesneler 709
Şifrelenmiş yerel depolama 718

Bölüm 40: AIR'de yerel SQL veritabanlarıyla çalışma
Yerel SQL veritabanları hakkında 722
Veritabanı oluşturma ve değiştirme 726
SQL veritabanı verilerini değiştirme 733
Senkronize ve senkronize olmayan veritabanı işlemlerini kullanma 761
Şifrelemeyi SQL veritabanlarıyla kullanma 766
SQL veritabanlarıyla çalışmak için stratejiler 783

Bölüm 41: Bayt dizileriyle çalışma
ByteArray okuma ve yazma 786
ByteArray örneği: Bir .zip dosyasını okuma 791

Bölüm 42: Ağ iletişimi ve iletişim temelleri
Ağ arabirimleri 799
Ağ bağlanabilirliği değişimleri 800
DNS (Alan Adı Sistemi) kayıtları 802

Bölüm 43: Soketler
TCP soketleri 805
UDP soketleri (AIR) 816
IPv6 adresleri 818

Bölüm 44: HTTP iletişimi
Harici verileri yükleme 820
Web hizmeti talepleri 829
Başka bir uygulamada URL açma 836

Bölüm 45: Diğer Flash Player ve AIR örneklerine bağlanma
LocalConnection sınıfı hakkında 839
İki uygulama arasında mesaj gönderme 841
Farklı etki alanlarında ve diğer AIR uygulamalarındaki içeriğe bağlanma 843

Bölüm 46: AIR uygulamasında yerel işlemlere bağlanma
Yerel işlem iletişimlerine genel bakış 846
Yerel işlem başlatma ve kapatma 847
Yerel işlemle iletişim kurma 848
Yerel işlem iletişimi için güvenlik etmenleri 849

Bölüm 47: Harici API'yi kullanma
Harici API'yi kullanma temelleri 851
Harici API gereksinimleri ve avantajları 853
ExternalInterface sınıfını kullanma 854
Harici API örneği: Bir web tarayıcısında ActionScript ve JavaScript arasındaki iletişim 858

Bölüm 48: AIR'de XML imza doğrulama
XML imza doğrulama temelleri 865
XML imzaları hakkında 870
IURIDereferencer arabirimini uygulama 872

Bölüm 49: İstemci sistemi ortamı
İstemci sistemi ortamının temelleri 880
System sınıfını kullanma 881
Capabilities sınıfını kullanma 882
Yetenekler örneği: Sistem yeteneklerini algılama 883

Bölüm 50: AIR uygulamasını başlatma ve sonlandırma
Uygulama başlatma 887
Komut satırı argümanlarını yakalama 888
Kullanıcının oturum açması sırasında bir AIR uygulaması başlatmak 892
Tarayıcıda bir AIR uygulaması başlatma 893
Uygulama sonlandırma 895

Bölüm 51: AIR çalışma zamanı ve işletim sistemi bilgileriyle çalışma
Dosya ilişkilendirmelerini yönetme 897
Çalıştırma zamanı sürümü ve yama düzeyini alma 898

AIR özelliklerini saptama 898
Kullanıcı varlığını izleme 898

Bölüm 52: AIR yerel pencereleriyle çalışma
AIR'de yerel pencerelerin temelleri 900
Pencereler oluşturma 907
Pencereleri yönetme 916
Pencere olaylarını dinleme 925
Tam ekran pencereleri görüntüleme 926

Bölüm 53: AIR'de görüntü ekranları
AIR'de görüntüleme ekranlarının temelleri 929
Ekranları numaralandırma 930

Bölüm 54: Yazdırma
Yazdırma temelleri 933
Sayfa yazdırma 934
Flash çalışma zamanı görevleri ve sistem yazdırma 935
Boyutu, ölçeği ve yönlendirmeyi ayarlama 937
Gelişmiş yazdırma teknikleri 940
Yazdırma örneği: Çoklu sayfa yazdırma 942
Yazdırma örneği: Ölçekleme, kırpma ve yanıtlama 944
Yazdırma örneği: Sayfa düzeni ve yazdırma seçenekleri 946

Bölüm 55: Geolocation
Coğrafi konum değişikliklerini saptama 948

Bölüm 56: Uygulamaları uluslararasılaştırma
Uygulamaları uluslararasılaştırma temelleri 952
flash.globalization paketine genel bakış 953
Yerel ayarı belirleme 954
Rakamları biçimlendirme 956
Para birimi değerlerini biçimlendirme 958
Tarihleri ve saatleri biçimlendirme 960
Dizeleri sıralama ve harmanlama 962
Büyük/Küçük harf dönüştürme 963
Örnek: Bir stok onaylayıcı uygulamasını uluslararasılaştırma 964

Bölüm 57: Uygulamaları yerelleştirme
Bir yerel ayar seçme 969
Flex içeriğini yerelleştirme 970
Flaş içeriğini yerelleştirme 970
AIR uygulamalarını yerelleştirme 970
Tarihleri, saatleri ve para birimlerini yerelleştirme 970

Bölüm 58: HTML ortamı hakkında
HTML ortamına genel bakış 973
AIR ve Webkit 976

Bölüm 59: AIR uygulamasında HTML ve JavaScript programlama

HTMLLoader sınıfı hakkında ... 991
Güvenlikle ilişkili JavaScript hatalarını önleme ... 993
AIR API sınıflarına JavaScript'ten erişme ... 997
AIR'deki URL'ler hakkında ... 999
ActionScript nesnelerini JavaScript için kullanılabilir duruma getirme ... 999
HTML DOM ve JavaScript nesnelerine ActionScript'ten erişme ... 1001
SWF içeriğini HTML'e gömme ... 1002
HTML sayfasında ActionScript kütüphaneleri kullanma ... 1005
Date ve RegExp nesnelerini dönüştürme ... 1007
HTML stil sayfasını ActionScript'ten değiştirme ... 1007
Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği ... 1008

Bölüm 60: AIR HTML Kabı için Komut Dosyası Oluşturma

HTMLLoader nesnelerinin görüntü özellikleri ... 1013
HTML içeriğini kaydırma ... 1016
HTML geçmiş listesine erişme ... 1017
HTML içeriği yüklenirken kullanılan kullanıcı aracısını ayarlama ... 1017
HTML içeriğinde kullanım için karakter kodlamasını ayarlama ... 1018
HTML içeriği için tarayıcı benzeri kullanıcı arabirimleri tanımlama ... 1018
HTMLLoader sınıfının alt sınıflarını oluşturma ... 1029

Bölüm 61: AIR'de HTML bağlantılı olayları işleme

HTMLLoader olayları ... 1030
DOM olaylarını ActionScript ile işleme ... 1031
Yakalanmayan JavaScript istisnalarına yanıt verme ... 1031
Çalışma zamanı olaylarını JavaScript'le işleme ... 1033

Bölüm 62: Mobil uygulamalarda HTML içeriğini görüntüleme

StageWebView nesneleri ... 1036
İçerik ... 1036
Gezinme olayları ... 1038
Geçmiş ... 1039
Odak ... 1040
Bitmap yakalama ... 1042

Bölüm 63: Eş zamanlılık için worker (işçi) öğelerini kullanma

Worker (işçi) öğelerini ve eş zamanlılığı anlama ... 1044
Worker (işçi) öğeleri oluşturma ve yönetme ... 1045
Worker (işçi) öğeleri arasında iletişim ... 1048

Bölüm 64: Güvenlik

Flash Platform güvenliğine genel bakış ... 1053
Güvenlik sanal alanları ... 1055
İzin denetimleri ... 1059
Ağ iletişim API'lerini kısıtlama ... 1067
Tam ekran modu güvenliği ... 1069
Tam ekran etkileşimli modu güvenliği ... 1070

İçerik yükleme 1071
Çapraz komut dosyası oluşturma 1074
Veri olarak yüklenen ortama erişme 1077
Veri yükleme 1080
Güvenlik etki alanına içe aktarılan SWF dosyalarından gömülü içerik yükleme 1083
Eski içerikle çalışma 1083
LocalConnection izinlerini ayarlama 1084
Giden URL erişimini denetleme 1084
Paylaşılan nesneler 1086
Kamera, mikrofon, pano, fare ve klavye erişimi 1087
AIR güvenliği 1088

Bölüm 65: ActionScript Örnekleri Nasıl Kullanılır?
Örnek türleri 1108
ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması 1110
ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması 1111
ActionScript 3.0 örneklerinin taşınabilir cihazlarda çalıştırılması 1112

Bölüm 66: yerel veritabanlarında SQL desteği
Desteklenen SQL söz dizimi 1117
Veri türü desteği 1137

Bölüm 67: SQL hata ayrıntı mesajları, kimlikler ve argümanlar

Bölüm 68: Adobe Grafikleri Çevirici Dili (AGAL)
AGAL bayt kodu biçimi 1146

Bölüm 1: Tarih ve saatlerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Zamanlama her şey olmayabilir ancak yazılım uygulamalarında önemli bir faktördür. ActionScript 3.0, takvim günlerinin, saatlerinin ve zaman aralıklarının yönetilmesi için güçlü yollar sağlar. Bu zamanlama işlevinin çoğunu iki ana sınıf sağlar: Date sınıfı ve flash.utils paketindeki yeni Timer sınıfı.

Tarih ve saatler, ActionScript programlarında kullanılan yaygın bir bilgi türüdür. Örneğin, diğer olasılıklar arasında haftanın geçerli gününü bilmeniz veya bir kullanıcının belirli bir ekranda ne kadar zaman harcadığını ölçmeniz gerekebilir. ActionScript'te, tarih ve saat bilgileri dahil olmak üzere, zaman içindeki tek bir anı temsil etmek için Date sınıfını kullanabilirsiniz. Date örneği içinde, her bir tarih ve saat birimi için değerler vardır. Bu değerler arasında, yıl, ay, gün, haftanın günü, saat, dakika, saniye, milisaniye ve saat dilimi yer alır. Daha ileri düzey kullanıcılar için ActionScript, belirli bir gecikmeden sonra veya art arda aralıklarla eylemleri gerçekleştirmek için kullanabildiğiniz Timer sınıfını da içerir.

Daha fazla Yardım konusu

Date

flash.utils.Timer

Takvim tarih ve saatlerini yönetme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'daki tüm takvim tarihi ve saati yönetimi işlevleri, üst düzey Date sınıfında yoğunlaşmıştır. Date sınıfı, Eşitlenmiş Evrensel Saat'te (UTC) veya bir saat dilimine özel yerel saatte tarih ve saatleri işlemenize olanak sağlayan yöntemleri ve özellikleri içerir. UTC, temelde Greenwich Saati (GMT) ile aynı olan standart bir saat tanımıdır.

Date nesneleri oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Date sınıfı, tüm çekirdek sınıfların en çok yönlü yapıcı yöntemlerinden birini sağlar. Dört farklı şekilde bunu çağırabilirsiniz.

İlk olarak, herhangi bir parametre verilmediyse, `Date()` yapıcısı, saat diliminizi esas alarak yerel saatte geçerli tarih ve saati içeren bir Date nesnesini döndürür. Aşağıda buna bir örnek verilmiştir:

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

İkinci olarak, tek bir sayısal parametre verildiyse, `Date()` yapıcısı bunu 1 Ocak 1970'ten bu yana geçen milisaniye sayısı olarak değerlendirir ve karşılık gelen bir Date nesnesini döndürür. İlettiğiniz milisaniye değerinin, UTC'de 1 Ocak 1970'ten bu yana geçen milisaniye sayısı olarak değerlendirildiğini unutmayın. Ancak değerleri almak ve görüntülemek için UTC'ye özel yöntemleri kullanmadığınız sürece Date nesnesi, yerel saat diliminizdeki değerleri gösterir. Tek bir milisaniye parametresi kullanarak yeni bir Date nesnesi oluştururseanız, yerel saatiniz ile UTC arasındaki saat dilimi farkını dikkate aldığınızdan emin olun. Aşağıdaki deyimler, UTC'de 1 Ocak 1970 gününün gece yarısına ayarlanmış bir Date nesnesi oluşturur:

```
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);
```

Üçüncü olarak, `Date()` yapıcısına birden çok sayısal parametre iletebilirsiniz. Yapıcı bu parametreleri sırayla yıl, ay, gün, saat, dakika, saniye ve milisaniye olarak değerlendirir ve karşılık gelen Date nesnesini döndürür. Bu girdi parametrelerinin UTC'de değil yerel saatte olduğu varsayılır. Aşağıdaki deyimler, yerel saatte 1 Ocak 2000 başlangıcında gece yarısına ayarlanmış bir Date nesnesi alır:

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

Dördüncü olarak, `Date()` yapıcısına tek bir dize parametresi iletebilirsiniz. Yapıcı bu dizeyi tarih veya saat bileşenlerine ayrıştırmayı dener ve sonra karşılık gelen bir Date nesnesini döndürür. Bu yaklaşımı kullanırsanız, ayrıştırma hatalarını yakalamak için `Date()` yapıcısının bir `try..catch` bloğunun içine alınması iyi bir fikirdir. `Date()` yapıcı, (Adobe Flash Platformu için ActionScript 3.0 Başvurusu öğesinde listelenen) birkaç farklı dize biçimi kabul eder. Aşağıdaki deyim, bir dize değeri kullanarak yeni bir Date nesnesini başlatır:

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

`Date()` yapıcısı dize parametresini başarıyla ayrıştıramazsa, istisna atmaz. Ancak sonuçta elde edilen Date nesnesi geçersiz bir tarih değeri içerir.

Zaman birimi değerlerini alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Date sınıfının özelliklerini veya yöntemlerini kullanarak bir Date nesnesi içindeki çeşitli zaman birimlerinin değerlerini ayıklayabilirsiniz. Aşağıdaki özelliklerin her biri, size Date nesnesindeki bir zaman biriminin değerini verir:

- `fullYear` özelliği
- Ocak için 0 ile başlayıp Aralık için 11 olacak şekilde sayısal biçimde olan `month` özelliği
- 1 ile 31 aralığında, ayın gününü ifade eden takvim numarası olan `date` özelliği
- Pazar için 0 olmak üzere sayısal biçimde haftanın günü olan `day` özelliği
- 0 ile 23 aralığı içinde `hours` özelliği
- `minutes` özelliği
- `seconds` özelliği
- `milliseconds` özelliği

  Aslına bakılırsa, Date sınıfı bu değerlerin her birinin alınması için size birçok yol sağlar. Örneğin, Date nesnesinin ay değerini dört farklı şekilde alabilirsiniz:

- `month` özelliği
- `getMonth()` yöntemi
- `monthUTC` özelliği
- `getMonthUTC()` yöntemi

  Bu dört yol da verimlilik açısından eşdeğerdir, böylece uygulamanıza en iyi şekilde uyan yaklaşımı kullanabilirsiniz.

  Listelenen özelliklerin tümü, toplam tarih değerinin bileşenlerini temsil eder. Örneğin, milisaniye değeri 1000'e ulaştığında saniye değeri 1 artıp milliseconds özelliği 0 değerine getirildiğinden, milisaniye değeri asla 999'dan büyük olmaz.

Date nesnesinin değerini, 1 Ocak 1970 (UTC) tarihinden bu yana geçen milisaniye sayısı olarak almak isterseniz, `getTime()` yöntemini kullanabilirsiniz. Bunun karşılığı olan `setTime()` yöntemi, 1 Ocak 1970'ten (UTC) bu yana geçen milisaniye sayısını kullanarak varolan bir Date nesnesinin değerini değiştirmenize olanak sağlar.

Tarih ve saat aritmetiği yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Date sınıfıyla tarih ve saatlerde toplama ve çıkarma işlemleri yapabilirsiniz. Tarih değerleri dahili olarak milisaniye cinsinde saklanır, böylece diğer değerleri Date nesnelerine eklemeden veya Date nesnelerinden çıkarmadan önce milisaniyeye dönüştürmeniz gerekir.

Uygulamanız çok sayıda tarih ve saat aritmetiği uygulayacaksa, aşağıdaki gibi, ortak zaman birimi değerlerini milisaniye cinsinde tutan sabitler oluşturulması yararlı olabilir:

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

Artık standart zaman birimlerini kullanarak tarih aritmetiği yapmak çok kolay. Aşağıdaki kod, `getTime()` ve `setTime()` yöntemlerini kullanarak geçerli saatten bir saate bir tarih değeri ayarlar:

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

Tarih değeri ayarlamanın başka bir yolu, tek bir milisaniye parametresi kullanarak yeni bir Date nesnesi oluşturulmasıdır. Örneğin, aşağıdaki kod, başka bir tarihi hesaplamak için bir tarihe 30 gün ekler:

```
// 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));
```

Ardından, 30 günlük zamanı temsil etmesi için `millisecondsPerDay` sabiti 30 ile çarpılır ve sonuç `invoiceDate` değerine eklenip `dueDate` değerini ayarlamak için kullanılır.

Saat dilimleri arasında dönüştürme yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tarih ve saat aritmetiği, tarihleri bir saat diliminden diğerine dönüştürmek istediğinizde kullanışlıdır. Date nesnesinin saat diliminin UTC'den farklı olduğu dakika cinsinden değeri döndüren `getTimezoneOffset()` yöntemi de aynı işleve sahiptir. Tüm saat dilimleri çift saatlik artışa ayarlanmadığından (bazıları yakın saat dilimlerinden yarım saat farklıdır), dakika cinsinden bir değer döndürür.

Aşağıdaki örnek, bir tarihi yerel saatten UTC'ye dönüştürmek için saat dilimi farkını kullanır. İlk olarak saat dilimi değerini milisaniye cinsinden hesaplayıp sonra Date değerini o miktara göre ayarlayarak dönüştürmeyi gerçekleştirir:

```
// 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);
```

Zaman aralıklarını denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe Flash CS4 Professional'ı kullanarak uygulamalar geliştirdiğinizde, uygulamanızda sabit, kare kare ilerleme sağlayan zaman çizelgesine erişiminiz olur. Yalnızca ActionScript projelerindeyse, diğer zamanlama mekanizmalarını kullanmanız gerekir.

Döngüler ve zamanlayıcılar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazı programlama dillerinde, `for` veya `do..while` gibi döngü deyimlerini kullanarak kendi zamanlama şemalarınızı tasarlamanız gerekir.

Döngü deyimleri genellikle yerel makinenin izin verdiği ölçüde hızlı şekilde çalıştırılır, başka bir deyişle, uygulama bazı makinelerde hızlı çalışırken, başka makinelerde yavaş çalışabilir. Uygulamanız için tutarlı bir zamanlama aralığı gerekirse, uygulamanızı gerçek bir takvim veya saat zamanına bağlamanız gerekir. Oyun, animasyon ve gerçek zamanlı denetleyiciler gibi birçok uygulama için makineden makineye tutarlı olan normal, zaman odaklı şekilde ilerleyen mekanizmalar gerekir.

ActionScript 3.0 Timer sınıfı güçlü bir çözüm sunar. Timer sınıfı, belirtilen zaman aralığına her ulaşıldığında ActionScript 3.0 olay modelini kullanarak zamanlayıcı olaylarını gönderir.

Timer sınıfı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da zamanlama işlevlerini işlemenin tercih edilen yolu, bir aralığa her ulaşıldığında olayları göndermek için kullanılabilen Timer sınıfının (flash.utils.Timer) kullanılmasıdır.

Bir zamanlayıcıyı başlatmak için, ilk olarak bir Timer sınıfı örneğini oluşturarak bu sınıfa ne sıklıkla bir zamanlayıcı olayı oluşturacağını ve durmadan önce kaç defa bunu yapacağını bildirirdiniz.

Örneğin, aşağıdaki kod, her saniye bir olay gönderen ve 60 saniye boyunca bu işleme devam eden bir Timer örneğini oluşturur:

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

Timer nesnesi, belirli bir aralığa her ulaşıldığında bir TimerEvent nesnesi gönderir. TimerEvent nesnesinin olay türü `timer` (`TimerEvent.TIMER` sabitiyle tanımlı) olur. TimerEvent nesnesi, standart bir Event nesnesiyle aynı özellikleri içerir.

Timer örneği sabit sayıda bir aralığa ayarlanırsa, son aralığa ulaştığında bir `timerComplete` olayı da (`TimerEvent.TIMER_COMPLETE` sabitiyle tanımlı) gönderir.

Aşağıda, Timer sınıfını eylem halinde gösteren küçük bir örnek uygulama yer almaktadır:

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

ShortTimer sınıfı oluşturulduğunda, bu sınıf beş saniye boyunca saniyede bir defa ilerleyecek bir Timer örneği oluşturur. Daha sonra zamanlayıcıya iki dinleyici ekler: biri ilerlemelerin her birini dinler, diğeri de `timerComplete` olayını dinler.

Ardından zamanlayıcı ilerlemesini başlatır ve bu noktadan itibaren bir saniyelik aralıklarla `onTick()` yöntemi çalıştırılır.

`onTick()` yöntemi geçerli ilerleme sayısını gösterir. Beş saniye geçtikten sonra, `onTimerComplete()` yöntemi çalıştırılarak zamanın dolduğunu bildirir.

Bu örneği çalıştırdığınızda, konsolunuzda veya izleme pencerenizde, saniyede birer satır hızında şu satırların görüntülenmesi gerekir:

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

flash.utils paketinde zamanlama işlevleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, ActionScript 2.0'dakine benzer çok sayıda zamanlama işlevi içerir. Bu işlevler, flash.utils paketinde paket düzeyinde işlevler olarak sağlanır ve ActionScript 2.0'daki gibi çalışır.

İşlev	Açıklama
`clearInterval(id:uint):void`	Belirtilen bir `setInterval()` çağrısını iptal eder.
`clearTimeout(id:uint):void`	Belirtilen bir `setTimeout()` çağrısını iptal eder.
`getTimer():int`	Adobe® Flash® Player veya Adobe® AIR™ başlatıldıktan sonra geçen milisaniye sayısını döndürür.
`setInterval(closure:Function, delay:Number, ... arguments):uint`	Bir işlevi belirtilen bir aralıkta (milisaniye olarak) çalıştırır.
`setTimeout(closure:Function, delay:Number, ... arguments):uint`	Belirtilen bir işlevi belirtilen bir gecikmeden sonra (milisaniye olarak) çalıştırır.

Bu işlevler, geriye doğru uyumluluk için ActionScript 3.0'da kalmıştır. Adobe, bunları yeni ActionScript 3.0 uygulamalarında kullanmanızı önermez. Genellikle uygulamalarınızda Timer sınıfının kullanılması hem daha kolay hem de daha verimlidir.

Tarih ve saat örneği: Basit analog saat

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Basit bir analog saat örneği bu iki tarih ve zaman kavramını örnekler.

- Geçerli tarih ve saati alma ve saat, dakika ve saniye değerlerini ayıklama
- Uygulamanın hızını ayarlamak için bir Timer öğesini kullanma

  Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. SimpleClock uygulama dosyalarını Samples/SimpleClock klasörü içinde bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
SimpleClockApp.mxml veya SimpleClockApp.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/simpleclock/SimpleClock.as	Ana uygulama dosyası.
com/example/programmingas3/simpleclock/AnalogClockFace.as	Yuvarlak bir saat yüzü çizer ve saati esas alarak akrep, yelkovan ve saniye göstergesini çizer.

SimpleClock sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Saat örneği basit olsa da, ileride kolayca genişletebilmeniz için basit uygulamaların da organize edilmesi iyi bir fikirdir. Bu nedenle SimpleClock uygulaması, açılış ve zaman saptama görevlerini işlemek için SimpleClock sınıfını kullanır ve sonra da gerçekten zamanı göstermek için de AnalogClockFace adında başka bir sınıfı kullanır.

Aşağıda, SimpleClock sınıfını tanımlayıp başlatan kod yer almaktadır (Flash sürümünde, bunun yerine SimpleClock öğesinin Sprite sınıfını genişlettiğini unutmayın):

```
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;
```

Bu sınıf iki önemli özelliğe sahiptir:

- AnalogClockFace sınıfının bir örneği olan `face` özelliği
- Timer sınıfının bir örneği olan `ticker` özelliği

  SimpleClock sınıfı varsayılan bir yapıcıyı kullanır. `initClock()` yöntemi gerçek kurulum çalışmasıyla ilgilenerek saat yüzünden başlar ve Timer örneği ilerlemesini başlatır.

Saat yüzünü oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SimpleClock kodundaki sonraki satırlar, saati görüntülemek için kullanılan saat yüzünü oluşturur:

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

Yüzün boyutu, `initClock()` yöntemine iletilebilir. Herhangi bir `faceSize` değeri iletilmezse, varsayılan 200 piksel boyut kullanılır.

Daha sonra uygulama, yüzü başlatır ve DisplayObject sınıfından miras alınan `addChild()` yöntemini kullanarak bu yüzü görüntüleme listesine ekler. Ve saat yüzünü bir defa görüntülemek için `AnalogClockFace.draw()` yöntemini çağırarak geçerli saati gösterir.

Zamanlayıcıyı başlatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Saat yüzü oluşturulduktan sonra, `initClock()` yöntemi bir zamanlayıcı ayarlar:

```
// 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();
```

İlk olarak bu yöntem, saniyede bir defa (her 1000 milisaniyede) olay gönderecek bir Timer örneği başlatır. `Timer()` yapıcısına herhangi bir `repeatCount` parametresi iletilmediğinden, Timer sınırsız sayıda yinelemeyi sürdürür.

`timer` olayı alındığında, `SimpleClock.onTick()` yöntemi saniyede bir defa çalıştırılır:

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

`AnalogClockFace.draw()` yöntemi yalnızca saat yüzünü ve akrep, yelkovan ve saniye göstergesini çizer.

Geçerli saati görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

AnalogClockFace sınıfındaki kodun büyük bir kısmı, saat yüzünün görüntüleme öğelerinin ayarlanmasını kapsar. AnalogClockFace başlatıldığında, dairesel bir anahat çizer, her saat işaretine sayısal bir metin etiketi yerleştirir ve sonra saatteki akrep, yelkovan ve saniye göstergesinin her biri için birer tane olmak üzere üç Shape nesnesi oluşturur.

SimpleClock uygulaması çalışmaya başladıktan sonra, aşağıdaki gibi her saniye `AnalogClockFace.draw()` yöntemini çağırır:

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

Bu yöntem geçerli saati bir değişkene kaydeder, böylece akrep, yelkovan ve saniye göstergesi çizilirken işlemin ortasında saat değişemez. Ardından aşağıda gösterildiği gibi, akrep, yelkovan ve saniye göstergesini görüntülemek için `showTime()` yöntemini çağırır:

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

İlk olarak bu yöntem, geçerli zamanın saat, dakika ve saniye değerlerini ayıklar. Daha sonra akrep, yelkovan ve saniye göstergesinin her birinin açısını hesaplamak için bu değerleri kullanır. Saniye göstergesi tam bir dönüşü 60 saniyede gerçekleştirdiğinden, bu gösterge her saniyede 6 derece (360/60) dönüş yapar. Yelkovan da her dakika aynı derece dönüş yapar.

Akrep de her dakika güncellenir, böylece dakikalar ilerledikçe o da aynı ilerlemeyi gösterebilir. Her saatte 30 derece (360/12) dönüş yapmanın yanı sıra her dakika da yarım derece döner (60 dakika/30 derece).

Bölüm 2: Dizelerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

String sınıfı, metin dizeleriyle çalışmanıza olanak sağlayan yöntemleri içerir. Dizeler birçok nesneyle çalışılmasında önemlidir. Burada açıklanan yöntemler, TextField, StaticText, XML, ContextMenu ve FileReference nesneleri gibi nesnelerde kullanılan dizelerle çalışılmasında kullanışlıdır.

Dizeler, karakterler sırasıdır. ActionScript 3.0, ASCII ve Unicode karakterlerini destekler.

Daha fazla Yardım konusu

Dize

RegExp

parseFloat()

parseInt()

Dizelerin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Programlama dilinde dize, bir metin değeridir, başka bir deyişle, harf, sayı ve diğer karakterlerin tek bir değer olarak birleştirilmiş sırasıdır. Örneğin, bu kod satırı, String veri türünde bir değişken oluşturur ve bu değişkene değişmez bir dize değeri atar:

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

Bu örnekte gösterildiği gibi, ActionScript'te, metni tırnak veya kesme işareti içine alarak bir dize değerini belirtebilirsiniz. Aşağıda başka dize örneklerine de yer verilmiştir:

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

ActionScript'te bir metin parçasını işlediğinizde bir dize değeriyle çalışıyorsunuz demektir. ActionScript String sınıfı, metin değerleriyle çalışmak için kullanabildiğiniz veri türüdür. String örnekleri genellikle özellikler, yöntem parametreleri ve diğer birçok ActionScript sınıfındaki öğeler için kullanılır.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesinde, dizilerle ilgili karşınıza çıkacak olan önemli terimler bulunmaktadır:

ASCII Bilgisayar programlarında metin karakterlerinin ve sembollerin temsil edilmesine yönelik bir sistem. ASCII sistemi, 26 harfli İngilizce alfabesini ve sınırlı bir ek karakter kümesini destekler.

Karakter Metin verisinin en küçük birimi (tek bir harf veya sembol)

Bitiştirme Birden çok dize değerini birbirine ekleyip bitiştirerek yeni bir dize değeri oluşturma.

Boş dize `""` şeklinde yazılan, herhangi bir metin, boşluk veya karakter içermeyen bir dize. Boş dize değeri, null değerine sahip bir String değişkeninden farklıdır—boş dize herhangi bir karakter içermeyen bir String örneğine sahipken, null String değeri, kendisine bir String örneği atanmamış değişkendir.

Dize Bir metin değeri (karakterler sırası).

Dize değişmezi (veya "değişmez dize") Kodda açıkça yazılan, tırnak veya kesme işareti içine alınmış bir dize değeri.

Alt dize Başka bir dizenin parçası olan bir dize.

Unicode Bilgisayar programlarında metin karakterlerinin ve sembollerin temsil edilmesine yönelik standart bir sistem. Unicode sistemi, herhangi bir yazma sisteminde tüm karakterlerin kullanılmasına olanak sağlar.

Dizeler oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da dize (metin) verilerini temsil etmek için String sınıfı kullanılır. ActionScript dizeleri hem ASCII hem de Unicode karakterlerini destekler. Dize oluşturmanın en basit yolu, bir dize değişmezinin kullanılmasıdır. Bir dize değişmezi bildirmek için tırnak işareti (`"`) veya kesme işareti (`'`) karakterlerini kullanın. Örneğin, şu iki dize eşdeğerdir:

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

Ayrıca aşağıdaki gibi, `new` operatörünü kullanarak da bir dize bildirebilirsiniz:

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

Aşağıdaki iki dize eşdeğerdir:

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

Kesme işareti (`'`) sınırlayıcıları ile tanımlanmış bir dize değişmesi içinde kesme işareti (`'`) kullanmak için, ters eğik çizgi karakterini (`\`) kullanın. Aynı şekilde, tırnak işareti (`"`) sınırlayıcıları ile tanımlanmış bir dize değişmesi içinde tırnak işareti (`"`) kullanmak için de ters eğik çizgi karakterini (`\`) kullanın. Aşağıdaki iki dize eşdeğerdir:

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

Aşağıdaki gibi, bir dize değişmezinde bulunan kesme işaretini veya tırnak işaretini esas alarak kesme işareti ya da tırnak işareti kullanmayı seçebilirsiniz:

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

ActionScript'in düz kesme işareti (`'`) ile sola veya sağa eğik kesme işaretini (`'`ya da `'`) ayırt ettiğini unutmayın. Tırnak işaretinde de aynı durum geçerlidir. Dize değişmezlerini tanımlamak için düz tırnak işaretlerini kullanın. Başka bir kaynaktan ActionScript'e metin yapıştırırken, doğru karakterleri kullandığınızdan emin olun.

Aşağıdaki tabloda gösterildiği gibi, dize değişmezlerinde diğer karakterleri tanımlamak için ters eğik çizgi karakterini (`\`) kullanabilirsiniz:

Kaçış sırası	Karakter
`\b`	Backspace
`\f`	Sonraki sayfaya geçme
`\n`	Yeni satır
`\r`	Satır başı
`\t`	Sekme
`\u`***nnnn***	*nnnn* onaltılık sayısıyla karakter kodu belirtilmiş Unicode karakteri; örneğin, `\u263a` gülen yüz karakteridir.
`\\x`***nn***	*nn* onaltılık sayısıyla karakter kodu belirtilmiş ASCII karakteri
`\'`	Kesme işareti
`\"`	Tırnak işareti
`\\`	Tek ters eğik çizgi karakteri

length özelliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Her dize, dizede bulunan karakter sayısına eşit olan bir `length` özelliğine sahiptir:

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

Aşağıdaki örnekte gösterildiği gibi, boş dize de null değerine sahip dize de 0 uzunluğuna sahiptir:

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

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

Dizelerdeki karakterlerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizedeki her karakter, dizede bir dizin konumuna sahiptir (tam sayı). Birinci karakterin dizin konumu 0'dır. Örneğin, aşağıdaki dizede, `y` karakteri 0 konumunda ve `w` karakteri de 5 konumundadır:

```
"yellow"
```

Bu örnekte olduğu gibi, `charAt()` yöntemini ve `charCodeAt()` yöntemini kullanarak çeşitli konumlardaki karakterleri teker teker inceleyebilirsiniz:

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

Bu kodu çalıştırdığınızda şu çıktı üretilir:

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

Ayrıca aşağıdaki örnekte gösterildiği gibi, `fromCharCode()` yöntemini kullanarak bir dizeyi tanımlamak için de karakter kodlarını kullanabilirsiniz:

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

Dizeleri harmanlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizeleri harmanlamak için şu operatörleri kullanabilirsiniz: `<`, `<=`, `!=`, `==`, `=>` ve `>`. Bu operatörler, aşağıdaki örnekte gösterildiği şekilde, `if` ve `while` gibi koşul deyimleriyle kullanılabilir:

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

Bu operatörler dizelerle kullanıldığında, ActionScript uygulaması aşağıda olduğu gibi, dizedeki her karakterin karakter kodu değerini dikkate alarak karakterleri soldan sağa karşılaştırır:

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

Aşağıdaki örnekte gösterildiği gibi, dizeleri birbiriyle ve diğer nesne türleriyle harmanlamak için `==` ve `!=` operatörlerini kullanın:

```
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
```

Diğer nesnelerin dize halinde temsilini alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Herhangi türdeki bir nesnenin String temsilini alabilirsiniz. Tüm nesneler bu amaçla bir `toString()` yöntemine sahiptir:

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

String nesneleri ile dize olmayan nesnelerin birleşimiyle `+` bitiştirme operatörünü kullandığınızda, `toString()` yöntemini kullanmanız gerekir. Bitiştirmeyle ilgili ayrıntılar için bir sonraki bölüme bakın.

`String()` genel işlevi, belirli nesne için `toString()` yöntemini çağıran nesne tarafından döndürülen değerle aynı değeri döndürür.

Dizeleri bitiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizelerin bitiştirilmesi, iki dizenin alınıp sırayla birbirine eklenmesi anlamına gelir. Örneğin, iki dizeyi bitiştirmek için `+` operatörünü kullanabilirsiniz:

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

Aşağıdaki örnekte gösterildiği gibi, aynı sonucu üretmek için `+=` operatörünü de kullanabilirsiniz:

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

Ayrıca, String sınıfı şu şekilde kullanılabilen bir `concat()` yöntemini içerir:

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

`+` operatörünü (veya `+=` operatörünü) bir String nesnesiyle ve dize *olmayan* bir nesneyle kullanırsanız, bu örnekte gösterildiği gibi, ActionScript ifadeyi değerlendirmek için dize olmayan nesneyi bir String nesnesine dönüştürür:

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

Ancak aşağıdaki örnekte gösterildiği gibi, `+` operatörü için bağlam sağlamak üzere gruplandırma için parantezleri kullanabilirsiniz:

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

Dizelerdeki alt dizeleri ve desenleri bulma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Alt dizeler, bir dize içindeki sıralı karakterlerdir. Örneğin, `"abc"` dizesi şu alt dizelere sahiptir: `""`, `"a"`, `"ab"`, `"abc"`, `"b"`, `"bc"`, `"c"`. Bir dizenin alt dizelerini bulmak için ActionScript yöntemlerini kullanabilirsiniz.

Desenler, ActionScript'te dizelerle veya normal ifadelerle tanımlanır. Örneğin, şu normal ifade belirli bir deseni tanımlar—ardından bir rakam karakterinin geldiği A, B ve C harfleri (normal ifade sınırlayıcıları, eğik çizgilerdir):

```
/ABC\d/
```

ActionScript, dizelerdeki desenlerin bulunmasına ve bulunan eşleşmelerin yerini alan alt dizelerlerle değiştirilmesine yönelik yöntemler içerir. Bunlar ilerleyen bölümlerde açıklanmaktadır.

Normal ifadeler karmaşık desenleri tanımlayabilir. Daha fazla bilgi için, bkz. "Normal ifadeler kullanma" sayfa 76.

Karakter konumuna göre bir alt dizeyi bulma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`substr()` ve `substring()` yöntemleri birbirine benzer. Her ikisi de bir dizenin alt dizesini döndürür. Her ikisi de iki parametre alır. Her iki yöntemde de birinci parametre, belirli bir dizedeki başlangıç karakterinin konumudur. Ancak, `substr()` yönteminde ikinci parametre, döndürülecek alt dizenin *uzunluğuyken*, `substring()` yönteminde ikinci parametre, alt dizenin *sonundaki* karakterin (döndürülen dizeye dahil edilmeyen) konumudur. Bu örnek, bu iki yöntem arasındaki farkı gösterir:

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

`slice()` yöntemi, `substring()` yöntemine benzer şekilde çalışır. Parametre olarak iki tane negatif olmayan tam sayı verildiğinde tamamen aynı çalışır. Ancak, `slice()` yöntemi negatif tam sayıları parametre olarak alabilir ve bu durumda aşağıdaki örnekte gösterildiği gibi, karakter konumu dizenin sonundan alınır:

```
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
```

`slice()` yönteminin parametreleri olarak negatif olmayan tam sayılarla negatif tam sayıları birleştirebilirsiniz.

Eşleşen bir alt dizenin karakter konumunu bulma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örnekte gösterildiği gibi, bir dizedeki eşleşen alt dizeleri bulmak için `indexOf()` ve `lastIndexOf()` yöntemlerini kullanabilirsiniz:

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

`indexOf()` yönteminin büyük/küçük harf duyarlı olduğuna dikkat edin.

Aşağıdaki gibi, aramanın başlatılacağı dizedeki dizin konumunu belirtmek için ikinci bir parametre belirtebilirsiniz:

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

`lastIndexOf()` yöntemi, dizedeki alt dizenin en son geçtiği yeri bulur:

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

`lastIndexOf()` yöntemine ikinci bir parametre dahil ederseniz, arama geriye doğru çalışacak şekilde (sağdan sola) dizedeki dizin konumundan yürütülür:

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

Alt dizelerin sınırlayıcıyla bölünmüş bir dizisini oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Alt dizelerin sınırlayıcı esas alınarak bölünmüş bir dizisini oluşturmak için `split()` yöntemini kullanabilirsiniz. Örneğin, virgül sınırlı veya sekme sınırlı dizeyi birden çok dizeye bölebilirsiniz.

Aşağıdaki örnekte, sınırlayıcı olarak ve (&) karakterinin kullanılmasıyla bir dizinin nasıl alt dizelere bölündüğü gösterilmektedir:

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

`split()` yönteminin isteğe bağlı olan ikinci parametresi, döndürülen dizinin maksimum boyutunu tanımlar.

Normal bir ifadeyi de sınırlayıcı olarak kullanabilirsiniz:

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

Daha fazla bilgi için bkz. “Normal ifadeler kullanma” sayfa 76 ve Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Dizelerdeki desenleri bulma ve alt dizeleri değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

String sınıfı, dizelerdeki desenlerle çalışılmasına yönelik şu yöntemleri içerir:

- Bir desenle eşleşen alt dizeleri bulmak için `match()` ve `search()` yöntemlerini kullanın.
- Bir desenle eşleşen alt dizeleri bulup bunları belirtilen bir alt dizeyle değiştirmek için `replace()` yöntemini kullanın.

Bunlar ilerleyen bölümlerde açıklanmaktadır.

Bu yöntemlerde kullanılan desenleri tanımlamak için dizeleri veya normal ifadeleri kullanabilirsiniz. Normal ifadeler hakkında daha fazla bilgi için bkz. “Normal ifadeler kullanma” sayfa 76.

Eşleşen alt dizeleri bulma

`search()` yöntemi, bu örnekte gösterildiği gibi, belirli bir desenle eşleşen birinci alt dizenin dizin konumunu döndürür:

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

Bu örnekte gösterildiği gibi, eşleşen deseni tanımlamak için normal ifadeleri kullanabilirsiniz:

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

Dizedeki birinci karakter 0 dizin konumunda olduğundan, `trace()` yönteminin çıktısı 0'dır. `i` bayrağı normal ifadede ayarlanır, bu nedenle arama büyük/küçük harf duyarlı değildir.

Normal ifadede `g` (genel) bayrağı ayarlanmış olsa da, `search()` yöntemi yalnızca bir eşleşme bulur ve bunun başlangıç dizin konumunu döndürür.

Aşağıdaki örnek, tırnak işareti içindeki bir dizeyle eşleşen, daha karmaşık bir normal ifadeyi gösterir:

```
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.)
```

`match()` yöntemi benzer şekilde çalışır. Eşleşen bir alt dizeyi arar. Ancak, aşağıdaki örnekte olduğu gibi, normal ifade deseninde genel bayrağını kullandığınızda, `match()` yöntemi, eşleşen alt dizelerin bir dizisini döndürür:

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

`results` dizisi şu şekilde ayarlanır:

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

Normal ifadeler hakkında daha fazla bilgi için, bkz. “Normal ifadeler kullanma” sayfa 76.

Eşleşen alt dizeleri değiştirme

Bir dizedeki belirtilen bir deseni aramak ve eşleşmeleri, belirtilen yerini alan dizeyle değiştirmek için, aşağıdaki örnekte gösterildiği gibi, `replace()` yöntemini kullanabilirsiniz:

```
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.
```

Normal ifadede `i` (`ignoreCase`) bayrağı ayarlanmış olduğundan, eşleşen dizelerin büyük/küçük harf duyarlı olmadığını ve `g` (`global`) bayrağı ayarlanmış olduğundan, birden çok eşleşmenin değiştirildiğini unutmayın. Daha fazla bilgi için bkz. “Normal ifadeler kullanma” sayfa 76.

Yerini alan dizeye şu `$` yerini alma kodlarınıdahil edebilirsiniz. Aşağıdaki tabloda gösterilen yerini alan metin, `$` yerini alma kodunun yerine eklenir:

$ Kodu	Yerini Alan Metin
`$$`	$
`$&`	Eşlenen alt dize.
`$` `	Eşleşen alt dizeden önce gelen dize bölümü. Bu kod, düz kesme işaretini (`'`) veya sola eğik kesme işaretini (`'`) değil, düz sola eğik kesme işareti karakterini (`` ` ``) kullanır.
`$'`	Eşleşen alt dizeden sonra gelen dize bölümü. Bu kod, düz kesme işaretini (`'`) kullanır.
$n	*n*. yakalanan parantez grubu eşleşmesi, burada n 1-9 arasında tek basamaklı bir sayıdır ve $n öğesinden sonra ondalık bir sayı gelmez.
$nn	*nn*. yakalanan parantez grubu eşleşmesi, burada *nn* 01–99 arasında iki basamaklı ondalık bir sayıdır. *nn*. yakalama tanımsızsa, yerini alan metin boş bir dizedir.

Örneğin, aşağıda, eşleşen birinci ve ikinci yakalama grubunu temsil eden`$2` ve `$1` yerini alma kodlarının kullanımını göstermektedir:

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

Bir işlevi `replace()` yönteminin ikinci parametresi olarak da kullanabilirsiniz. Eşleşen metin, işlevin döndürülen değeriyle değiştirilir.

```
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;
}
```

Bir işlevi, `replace()` yönteminin ikinci parametresi olarak kullandığınızda, şu argümanlar işleve iletilir:

- Dizenin eşleşen bölümü.
- Herhangi bir yakalama parantez grubu eşleşmesi. Bu şekilde iletilen argüman sayısı, parantez eşleşmelerinin sayısına bağlı olarak değişiklik gösterir. İşlev kodu içinde `arguments.length - 3` öğesini kontrol ederek, parantez eşleşmelerinin sayısını belirleyebilirsiniz.
- Eşlemenin başladığı dize içindeki dizin konumu.
- Tam dize.

Dizeleri büyük harfe veya küçük harfe dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örneklerde de gösterildiği gibi, `toLowerCase()` yöntemi ve `toUpperCase()` yöntemi, sırayla bir dizedeki alfabetik karakterleri küçük harfe ve büyük harfe dönüştürür:

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

Bu yöntemler çalıştırıldıktan sonra, kaynak dize değişmeden kalır. Kaynak dizeyi dönüştürmek için, şu kodu kullanın:

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

Bu yöntemler yalnızca a–z ve A–Z karakterleriyle değil, genişletilmiş karakterlerle çalışır:

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

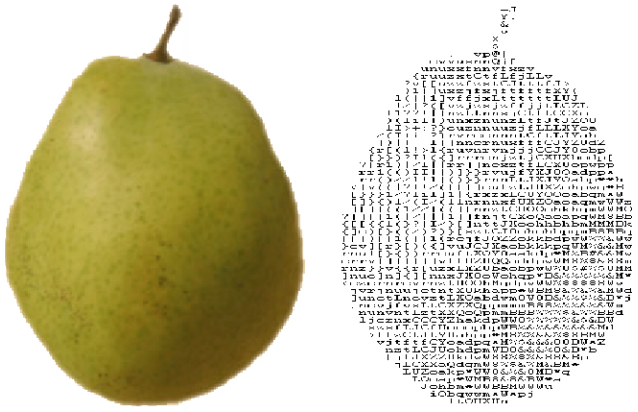
Dize örneği: ASCII sanatı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu ASCII Art örneği, ActionScript 3.0'da String sınıfıyla çalışılmasının aşağıda örnekleri verilen birçok özelliğini gösterir:

- String sınıfının `split()` yöntemi, karakter sınırlı bir dizeden (sekme sınırlı bir metin dosyasındaki görüntü bilgileri) değerleri ayıklamak için kullanılır.
- `split()`, bitiştirme ve `substring()` ile `substr()` yöntemlerini kullanarak dizenin bir bölümünü ayıklama gibi birçok dize işleme tekniği, görüntü başlıklarındaki her sözcüğün ilk harfini büyük harfli yapmak için kullanılır.
- `getCharAt()` yöntemi, bir dizeden tek bir karakter almak için kullanılır (gri tonlamalı bitmap verisine karşılık gelen ASCII karakterini belirlemek için).
- Dize bitiştirme, bir görüntünün bir defada tek bir karakterle ASCII art temsilini oluşturmak için kullanılır.

ASCII art terimi, bir görüntünün metin temsilini ifade eder, burada, Courier New karakterleri gibi tek boşluklu font karakterlerinin bir ızgarası resmi çizer. Aşağıdaki görüntü, uygulama tarafından oluşturulmuş bir ASCII art örneğini gösterir:



Grafiğin ASCII art sürümü sağda gösterilir.

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. ASCIIArt uygulama dosyalarını Samples/AsciiArt klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
AsciiArtApp.mxml veya AsciiArtApp.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	Metin dosyasından görüntü meta verisini ayıklama, görüntüleri yükleme ve görüntü-metin dönüştürme işlemini yönetme gibi, uygulamanın ana işlevlerini sağlayan sınıf.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	Görüntü verisinin bir String sürümüne dönüştürülmesi için `parseBitmapData()` yöntemini sağlayan bir sınıf.
com/example/programmingas3/asciiArt/Image.as	Yüklenen bir bitmap görüntüsünü temsil eden sınıf.
com/example/programmingas3/asciiArt/ImageInfo.as	ASCII art görüntüsünün meta verilerini (örn. başlık, görüntü dosyası URL'si, vb.) temsil eden sınıf.
image/	Uygulama tarafından kullanılan görüntüleri içeren klasör.
txt/ImageData.txt	Uygulama tarafından yüklenecek görüntülerle ilgili bilgileri içeren, sekme sınırlı metin dosyası.

Sekme sınırlı değerleri ayıklama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnek, yaygın bir uygulama olan, uygulama verilerinin uygulamadan ayrı saklanmasını kullanır; bu şekilde, veri değişirse (örneğin, başka bir görüntü eklenirse veya bir görüntünün başlığı değişirse), SWF dosyasının yeniden oluşturulması gerekmez. Bu durumda, görüntü başlığı, gerçek görüntü dosyasının URL'si ve görüntüyü işlemek için kullanılan bazı değerler gibi görüntü meta verileri, bir metin dosyasında (projedeki txt/ImageData.txt dosyasında) saklanır. Metin dosyasının içerikleri şunlardır:

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

Dosya belirli bir sekme sınırlı biçimi kullanır. Birinci satır, başlık satırıdır. Kalan satırlar, yüklenecek her bitmap için şu verileri içerir:

- Bitmap'in dosya adı.
- Bitmap'in görüntü adı.
- Bitmap'ler için beyaz eşiği ve siyah eşiği değerleri. Bunlar bir pikselin tamamen beyaz veya tamamen siyah olarak değerlendirilmesi için ölçüt olan onaltılık değerlerdir.

Uygulama başlatıldığı anda AsciiArtBuilder sınıfı, AsciiArtBuilder sınıfının `parseImageInfo()` yönteminden şu kodu kullanarak, görüntülenecek görüntülerin "yığınını" oluşturmak için metin dosyasının içeriklerini yükler ve ayrıştırır:

```
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);
    }
}
```

Metin dosyasının tüm içerikleri, tek bir String örneği olan `_imageInfoLoader.data` özelliğinde barındırılır. Parametre olarak yeni satır karakteriyle (`"\n"`) `split()` yöntemi kullanılarak, String örneği, öğelerinin her biri metin dosyası satırı olan bir Array (`lines`) öğesine bölünür. Daha sonra kod, satırların her biriyle çalışmak için bir döngü kullanır (gerçek içerik yerine yalnızca başlık içerdiğinden, birinci satır hariç). Döngünün içinde, tek bir satırın içeriklerini, değerler kümesine (`imageProperties` adındaki Array nesnesi) bölmek için bir kez daha `split()` yöntemi kullanılır. Bu durumda, her satırdaki karakterler sekme karakterleriyle ayrıldığından, `split()` yöntemiyle kullanılan parametre, sekme (`"\t"`) karakteridir.

Görüntü başlıklarını normalleştirmek için String yöntemlerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu uygulamanın tasarım kararlarından biri, görüntü başlıklarının, sözcüklerin her birinin baş harfi büyük olacak şekilde (yaygın olarak baş harfi küçük harfle yazılan birkaç sözcük dışında) standart biçim kullanılarak görüntülenmesidir. Uygulama, metin dosyasının uygun şekilde biçimlendirilmiş başlıklar içerdiğini varsaymak yerine, metin dosyasından ayıklanırken başlıkları biçimlendirir.

Önceki kod listesinde, tek bir görüntünün meta veri değerlerinin ayıklanmasının bir parçası olarak şu kod satırı kullanılır:

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

Bu kodda, görüntünün başlığı ImageInfo nesnesinde saklanmadan önce `normalizeTitle()` yöntemiyle metin dosyasından iletilir:

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

Bu yöntem, başlığı tek tek sözcüklere bölmek için (boşluk karakteriyle ayrılmış şekilde) `split()` yöntemini kullanır, `capitalizeFirstLetter()` yöntemi yoluyla sözcüklerin her birini iletir ve sonra sözcükleri tekrar tek bir dize halinde birleştirmek için Array sınıfının `join()` yöntemini kullanır.

`capitalizeFirstLetter()` yöntemi, her sözcüğün birinci harfini büyük harf yapma çalışmasını gerçekleştirir.

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

Türkçe'de, başlıkta şu sözcüklerden biri bulunuyorsa, bu sözcüklerin baş harfi büyük harfli *yapılmaz*: “ve”, “için”, “ancak”, “fakat”, “veya”, “de” “da” veya “ki” (Bu, kuralların basitleştirilmiş bir sürümüdür.) Bu mantığı çalıştırmak için kod ilk olarak bir `switch` deyimini kullanarak sözcüğün büyük harfli yapılmaması gereken sözcüklerden biri olup olmadığını kontrol eder. Büyük harfli yapılmaması gereken sözcüklerden biriyse, kod `switch` deyimini atlar. Diğer yandan, sözcüğün büyük harfli yapılması gerekiyorsa, bu da birkaç adım uygulanarak şu şekilde yapılır:

1. `substr(0, 1)` yöntemi kullanılarak sözcüğün birinci harfi ayıklanır, böylece 0 dizinindeki karakterle (`0` birinci parametresi tarafından belirtildiği gibi, dizedeki birinci harf) başlayan bir alt dize ayıklanır. Alt dize bir karakter uzunluğundadır (`1` ikinci parametresi tarafından belirtildiği gibi).
2. `toUpperCase()` yöntemi kullanılarak bu karakter büyük harfli yapılır.
3. Orijinal sözcüğün kalan karakterleri, `substring(1)` yöntemi kullanılarak ayıklanır, böylece dizin 1'de başlayan bir alt dize (ikinci harf), dizenin sonuna doğru ayıklanır (`substring()` yönteminin ikinci parametresinin bırakılmasıyla belirtildiği gibi).
4. Dize bitiştirmesi kullanılarak, yeni büyük harfli yapılan birinci harf, kalan harflerle birleştirilerek son sözcük oluşturulur: `firstLetter + otherLetters`.

ASCII art metni oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

BitmapToAsciiConverter sınıfı, bitmap görüntüsünün ASCII metin temsiline dönüştürülmesi işlevselliğini sağlar. Bu işlem, kısmen burada gösterilen `parseBitmapData()` yöntemi tarafından gerçekleştirilir:

```
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;
```

Bu kod ilk olarak, bitmap görüntüsünün ASCII art sürümünü oluşturmak için kullanılacak `result` adındaki bir String örneğini tanımlar. Daha sonra, kaynak bitmap görüntüsünün tek tek pikselleri üzerinde döngü gerçekleştirir. Birçok renk işleme tekniğini kullanarak (kısa tutmak amacıyla burada yer verilmemiştir), tek bir pikselin kırmızı, yeşil ve mavi renk değerlerini tek bir gri tonlamalı değere (0 ile 255 arasında bir değer) dönüştürür. Daha sonra kod bu değeri 0-63 ölçeğinde bir değere dönüştürmek için 4'e böler (gösterildiği gibi) ve sonrada bu değer `index` dizininde saklanır. (Bu uygulama tarafından kullanılan mevcut ASCII karakterlerinin "paleti" 64 değer içerdiğinden, 0-63 ölçeği kullanılır.) Karakterlerin paleti, BitmapToAsciiConverter sınıfında bir String örneği olarak tanımlanır:

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

`index` değişkeni, paletteki hangi ASCII karakterinin bitmap görüntüsündeki geçerli piksele karşılık geldiğini tanımladığından, `charAt()` yöntemi kullanılarak `palette` String öğesinden o karakter alınır. Daha sonra bitiştirme ataması (`+=`) operatörü kullanılarak bu karakter `result` String örneğine eklenir. Ayrıca, her piksel satırının sonunda, `result` String öğesinin sonuna yeni satır karakteri eklenerek, satır yeni bir "piksel" karakteri satırı oluşturmaya zorlanır.

Bölüm 3: Dizilerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Diziler tek bir veri yapısında birden fazla değer saklamanıza olanak sağlar. Değerleri sabit sıralı tam sayı dizinleri kullanarak saklayan basit dizinlenmiş dizileri veya değerleri rastgele anahtarlar kullanarak saklayan karmaşık ilişkili dizileri kullanabilirsiniz. Diziler ayrıca başka diziler içerebilir ve çok boyutlu olabilirler. Son olarak, öğeleri aynı veri türünün örneği olan bir dizi için bir Vector öğesi kullanabilirsiniz.

Daha fazla Yardım konusu

Dizi

Vektör

Dizilerin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Genellikle programlamada, tek bir nesne yerine bir öğeler kümesiyle çalışmanız gerekir. Örneğin, bir müzik çalar uygulamasında, oynatılmayı bekleyen bir şarkı listesinin olmasını isteyebilirsiniz. Bu listedeki her şarkı için ayrı bir değişken oluşturmak zorunda kalmak istemezsiniz. Tüm Song nesnelerinin bir paket içinde bir arada olması ve bunlarla bir grup olarak çalışılabilmesi tercih edilir.

Dizi, şarkı listesi gibi bir öğeler kümesi konteyneri olarak hareket eden bir programlama öğesidir. Genellikle dizideki öğelerin tümü aynı sınıfın örnekleridir ancak bu ActionScript'te bir zorunluluk değildir. Dizideki öğelerin her biri, dizinin *öğeleri* olarak bilinir. Diziyi, değişkenlere yönelik bir dosya çekmecesi olarak düşünebilirsiniz. Değişkenler, öğeler olarak diziye eklenebilir, bu tıpkı bir dosya çekmecesine klasör yerleştirilmesine benzer. Diziyle tek bir değişken olarak çalışabilirsiniz (çekmecenin tamamını farklı bir konuma taşımak gibi). Değişkenlerle bir grup olarak çalışabilirsiniz (bir bilgi ararken klasörlerde teker teker gezinmek gibi). Bunlara ayrı ayrı da erişebilirsiniz (çekmeceyi açıp tek bir klasörü seçmek gibi).

Örneğin, kullanıcının birden çok şarkıyı seçip oynatma listesine ekleyebildiği bir müzik çalar uygulaması oluşturduğunuzu varsayın. ActionScript kodunuzda, tek bir diziyi parametre olarak kabul eden, `addSongsToPlaylist()` adında bir yönteminiz vardır. Listeye kaç şarkı eklemek isterseniz isteyin (az, çok veya yalnızca bir adet), `addSongsToPlaylist()` yöntemini yalnızca bir defa çağırarak, Song nesnelerini içeren diziye iletirsiniz. `addSongsToPlaylist()` yönteminin içinde, dizi öğeleri (şarkılar) arasında birer birer ilerleyip bu öğeleri gerçekten oynatma listesine eklemek için bir döngü kullanabilirsiniz.

En yaygın ActionScript dizisi türü *dizinlenmiş dizidir*. Dizinlenmiş dizide her öğe numaralandırılmış bir yuvada (*dizin* olarak bilinir) saklanır. Bu öğelere, adres gibi numara kullanılarak erişilir. Dizinlenmiş diziler çoğu programlama ihtiyacı için işe yarar. Array sınıfı, dizinlenmiş diziyi temsil etmek için kullanılan yaygın bir sınıftır.

Dizinlenmiş dizi genellikle aynı türde birden çok öğeyi (aynı sınıfın örnekleri olan nesneler) saklamak için de kullanılır. Array sınıfı, içerdiği öğelerin türünü kısıtlamaya yönelik bir araç içermez. Vector sınıfı, tek bir dizideki tüm öğelerin aynı türde olduğu bir dizinlenmiş dizi türüdür. Array örneği yerine Vector örneğinin kullanılması, performans artışı ve başka avantajlar da sağlar. Vector sınıfı, Flash Player 10 ve Adobe AIR 1.5'ten itibaren kullanılabilir.

Dizinlenmiş dizinin özel bir kullanımı *çok boyutlu dizidir*. Çok boyutlu dizi, öğeleri de dizinlenmiş dizi olan (böylece başka öğeler içeren) bir dizinlenmiş dizidir.

Başka bir dizi türü de, tek tek öğeleri tanımlamak için sayısal bir dizin yerine bir *anahtar* dizesi kullanan *ilişkilendirilebilir dizidir*. Son olarak, ActionScript 3.0 aynı zamanda bir *sözlüğü* temsil eden Dictionary sınıfını da içerir. Sözlük, öğeleri ayırt etmek için herhangi türde bir nesneyi anahtar olarak kullanmanıza olanak sağlayan bir dizidir.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi dizi ve vektör işleme yordamlarını programlarken karşılaşacağınız önemli terimleri içerir.

Dizi Birden çok nesneyi gruplamak için kap görevi gören bir nesne.

Dizi erişimi ([]) operatörü Benzersiz şekilde dizi öğesini tanımlayan bir dizi veya anahtarı çevreleyen köşeli ayraç çifti. Bu sözdizimi, dizinin tamamını değil, dizinin tek bir öğesini belirtmek için bir dizi değişkeni adından sonra kullanılır.

İlişkilendirilebilir dizi Tek tek öğeleri tanımlamak için dize anahtarlarını kullanan bir dizi.

Taban türü Vector örneğinde depolanmasına izin verilen nesnelerin veri türü.

Dictionary Öğeleri arasında anahtar ve değer olarak bilinen nesne çiftleri yer alan bir dizi. Tek bir öğeyi tanımlamak için sayısal dizin yerine anahtar kullanılır.

Öğe Dizideki tek bir öğe.

Dizin Dizinlenmiş dizideki tek bir öğeyi tanımlamak için kullanılan sayısal "adres".

Dizinlenmiş dizi Öğelerin her birini numaralandırılmış konumda saklayan standart bir dizi türüdür ve tek tek öğeleri tanımlamak için numaraları (dizin) kullanır.

Anahtar İlişkilendirilebilir dizi veya sözlükteki tek bir öğeyi tanımlamak için kullanılan dize ya da nesne.

Çok boyutlu dizi Tek değerler yerine diziler niteliğindeki öğeleri içeren bir dizi.

T Taban türü ne olursa olsun, Vector örneğinin taban türünü temsil etmek için bu belgede kullanılan standart kural. T kuralı, Type parametresi açıklamasında gösterildiği gibi, bir sınıf adını temsil etmek için kullanılır. (“T”, “veri türü” içinde olduğu gibi “tür” sözcüğünü ifade eder).

Type parametresi Vector öğesinin taban türünü (depoladığı nesnelerin veri türünü) belirtmek için, Vector sınıfı adıyla kullanılan sözdizimi. Bu sözdizimi bir nokta (`.`) ve ardından açılı ayraç (`<>`) içine alınmış veri türü adını içerir. Tamamı şöyle görünür: `Vector.<T>`. Bu belgede, type parametresinde belirtilen sınıf genel olarak `T` ile belirtilir.

Vektör Öğelerinin tümü aynı veri türünde öğeler olan bir dizi türü.

Dizinlenmiş diziler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizinlenmiş diziler, verilerin her birine işaretsiz bir tam sayı değeri ile erişilebilecek şekilde bir veya birden fazla değer dizisi saklar. İlk dizin her zaman 0 numaradır ve dizin, diziye eklenen her alt öğe için 1 artar. ActionScript 3.0'da iki sınıf dizinlenmiş dizi olarak kullanılır: Array sınıfı ve Vector sınıfı.

Dizinlenmiş diziler, dizin sayısı için işaretsiz bir 32-bit tam sayı kullanır. Dizinlenmiş bir dizinin maksimum boyutu 2^{32} - 1 veya 4.294.967.295'tir. Maksimum boyuttan daha büyük bir dizi oluşturma girişimi sonunda çalışma zamanı hatası oluşur.

Dizinlenmiş dizinin tek bir öğesine erişmek için dizi erişimi (`[]`) operatörünü kullanarak, erişmek istediğiniz öğenin dizin konumunu belirtirsiniz. Örneğin, şu kod `songTitles` adındaki dizinlenmiş dizide bulunan ilk öğeyi (dizin 0'daki öğe) temsil eder:

```
songTitles[0]
```

Dizi değişkeni adı ile ardından gelen köşeli ayraç içindeki dizin birleşimi, tek bir tanımlayıcı olarak hareket eder. (Başka bir deyişle, bu bir değişken adı gibi kullanılabilir.) Atama deyiminin sol tarafındaki adı ve dizini kullanarak dizinlenmiş dizi öğesine bir değer atayabilirsiniz:

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

Aynı şekilde, atama deyiminin sağ tarafındaki adı ve dizini kullanarak dizinlenmiş dizi öğesinin değerini alabilirsiniz:

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

Ayrıca açıkça bir değer sağlamak yerine, köşeli ayraç içinde bir değişken de kullanabilirsiniz. (Değişken, uint, pozitif int veya pozitif tam sayı olan bir Number örneği gibi negatif olmayan bir tam sayı içermelidir.) Bu teknik genellikle dizinlenmiş dizideki öğeler üzerinde "döngü gerçekleştirmek" ve öğelerin bazılarında veya tümünde bir işlem gerçekleştirmek için yaygın olarak kullanılır. Aşağıdaki kod listesi bu tekniği göstermektedir. Bu kod, `oddNumbers` adındaki bir Array nesnesinde bulunan değerlerin her birine erişmek için bir döngü kullanır. Değerlerin her birini "oddNumber[*index*] = *value*" biçiminde yazdırmak için `trace()` deyimini kullanır:

```
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());
}
```

Array sınıfı

Birinci dizinlenmiş dizi türü, Array sınıfıdır. Bir Array örneği herhangi bir veri türünün değerini bulundurabilir. Aynı Array nesnesi, farklı veri türlerindeki nesneleri bulundurabilir. Örneğin, tek bir Array örneği, dizin 0'da bir String değerine, dizin 1'de bir Number örneğine ve dizin 2'de bir XML nesnesine sahip olabilir.

Vector sınıfı

ActionScript 3.0'da kullanılabilir olan başka bir dizinlenmiş dizi türü de Vector sınıfıdır. Vector örneği *türlenmiş dizidir*, başka bir deyişle, Vector örneğindeki tüm öğeler her zaman aynı veri türüne sahiptir.

Not: *Vector sınıfı, Flash Player 10 ve Adobe AIR 1.5'ten itibaren kullanılabilir.*

Bir Vector değişkeni bildirdiğinizde veya bir Vector nesnesini başlattığınızda, Vector öğesinin içerebildiği nesnelerin veri türünü açıkça belirtirsiniz. Belirtilen veri türü, Vector öğesinin *taban türü* olarak bilinir. Çalışma zamanında ve derleme zamanında (katı modda), Vector öğesinin değerini ayarlayan veya Vector öğesinden bir değer alan herhangi bir kod kontrol edilir. Eklenen veya alınan nesnenin veri türü, Vector öğesinin taban türüyle eşleşmezse bir hata oluşur.

Veri türü kısıtlamasına ek olarak, Vector sınıfının Array sınıfından ayırt edilmesini sağlayan başka kısıtlamaları vardır:

- Vector yoğun bir dizidir. Array nesnesi, 1-6 arasındaki konumlarda değer içermese de, 0 ve 7 dizinlerinde değerler içerebilir. Ancak bir Vector öğesinin her dizinde bir değer (veya `null` değeri) içermesi gerekir.
- Vector isteğe bağlı olarak sabit uzunlukta olabilir. Başka bir deyişle, Vector öğesinin içerdiği öğelerin sayısı değişemez.

- Vector öğelerine erişim sınırlarla denetlenir. Son öğeden büyük bir dizinden (`length` - 1) asla bir değer okuyamazsınız. Asla geçerli son dizinin ötesinde birden çok dizin içeren bir değer ayarlayamazsınız. (Başka bir deyişle, varolan bir dizinde veya `[length]` dizininde yalnızca bir değer ayarlayabilirsiniz.)

Vector öğesinin, sahip olduğu kısıtlamalar nedeniyle, tüm öğeleri tek bir sınıfın örnekleri olan Array örneğine kıyasla üç tane başlıca avantajı vardır:

- Performans: Array örneği yerine Vector örneği kullanıldığında, dizi öğesinin erişimi ve yinelemesi daha hızlıdır.
- Tür güvenliği: katı modda derleyici veri türü hatalarını tanımlayabilir. Bu tür hata örnekleri arasında, Vector öğesine yanlış veri türünde bir değer atanması veya Vector öğesinden değer okunurken yanlış veri türü beklenmesi yer alır. Çalışma zamanında, Vector nesnesine veri eklenirken veya Vector öğesinden veri okunurken de veri türleri kontrol edilir. Ancak bir Vector öğesine değer eklemek için `push()` yöntemini veya `unshift()` yöntemini kullandığınızda, argümanların veri türlerinin derleme zamanında kontrol edilmediğini unutmayın. Bu yöntemler kullanılırken, değerler çalışma zamanında kontrol edilir.
- Güvenilirlik: çalışma zamanı aralık kontrolü (veya sabit uzunluk kontrolü) Dizilere kıyasla güvenilirliği önemli derecede artırır.

Ek kısıtlama ve avantajların yanı sıra, Vector sınıfı Array sınıfına çok benzer. Bir Vector nesnesinin özellikleri ve yöntemleri birçok durumda Array öğesinin özelliklerine ve yöntemlerine benzer. Tüm öğelerin aynı veri türüne sahip olduğu bir Dizi kullandığınız çoğu durumunda, Vector örneği tercih edilebilir.

Dizi oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Array örneği veya Vector örneği oluşturmak için birçok teknik kullanabilirsiniz. Ancak, her dizi oluşturma tekniği birbirinden farklılık gösterir.

Array örneği oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`Array()` yapıcısını çağırarak veya Array değişmez sözdizimini kullanarak bir Array nesnesi oluşturursunuz.

`Array()` yapıcısı işlevi üç farklı şekilde kullanılabilir. İlk olarak, yapıcıyı argüman olmadan çağırırsanız, boş bir dizi alırsınız. Dizide herhangi bir öğe olmadığını doğrulamak için Array sınıfının `length` özelliğini kullanabilirsiniz. Örneğin, şu kod `Array()` yapıcısını argüman olmadan çağırır:

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

İkinci olarak, `Array()` yapıcısına tek parametre olarak bir sayıyı kullanırsanız, o sayı değerinin belirttiği uzunlukta bir dizi oluşturulur ve öğelerin her birinin değeri `undefined` olarak ayarlanır. Argümanın, 0 ile 4.294.967.295 arasında işaretsiz bir tam sayı olması gerekir. Örneğin, şu kod `Array()` yapıcısını tek bir sayısal argüman ile çağırır:

```
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
```

Üçüncü olarak, yapıcıyı çağırır ve parametre olarak bir öğe listesi iletirseniz, parametrelerin her birine karşılık gelen öğelerle bir dizi oluşturulur. Şu kod `Array()` yapıcısına üç argüman iletir:

```
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
```

Array değişmezleriyle de diziler oluşturabilirsiniz. Aşağıdaki örnekte gösterildiği gibi, bir Array değişmezi doğrudan bir dizi değişkenine atanabilir:

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

Vector örneği oluşturma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`Vector.<T>()` yapıcısını çağırarak bir Vector örneği oluşturduğunuzda ayrıca taban türünü de belirtirsiniz. `Vector.<T>()` global işlevini çağırarak da bir Vector oluşturabilirsiniz. Bu işlev, belirtilen bir nesneyi Vector örneğine dönüştürür. Flash Professional CS5 ve üzerinde, Flash Builder 4 ve üzerinde, Flex 4 ve üzerinde Vektör değişmez sözdizimi kullanarak vektör örneği oluşturabilirsiniz.

Her Vector değişkeni (veya aynı şekilde bir Vector yöntemi parametresi ya da yöntem döndürme türü) bildirdiğinizde, Vector değişkeninin taban türünü belirtirsiniz. `Vector.<T>()` yapıcısını çağırarak bir Vector örneği oluşturduğunuzda ayrıca taban türünü de belirtirsiniz. Başka bir deyişle, ActionScript'te `Vector` terimini her kullandığınızda, bir taban türü bu terime eşlik eder.

type parametresi sözdizimini kullanarak Vector öğesinin taban türünü belirtirsiniz. type parametresi, kodda `Vector` sözcüğünden hemen sonra gelir. Bu örnekte gösterildiği gibi, bir nokta (`.`) ve ardından açılı ayraç (`<>`) içine alınmış taban sınıfı adını içerir:

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

Örneğin birinci satırında, `v` değişkeni `Vector.<String>` örneği olarak bildirilir. Başka bir deyişle, bu yalnızca String örneklerini barındırabilen bir dizinlenmiş diziyi temsil eder. İkinci satır, aynı Vector türünde bir örnek (başka bir deyişle, öğelerinin tümü String örneği olan bir Vector) oluşturmak için `Vector()` yapıcısını çağırır. Bu nesneyi `v` öğesine atar.

Vector.<T>() yapıcısını kullanma

`Vector.<T>()` yapıcısını argüman olmadan kullanırsanız, bu boş bir Vector örneği oluşturur. `length` özelliğini kontrol ederek bir Vector öğesinin boş olup olmadığını test edebilirsiniz. Örneğin, şu kod, `Vector.<T>()` yapıcısını argüman olmadan çağırır:

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

Vector öğesinin başlangıçta kaç tane öğeye ihtiyacı olduğunu önceden bilirseniz, Vector öğesinde öğe sayısını önceden tanımlayabilirsiniz. Belirli sayıda öğe içeren bir Vector öğesi oluşturmak için, öğe sayısını birinci parametre (`length` parametresi) olarak iletin. Vector öğeleri boş olamayacağından, öğeler taban türünde örneklerle doldurulur. Taban türü, `null` değerlerine izin veren bir başvuru türüyse, öğelerin tümü `null` değerini içerir. Aksi takdirde, öğelerin tümü sınıfın varsayılan değerini içerir. Örneğin, bir uint değişkeni `null` olamaz. Sonuç olarak, aşağıdaki kod listesinde, her biri 0 değerini içeren yedi öğeyle `ages` adındaki Vector öğesi oluşturulur:

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

Son olarak, `Vector.<T>()` yapıcısını kullanarak, ikinci parametre (`fixed` parametresi) için `true` değerini iletip sabit uzunlukta bir Vector öğesi de oluşturabilirsiniz. Bu durumda, belirtilen öğe sayısına sahip bir Vector öğesi oluşturulur ve öğelerin sayısı değiştirilemez. Ancak, yine de sabit uzunluktaki Vector öğelerinin değerlerini değiştirebildiğinizi unutmayın.

Vektör değişmez sözdizimi oluşturucu

Flash Professional CS5 ve üzerinde, Flash Builder 4 ve üzerinde, Flex 4 ve üzerinde `Vector<T>()` yapıcısına Vector öğesinin ilk değerlerini belirtmek için değerler listesi iletebilirsiniz:

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

Aşağıdaki bilgiler bu sözdizimi için geçerlidir:

- Son virgül isteğe bağlıdır.
- Dizideki boş öğeler desteklenmez; `var v:Vector.<int> = new <int>[0,,2,]` gibi bir ifade bir derleyici hatası atar.
- Vector örneği için varsayılan bir uzunluk belirtemezsiniz. Uzunluk, başlatma listesindeki öğe sayısı kadardır.
- Vector örneğinin sabit bir uzunluğa sahip olup olmadığını belirtemezsiniz. Onun yerine, `fixed` özelliğini kullanın.
- Değerler olarak iletilen öğeler belirtilen türle eşleşmezse veri kaybı veya hatalar oluşabilir. Örneğin:

  ```
  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
  ```

Vector.<T>() global işlevini kullanma

Vector nesnesi oluşturmak için `Vector.<T>(` ve Vector değişmez sözdizim yapıcılarının yanı sıra `Vector.<T>()` global işlevini de kullanabilirsiniz. `Vector.<T>()` global işlevi bir dönüştürme işlevidir. `Vector.<T>()` global işlevini çağırdığınızda, yöntemin döndürdüğü Vector öğesinin taban türünü belirtirsiniz. Argüman olarak tek bir dizinlenmiş dizi (Array veya Vector örneği) iletirsiniz. Bu yöntem daha sonra belirtilen taban türüne sahip, kaynak dizi argümanındaki değerleri içeren bir Vector öğesi döndürür. Aşağıdaki kod listesi, `Vector.<T>()` global işlevini çağırmak için sözdizimini gösterir:

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

`Vector.<T>()` global işlevi, iki düzeyde veri türü dönüştürmesi gerçekleştirir. İlk olarak, bir Array örneği işleve iletildiğinde, Vector örneği döndürülür. İkinci olarak, kaynak dizi bir Array veya Vector örneği de olsa, işlev kaynak dizinin öğelerini taban türünün değerlerine dönüştürmeye çalışır. Dönüştürme, standart ActionScript veri türü dönüştürme kurallarını kullanır. Örneğin, aşağıdaki kod listesi, kaynak Array öğesindeki String değerlerini sonuç Vector öğesindeki tam sayılara dönüştürür. Birinci değerin ondalık kısmı (`"1.5"`) kırpılır ve sayısal olmayan üçüncü değer (`"Waffles"`) sonuçta 0'a dönüştürülür:

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

Kaynak öğelerden herhangi biri dönüştürülemezse, bir hata oluşur.

Kod, `Vector.<T>()` global işlevini çağırdığında, kaynak dizideki bir öğe belirtilen taban türünün alt sınıfının bir örneğiyse, öğe sonuçta elde edilen Vector öğesine eklenir (herhangi bir hata oluşmaz). `Vector.<T>()` global işlevinin çağrılması, `T` taban türüne sahip bir Vector öğesinin, `T` öğesinin üst sınıfı olan bir taban türüne sahip Vector öğesine dönüştürülmesinin tek yoludur.

Dizi öğeleri ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizinlenmiş diziye bir öğe eklemenin en temel yolu, dizi erişimi (`[]`) operatörünün kullanılmasıdır. Dizinlenmiş dizi öğesinin değerini ayarlamak için, Array veya Vector nesne adını ve atama deyiminin sol tarafındaki dizin sayısını kullanın:

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

Array veya Vector öğesi, o dizinde bir öğe içermiyorsa, dizin oluşturulur ve değer orada saklanır. O dizinde değer varsa, yeni değer varolan değerin yerini alır.

Array nesnesi, herhangi bir dizinde öğe oluşturmanıza olanak sağlar. Ancak bir Vector nesnesiyle yalnızca varolan bir dizine veya sonraki mevcut dizine değer atayabilirsiniz. Sonraki mevcut dizin, Vector nesnesinin `length` özelliğine karşılık gelir. Vector nesnesine yeni bir öğe eklemenin en güvenli yolu, bu listeye benzer bir kod kullanılmasıdır:

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

Array ve Vector sınıfı yöntemlerinin üçü (`push()`, `unshift()` ve `splice()`) bir dizinlenmiş diziye öğeler eklemenize olanak sağlar. `push()` yöntemi bir dizinin sonuna bir veya daha fazla öğe ekler. Başka bir deyişle, `push()` yöntemi kullanılarak bir diziye eklenen son öğe, en yüksek dizin numarasına sahip olacaktır. `unshift()` yöntemi bir dizinin başına, her zaman 0 dizin numaralı olan bir veya birden fazla öğe ekler. `splice()` yöntemi dizideki belirli bir dizine istenen sayıda öğe ekler.

Aşağıdaki örnek, bu yöntemlerin üçünü de göstermektedir. Gezegenlerin güneşe yakınlık sıralarına göre gezegenlerin adlarını saklamak için `planets` adlı bir dizi oluşturulur. Önce, ilk öğe olan `Mars` öğesini eklemek için `push()` yöntemi çağrılır. Ardından dizinin önüne ait olan `Mercury` öğesini eklemek için `unshift()` yöntemi çağrılır. Son olarak `Venus` ve `Earth` öğelerini, `Mercury` öğesinin arkasına `Mars` öğesinin önüne eklemek için `splice()` yöntemi çağrılır. `splice()` öğesine gönderilen ilk argüman olan 1 tam sayısı, eklemenin dizin 1'den başlaması komutunu verir. `splice()` öğesine gönderilen ikinci argüman olan 0 tam sayısı, herhangi bir öğenin silinmeyeceğini belirtir. Son olarak, `splice()` öğesine gönderilen üçüncü ve dördüncü argümanlar olan `Venus` ve `Earth` eklenir.

```
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
```

`push()` ve `unshift()` yöntemleri, değiştirilen dizinin uzunluğunu temsil eden işaretsiz bir tam sayı döndürür. `splice()` yöntemi öğe eklemek için kullanıldığında boş bir dizi döndürür; bu her ne kadar garip gelse de `splice()` yönteminin çok yönlülüğü göz önüne alındığında daha kolay anlaşılabilir. `splice()` yöntemini yalnızca bir diziye öğe eklemek için değil ayrıca bir diziden öğe kaldırmak için de kullanabilirsiniz. `splice()` yöntemi öğeleri kaldırmak için kullanıldığında, kaldırılan öğeleri içeren bir dizi döndürür.

Not: *Vector nesnesinin `fixed` özelliği `true` olursa, Vector öğesindeki toplam öğe sayısı değiştirilemez. Burada açıklanan teknikleri kullanarak sabit uzunluktaki Vector öğesine yeni bir öğe eklemeyi denerseniz, bir hata oluşur.*

Değerleri alma ve dizi öğelerini kaldırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizinlenmiş diziden bir öğenin değerini almanın en basit yolu, dizi erişimi (`[]`) operatörünün kullanılmasıdır. Dizinlenmiş dizi öğesinin değerini almak için, Array veya Vector nesne adını ve atama deyiminin sağ tarafındaki dizin sayısını kullanın:

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

Herhangi bir öğenin bulunmadığı bir dizin kullanılarak Array veya Vector öğesinden değer almaya çalışılması mümkündür. Bu durumda, Array nesnesi undefined değerini döndürür ve Vector öğesi bir RangeError istisnası atar.

Array ve Vector sınıfının üç yöntemi (`pop()`, `shift()` ve `splice()`) öğeleri kaldırmanıza olanak sağlar. `pop()` yöntemi, dizinin sonundaki öğeyi kaldırır. Başka bir deyişle, en yüksek dizin numarasına sahip öğeyi kaldırır. `shift()` yöntemi, dizinin başındaki öğeyi, başka bir deyişle, her zaman 0 dizin numarasına sahip olan öğeyi kaldırır. Öğe eklemek için de kullanılabilen `splice()` yöntemi, yönteme gönderilen ilk argüman tarafından belirtilen dizin numarasından başlayarak rastgele sayıda öğeyi kaldırır.

Aşağıdaki örnek, bir Array örneğinden öğe kaldırmaya yönelik üç yöntemi de kullanır. Geniş su kütlelerinin adlarını saklamak için `oceans` adında bir Array öğesi oluşturulur. Array öğesindeki adların bazıları okyanus olmayıp göl olduğundan, bu adların kaldırılması gerekir.

İlk olarak `Aral` ve `Superior` öğelerini kaldırmak ve `Atlantic` ve `Indian` öğelerini eklemek için `splice()` yöntemi kullanılır. `splice()` öğesine gönderilen ilk argüman olan 2 tam sayısı, işlemin dizin 2'deki listede bulunan üçüncü öğeden başlaması gerektiğini belirtir. İkinci argüman olan 2, iki öğenin kaldırılması gerektiğini belirtir. Kalan argümanlar olan `Atlantic` ve `Indian`, dizin 2 konumuna eklenecek değerlerdir.

İkinci olarak, dizideki son öğe olan `Huron` öğesini kaldırmak için `pop()` yöntemi kullanılır. Üçüncü olarak da, dizideki ilk öğe olan `Victoria` öğesini kaldırmak için `shift()` yöntemi kullanılır.

```
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
```

`pop()` ve `shift()` yöntemleri, kaldırılan öğeyi döndürür. Bir Array örneği için, diziler herhangi bir veri türünün değerlerini bulundurabildiğinden, döndürülen değerin veri türü Object olur. Vector örneği için, döndürülen değerin veri türü, Vector öğesinin taban türüdür. `splice()` yöntemi, kaldırılan değerleri içeren bir Array veya Vector öğesini döndürür. Aşağıdaki örnekte gösterildiği gibi, `splice()` öğesine yapılan bir çağrı, döndürülen Array öğesini yeni bir Array değişkenine atayacak şekilde `oceans` Array örneğini değiştirebilirsiniz:

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

Bir Array nesnesi öğesinde `delete` operatörünü kullanan bir kodla karşılaşabilirsiniz. `delete` operatörü, bir Array öğesinin değerini `undefined` olarak ayarlar ancak öğeyi Array öğesinden kaldırmaz. Örneğin, şu kod `oceans` Array içindeki üçüncü öğede `delete` operatörünü kullanır, ancak Array öğesinin uzunluğu 5 olarak kalır:

```
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
```

Bir dizinin `length` özelliğini kullanarak Array veya Vector öğesini kırpabilirsiniz. Dizinlenmiş dizinin `length` özelliğini geçerli dizi uzunluğundan düşük bir uzunluğa ayarlarsanız, yeni `length` eksi 1 değerinden yüksek dizin sayılarında saklanan öğeler kaldırılarak dizi kırpılır. Örneğin, `oceans` dizisi, tüm geçerli girişler dizinin başında olacak şekilde sıralansaydı, şu kodda gösterildiği gibi, dizinin sonundaki girişleri kaldırmak için `length` özelliğini kullanabilirdiniz:

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

Not: *Vector nesnesinin* `fixed` *özelliği* `true` *olursa, Vector öğesindeki toplam öğe sayısı değiştirilemez. Burada açıklanan teknikleri kullanarak sabit uzunluktaki Vector öğesinden bir öğeyi kaldırmayı veya söz konusu Vector öğesini kırpmayı denerseniz, bir hata oluşur.*

Diziyi sıralama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sıralamayı düzenleyerek veya ters çevirerek dizinlenmiş dizinin sıralamasını değiştirmenize olanak sağlayan üç yöntem vardır: `reverse()`, `sort()` ve `sortOn()` Bu yöntemlerin tümü, varolan diziyi değiştirir. Aşağıdaki tabloda, bu yöntemler ve bu yöntemlerin Array ve Vector nesnelerine yönelik davranışı özetlenmektedir:

Yöntem	Array davranışı	Vector davranışı
`reverse()`	Son öğe ilk öğe olacak ve sondan bir önceki öğe ikinci öğe olacak, vb. şekilde dizinin sırasını değiştirir	Array davranışıyla aynıdır
`sort()`	Array içindeki öğeleri, alfabetik veya sayısal sıralama gibi önceden tanımlı çeşitli şekillerde sıralamanıza olanak sağlar. Özel bir sıralama algoritması da belirtebilirsiniz.	Öğeleri, belirttiğiniz özel sıralama algoritmasına göre sıralar
`sortOn()`	Sıralama anahtarı olarak kullanmak üzere özellik veya özellikleri belirterek bir veya birkaç ortak özelliğe sahip nesneleri sıralamanıza olanak sağlar	Vector sınıfında kullanılamaz

reverse() yöntemi

`reverse()` yöntemi herhangi bir parametre almaz ve bir değer döndürmez, ancak dizinizin sırasını geçerli durumundan tersi sıralamaya geçirmenizi sağlar. Şu örnek, `oceans` dizisinde listelenen okyanusların sırasını tersine çevirir:

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

sort() yöntemi ile temel sıralama (yalnızca Array sınıfı)

Array örneği için, `sort()` yöntemi *varsayılan sıralama düzenini* kullanarak dizideki öğeleri yeniden düzenler. Varsayılan sıralama düzeni şu özelliklere sahiptir:

- Sıralama büyük/küçük harf duyarlıdır, başka bir deyişle, büyük harfli karakterler küçük harfli karakterlerden önce gelir. Örneğin, D harfi b harfinden önce gelir.
- Sıralama artan sıradadır, başka bir deyişle, düşük karakter kodları (örn. A), daha yüksek karakter kodlarından (örn. B) önce gelir.
- Sıralama, aynı değerleri belirli bir sıralama olmaksızın birbiriyle bitiştirir.
- Sıralama dize tabanlıdır, başka bir deyişle, öğeler karşılaştırılmadan önce dizelere dönüştürülür. (Örneğin, `"1"` dizesi, `"3"` dizesinden daha düşük karakter koduna sahip olduğundan 10, 3'ten önce gelir.)

Büyük/küçük harf duyarlılığı olmadan veya azalan sırada dizinizi sıralamanız gerekebilir ya da dizinizde alfabetik değil de sayısal olarak sıralamak istediğiniz sayılar bulunabilir. Array sınıfının `sort()` yöntemi, varsayılan sıralama düzeninin her bir özelliğini değiştirmenize olanak sağlayan bir `options` parametresine sahiptir. Bu seçenekler, aşağıdaki listede gösterildiği gibi, Array sınıfındaki statik sabitler kümesiyle tanımlanır:

- `Array.CASEINSENSITIVE`: Bu seçenek, sıralamanın büyük/küçük harf duyarlı olmasını önler. Örneğin, küçük b harfi, büyük D harfinden önce gelir.

- `Array.DESCENDING`: Bu seçenek varsayılan artan sıralamayı ters çevirir. Örneğin, B harfi A harfinden önce gelir.
- `Array.UNIQUESORT`: Bu seçenek iki aynı değer bulunduğunda sıralamanın durdurulmasına neden olur.
- `Array.NUMERIC`: Bu seçenek sayısal sıralama sağlar, böylece 3, 10 sayısından önce gelir.

Aşağıdaki örnek, bu seçeneklerden bazılarını vurgular. Birçok farklı seçenek kullanılarak sıralanan `poets` adında bir Array öğesi oluşturulur.

```
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
```

sort() yöntemiyle özel sıralama (Array ve Vector sınıfları)

Array nesnesi için kullanılabilir olan temel sıralamaya ek olarak, özel bir sıralama kuralı da tanımlayabilirsiniz. Bu teknik, Vector sınıfı için kullanılabilir olan tek `sort()` yöntemi biçimidir. Özel bir sıralama tanımlamak için, özel bir sıralama işlevi yazıp bunu argüman olarak `sort()` yöntemine iletirsiniz.

Örneğin, her liste öğesinde kişinin tam adının bulunduğu bir ad listeniz varsa ve listeyi soyadına göre sıralamak istiyorsanız, her bir öğeyi ayrıştırmak için özel bir sıralama işlevi kullanmanız ve sıralama işlevinde soyadını kullanmanız gerekir. Aşağıdaki kod `Array.sort()` yöntemine yönelik bir parametre olarak kullanılan özel bir işlevle bunun nasıl yapılabileceğini gösterir:

```
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
```

`orderLastName()` özel sıralama işlevi, karşılaştırma işlemi için kullanılmak üzere her bir öğeden soyadını ayıklamak için normal bir ifade kullanır. `orderLastName` işlev tanımlayıcısı, `names` dizisinde `sort()` yöntemi çağrılırken tek parametre olarak kullanılır. Sıralama işlevi aynı ada iki dizi öğesinde çalıştığından, `a` ve `b` olmak üzere iki parametreyi kabul eder. Sıralama işlevinin döndürme değeri, öğelerin nasıl sıralanması gerektiğini belirtir:

- -1 döndürme değeri, birinci parametrenin (`a`) ikinci parametreden (`b`) önce geldiğini belirtir.
- 1 döndürme değeri, ikinci parametrenin (`b`) birinci parametreden (`a`) önce geldiğini belirtir.
- 0 döndürme değeri, öğelerin eşit sıralama önceliğine sahip olduğunu belirtir.

sortOn() yöntemi (yalnızca Array sınıfı)

`sortOn()` yöntemi, nesneleri içeren öğelerin bulunduğu Array nesneleri için tasarlanmıştır. Bu nesnelerde, sıralama anahtarı olarak kullanılabilecek en az bir ortak özelliğin bulunması beklenir. Başka herhangi bir türdeki dizi için `sortOn()` yönteminin kullanılması beklenmeyen sonuçlara yol açar.

Not: *Vector sınıfı `sortOn()` yöntemini içermez. Bu yöntem yalnızca Array nesneleri için kullanılabilir.*

Aşağıdaki örnek, her bir öğe dize değil de nesne olacak şekilde `poets` Array öğesini değiştirir. Her nesne, şairin soyadını ve doğum yılını içerir.

```
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"});
```

Array öğesini `born` özelliğine göre sıralamak için `sortOn()` yöntemini kullanabilirsiniz. `sortOn()` yöntemi şu iki parametreyi tanımlar: `fieldName` ve `options`. `fieldName` argümanının dize olarak belirtilmesi gerekir. Aşağıdaki örnekte, `sortOn()` öğesi `"born"` ve `Array.NUMERIC` olmak üzere iki argümanla çağrılır. Sıralamanın alfabetik değil sayısal olarak yapılmasını sağlamak için `Array.NUMERIC` argümanı kullanılır. Diziye daha az veya daha çok basamaklı bir sayı da eklense sıralamanın beklendiği gibi davranacağı kesin olduğundan, tüm sayılar aynı sayıda basamak içerse de bu iyi bir uygulamadır.

```
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
*/
```

Orijinal diziyi değiştirmeden sıralama (yalnızca Array sınıfı)

Genellikle, `sort()` ve `sortOn()` yöntemleri bir Array öğesini değiştirir. Varolan diziyi değiştirmeden Array öğesini sıralamak isterseniz, `options` parametresinin parçası olarak `Array.RETURNINDEXEDARRAY` sabitini iletin. Bu seçenek, sıralamayı yansıtan yeni bir Array öğesi döndürmesini ve orijinal Array öğesini olduğu gibi bırakmasını yöntemlere bildirir. Yöntemler tarafından döndürülen Array öğesi, yeni sıralama düzenini yansıtan basit bir dizin sayıları Array öğesi olup orijinal Array içindeki öğeleri içermez. Örneğin, Array öğesini değiştirmeden `poets` Array öğesini doğum yılına göre sıralamak için `options` parametresi için iletilen argümanın parçası olarak `Array.RETURNINDEXEDARRAY` sabitini dahil edin.

Aşağıdaki örnek, döndürülen dizin bilgilerini `indices` adında bir Array öğesinde saklar ve şairleri doğum yılına göre sıralanmış şekilde vermek için `indices` dizisini değiştirilmemiş `poets` dizisiyle birlikte kullanır:

```
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
*/
```

Diziyi sorgulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Array ve Vector sınıfının dört yönteminin tümü (`concat()`, `join()`, `slice()` ve `toString()`) bilgi için diziyi sorgular ancak diziyi değiştirmez. `concat()` ve `slice()` yöntemleri yeni diziler döndürürken, `join()` ve `toString()` yöntemleri de dize döndürür. `concat()` yöntemi, argüman olarak yeni bir diziyi veya öğeler listesini alır ve yeni bir dizi oluşturmak için bunları varolan diziyle birleştirir. `slice()` yöntemi, `startIndex` ve `endIndex` adında iki parametre içerir ve varolan diziden "dilimli" öğelerin bir kopyasını içeren yeni bir dizi döndürür. Dilim, `startIndex` parametresindeki öğeyle başlar ve `endIndex` parametresinden hemen önceki öğeyle sona erer. Şunu yinelemekte fayda var: `endIndex` parametresindeki öğe döndürme değerine dahil edilmez.

Aşağıdaki örnek, diğer dizilerin öğelerini kullanarak yeni diziler oluşturmak için `concat()` ve `slice()` öğelerini kullanır:

```
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
```

Diziyi sorgulayıp dizinin içeriklerini dize olarak döndürmek için `join()` ve `toString()` yöntemlerini kullanabilirsiniz. `join()` yöntemi için herhangi bir parametre kullanılmıyorsa, iki yöntem de aynı şekilde hareket ederek dizideki tüm öğelerin virgül sınırlı listesini içeren bir dize döndürür. `join()` yöntemi, `toString()` yönteminden farklı olarak, döndürülen dizedeki her öğe arasında ayırıcı olarak kullanılacak sembolü seçmenizi sağlayan `delimiter` adında bir parametreyi kabul eder.

Aşağıdaki örnek, `rivers` adında bir Array öğesi oluşturur ve Array öğesindeki değerleri dize olarak döndürmek için hem `join()` hem de `toString()` öğelerini çağırır. `toString()` yöntemi virgül sınırlı değerler (`riverCSV`) döndürmek için kullanılırken, `join()` yöntemi `+` karakteriyle ayrılmış değerler döndürmek için kullanılır.

```
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
```

`join()` yöntemiyle ilgili bilinmesi gereken bir nokta, aşağıdaki örnekte gösterildiği gibi, ana dizi öğeleri için hangi ayırıcıyı belirttiğinize bakılmaksızın, yuvalanmış Array veya Vector örneklerinin her zaman virgülle ayrılmış değerlerle döndürülmesidir:

```
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
```

İlişkilendirilebilir diziler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazen *karma* veya *eşleme* olarak da adlandırılan ilişkilendirilebilir bir dizi, saklanan değerleri organize etmek için sayısal bir dizin yerine *anahtarları* kullanır. İlişkilendirilebilir bir dizideki her anahtar, saklanan değere erişmek için kullanılan benzersiz bir dizedir. İlişkilendirilebilir bir dizi, Object sınıfının bir örneği olup bu, her anahtarın bir özellik adına karşılık geldiği anlamına gelir. İlişkilendirilebilir diziler, anahtar ve değer çiftlerinin sıralanmamış koleksiyonlarıdır. Kodunuzun belirli bir sırada olması için ilişkilendirilebilir dizinin anahtarlarını beklememesi gerekir.

ActionScript 3.0, *sözlük* adında gelişmiş bir ilişkilendirilebilir dizi türünü de içerir. flash.utils paketinde Dictionary sınıfının örnekleri olan sözlükler, herhangi bir veri türündeki anahtarları kullanır. Başka bir deyişle, sözlük anahtarları, String türünün değerleriyle sınırlı değildir.

Dize anahtarları ile ilişkilendirilebilir diziler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da ilişkilendirilebilir dizi oluşturmanın iki yolu vardır. Birinci yol, Object örneği kullanılmasıdır. Object örneği kullanarak, bir nesne değişmeziyle dizinizi başlatabilirsiniz. Object sınıfının *genel nesne* olarak da adlandırılan bir örneği, ilişkilendirilebilir bir diziyle aynı işleve sahiptir. Genel nesnenin özellik adlarının her biri, saklanan değere erişilmesini sağlayan bir anahtar görevi görür.

Aşağıdaki örnek iki anahtar ve değer çiftiyle diziyi başlatmak için bir nesne değişmezi kullanan `monitorInfo` adında ilişkilendirilebilir bir dizi oluşturur:

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

Bildirim zamanında diziyi başlatmanız gerekmiyorsa, diziyi oluşturmak için Object yapıcısını şu şekilde kullanabilirsiniz:

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

Bir nesne değişmezi veya Object sınıfı yapıcısı kullanılarak dizi oluşturulduktan sonra, dizi erişimi (`[]`) operatörünü veya nokta operatörünü (`.`) kullanarak diziye yeni değerler ekleyebilirsiniz. Aşağıdaki örnek `monitorArray` öğesine iki yeni değer ekler:

```
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
```

`aspect ratio` adındaki anahtarın bir boşluk karakteri içerdiğini unutmayın. Bu, dizi erişimi (`[]`) operatörüyle mümkün olsa da, nokta operatörüyle denendiğinde hataya yol açar. Anahtar adlarınızda boşluk kullanılması önerilmez.

İlişkilendirilebilir bir dizi oluşturmanın ikinci yolu, Array yapıcısı (veya herhangi bir dinamik sınıf yapıcısı) kullanıldıktan sonra diziye anahtar ve değer çiftleri eklemek için dizi erişimi (`[]`) operatörünün veya nokta operatörünün (`.`) kullanılmasıdır. İlişkilendirilebilir dizinizin Array türünde olmasını bildirirseniz, diziyi başlatmak için bir nesne değişmezi kullanamazsınız. Aşağıdaki örnek, Array yapıcısını kullanarak `monitorInfo` adında ilişkilendirilebilir bir dizi oluşturur ve `type` adında bir anahtar ile `resolution` adında bir anahtarı değerleriyle birlikte ekler:

```
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
```

İlişkilendirilebilir bir dizi oluşturmak için Array yapıcısının kullanılması herhangi bir avantaj sağlamaz. Array yapıcısını veya Array veri türünü kullanıyor olsanız da, ilişkilendirilebilir dizilerle `Array.length` özelliğini veya Array sınıfı yöntemlerinden herhangi birini kullanamazsınız. Dizinlenmiş diziler oluşturmanın en iyi yolu Array yapıcısının kullanılmasıdır.

Nesne anahtarları ile ilişkilendirilebilir diziler (Sözlükler)

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Anahtarlar için dize yerine nesne kullanan ilişkilendirilebilir bir dizi oluşturmak üzere Dictionary sınıfını kullanabilirsiniz. Bu diziler bazen sözlük, karma veya eşleme olarak adlandırılır. Örneğin, belirli bir konteynerle ilişkisini esas alarak bir Sprite nesnesinin konumunu belirleyen bir uygulamayı göz önünde bulundurun. Sprite nesnelerinin her birini bir konteynere eşlemek için Dictionary nesnesini kullanabilirsiniz.

Aşağıdaki kod, Dictionary nesnesi için anahtar görevi gören üç Sprite sınıfı örneği oluşturur. Her anahtara `GroupA` veya `GroupB` değeri atanır. Değerler herhangi bir veri türünde olabilir ancak bu örnekte hem `GroupA` hem de `GroupB` Object sınıfının örnekleridir. Daha sonra, aşağıdaki kodda gösterildiği gibi, dizi erişimi (`[]`) operatörü ile anahtarların her biriyle ilişkilendirilmiş değere erişebilirsiniz:

```
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");
}
```

Nesne anahtarlarıyla yineleme

`for..in` döngüsü veya `for each..in` döngüsü ile Dictionary nesnesinin içeriklerini yineleyebilirsiniz. `for..in` döngüsü, anahtarları esas alarak yineleme yapmanızı sağlarken, `for each..in` döngüsü, her bir anahtarla ilişkilendirilmiş değerleri esas alarak yineleme yapmanıza olanak sağlar.

Dictionary nesnesinin nesne anahtarlarına doğrudan erişmek için `for..in` döngüsünü kullanın. Dizi erişimi (`[]`) operatörüyle Dictionary nesnesinin değerlerine de erişebilirsiniz. Aşağıdaki kod önceki `groupMap` sözlüğü örneğini kullanarak, `for..in` döngüsü ile Dictionary nesnesinin nasıl yinelendiğini gösterir:

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

Dictionary nesnesinin değerlerine doğrudan erişmek için `for each..in` döngüsünü kullanın. Aşağıdaki kod, ayrıca `groupMap` sözlüğünü kullanarak, `for each..in` döngüsü ile Dictionary nesnesinin nasıl yinelendiğini gösterir:

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

Nesne anahtarları ve bellek yönetimi

Adobe® Flash® Player ve Adobe® AIR™, artık kullanılmayan belleği kurtarmak için bir çöp toplama sistemi kullanır. Bir nesne kendisini işaret eden bir başvuru içermiyorsa, nesne çöp toplama için uygun hale gelmiş demektir ve çöp toplama sistemi tekrar çalıştırıldığında bellek kurtarılır. Örneğin, aşağıdaki kod yeni bir nesne oluşturur ve o nesnenin başvurusunu `myObject` değişkenine atar:

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

Nesneyi işaret eden bir başvuru olduğu sürece, çöp toplama sistemi nesnenin kapladığı belleği kurtarmaz. `myObject` değeri, farklı bir nesneyi işaret edecek şekilde değiştirilirse veya `null` değerine ayarlanırsa, yalnızca orijinal nesneyi işaret eden başka bir başvuru olmaması durumunda orijinal nesnenin kapladığı bellek çöp toplama işlemi için uygun hale gelir.

`myObject` öğesini Dictionary nesnesinde bir anahtar olarak kullanırsanız, orijinal nesneyi işaret eden başka bir başvuru oluşturuyor olursunuz. Örneğin, şu kod, `myObject` değişkeni ve `myMap` nesnesindeki anahtar olmak üzere bir nesneye iki başvuru oluşturur:

```
import flash.utils.Dictionary;

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

`myObject` tarafından başvurulan nesneyi çöp toplama işlemi için uygun hale getirmek istiyorsanız, bu nesneyi işaret eden tüm başvuruları kaldırmanız gerekir. Bu durumda, aşağıdaki kodda gösterildiği gibi, `myObject` öğesinin değerini değiştirmeniz ve `myMap` öğesinden `myObject` anahtarını silmeniz gerekir:

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

Alternatif olarak, tüm sözlük anahtarlarını *zayıf başvurular* haline getirmek için, Dictionary öğesinin `useWeakReference` parametresini kullanabilirsiniz. Çöp toplama sistemi, zayıf başvuruları yok sayar, başka bir deyişle, yalnızca zayıf başvurular içeren bir nesne de çöp toplama işlemi için uygundur. Örneğin, aşağıdaki kodda nesneyi çöp toplama işlemi için uygun hale getirmek üzere `myMap` öğesinden `myObject` anahtarını silmeniz gerekmez:

```
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.
```

Çok boyutlu diziler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çok boyutlu diziler öğe olarak başka diziler içerir Örneğin, dizinlenmiş dizeler dizisi olarak saklanan bir görev listesini göz önünde bulundurun:

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

Haftanın her günü için ayrı bir görevler listesi saklamak istiyorsanız, haftanın her günü için tek bir öğe içeren çok boyutlu bir dizi oluşturabilirsiniz. Her öğe, görevler listesinin saklandığı `tasks` dizisine benzer şekilde dizinlenmiş bir dizi içerir. Çok boyutlu dizilerde, dizinlenmiş veya ilişkilendirilebilir dizilerin herhangi bir birleşimini kullanabilirsiniz. Aşağıdaki bölümlerde verilen örnekler, iki dizinlenmiş dizi veya dizinlenmiş dizilerin ilişkilendirilebilir bir dizisini kullanır. Pratik amacıyla diğer birleşimleri de denemek isteyebilirsiniz.

İki dizinlenmiş dizi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İki dizinlenmiş dizi kullandığınızda, sonucu bir tablo veya elektronik tablo olarak görselleştirebilirsiniz. Birinci dizinin öğeleri, tablonun satırlarını temsil ederken, ikinci dizinin öğeleri de sütunları temsil eder.

Örneğin, aşağıdaki çok boyutlu dizi, haftanın her bir günü için görev listelerini izlemek üzere iki dizinlenmiş dizi kullanır. Array sınıfı yapıcısı kullanılarak birinci dizi (`masterTaskList`) oluşturulur. 0 değeri Pazartesi'yi ve 6 değeri Pazar'ı temsil edecek şekilde, dizinin her bir öğesi haftanın bir gününü temsil eder. Bu öğeler, tablodaki satırlar olarak değerlendirilebilir. `masterTaskList` dizisinde oluşturduğunuz yedi öğenin her birine bir dizi değişmezi atayarak her günün görevini oluşturabilirsiniz. Dizi değişmezleri, tablodaki sütunları temsil eder.

```
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"];
```

Dizi erişimi (`[]`) operatörünü kullanarak görev listelerinden herhangi birinde bulunan tek tek öğelere erişebilirsiniz. Birinci ayraç kümesi haftanın gününü temsil ederken, ikinci ayraç kümesi de o günün görev listesini temsil eder. Örneğin, ikinci görevi Çarşamba'nın listesinden almak üzere Çarşamba için dizin 2'yi ve listedeki ikinci görev için dizin 1'i kullanın.

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

Birinci görevi Pazar'ın listesinden almak üzere Pazar için dizin 6'yı ve listedeki birinci görev için dizin 0'ı kullanın.

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

Dizinlenmiş bir dizi ile ilişkilendirilebilir dizi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tek tek dizileri daha kolay erişilebilir duruma getirmek üzere haftanın günleri için ilişkilendirilebilir bir dizi ve görev listeleri için dizinlenmiş bir dizi kullanabilirsiniz. İlişkilendirilebilir bir dizi kullanılması, haftanın belirli bir günü ifade edilirken nokta sözdizimini kullanmanıza olanak sağlar ancak bu durumda ilişkilendirilebilir dizinin her bir öğesine erişmek için fazladan çalışma zamanı işleme maliyeti oluşur. Aşağıdaki örnek, haftanın her bir günü için bir anahtar ve değer çiftiyle, bir görev listesinin temeli olarak ilişkilendirilebilir bir dizi kullanır:

```
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"];
```

Nokta sözdizimi, birden çok ayraç kümesinin önlenmesini sağlayarak kodu daha okunaklı hale getirir.

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

`for..in` döngüsünü kullanarak görev listesini yineleyebilirsiniz, ancak her bir anahtarla ilişkilendirilmiş değere erişmek için nokta sözdizimi yerine dizi erişimi (`[]`) operatörünü kullanmanız gerekir. `masterTaskList` öğesi ilişkilendirilebilir bir dizi olduğundan, aşağıdaki örnekte gösterildiği gibi, öğeler mutlaka beklediğiniz sırada alınmayabilir:

```
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
*/
```

Dizileri klonlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Array sınıfı, dizilerin kopyasının oluşturulmasına yönelik yerleşik bir yöntem içermez. Herhangi bir argüman olmadan `concat()` veya `slice()` yöntemlerinden birini çağırarak bir dizinin *basitkopyasını* oluşturabilirsiniz. Basit kopyada, orijinal dizi, nesne olan öğeler içeriyorsa, nesnelerin kendisi değil yalnızca nesnelerin başvuruları kopyalanır. Kopya, orijinallerle aynı nesneleri işaret eder. Nesneler üzerinde yapılan tüm değişiklikler her iki dizide de yansıtılır.

Kapsamlı kopyada, orijinal dizide bulunan tüm nesneler de kopyalanır ve böylece yeni dizi, orijinal diziyle aynı nesneleri işaret etmez. Kapsamlı kopyalamada, genellikle bir işlevin oluşturulması için çağrı yapan birden çok kod satırı gerekir. Bu tür bir işlev, genel bir yardımcı program işlevi olarak veya Array alt sınıfının bir yöntemi olarak oluşturulabilir.

Aşağıdaki örnek, kapsamlı kopyalama yapan `clone()` adında bir işlevi tanımlar. Algoritma, ortak bir Java programlama tekniğinden alınmıştır. Bu işlev, diziyi bir ByteArray sınıfı örneğine serileştirip sonra diziyi geri yeni bir diziye okuyarak kapsamlı bir kopya oluşturur. Bu işlev, aşağıdaki kodda gösterildiği gibi, hem dizinlenmiş dizilerle hem ilişkilendirilebilir dizilerle kullanılabilecek şekilde bir nesneyi kabul eder:

```
import flash.utils.ByteArray;

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

Array sınıfını genişletme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Array sınıfı, son olmayan birkaç çekirdek sınıftan biridir, başka bir deyişle, kendi Array alt sınıflarınızı oluşturabilirsiniz. Bu bölüm, Array alt sınıfının nasıl oluşturulacağını gösteren bir örnek sağlar ve işlem sırasında ortaya çıkabilecek bazı sorunları ele alır.

Önceden de belirtildiği gibi, ActionScript'teki diziler türlenmiş değildir ancak yalnızca belirli bir veri türündeki öğeleri kabul eden bir Array alt sınıfı oluşturabilirsiniz. Aşağıdaki bölümlerde verilen örnek, öğelerini birinci parametrede belirtilen veri türünün değerleriyle sınırlandıran TypedArray adında bir Array alt sınıfını tanımlar. TypedArray sınıfı yalnızca Array sınıfının nasıl genişletildiğini gösteren bir örnek olarak sunulmuş olup birçok nedenden dolayı üretim amaçları için uygun olmayabilir. İlk olarak, tür denetleme, derleme zamanında değil çalışma zamanında gerçekleşir. İkinci olarak, yöntemler istisna atacak şekilde kolayca değiştirilebilse de, TypedArray yöntemi bir uyuşmazlıkla karşılaştığında uyuşmazlık yoksayılır ve herhangi bir istisna atılmaz. Üçüncü olarak, sınıf, herhangi bir türdeki değerleri diziye eklemek için dizi erişimi operatörünün kullanılmasını önleyemez. Dördüncü olarak, kodlama stili, basitliği performans en iyileştirmesinden daha üstün tutar.

Not: *Türlenmiş dizi oluşturmak için burada açıklanan tekniği kullanabilirsiniz. Ancak, Vector nesnesinin kullanılması daha iyi bir yaklaşım olacaktır. Vector örneği gerçek bir türlenmiş dizi olup Array sınıfına veya herhangi bir alt sınıfa göre daha yüksek performans ve iyileştirmeler sağlar. Bu açıklamanın amacı, Array alt sınıfının nasıl oluşturulduğunu göstermektir.*

Alt sınıfı bildirme

Sınıfın bir Array alt sınıfı olduğunu belirtmek için `extends` anahtar sözcüğünü kullanın. Array alt sınıfının, Array sınıfı gibi `dynamic` niteliğini kullanması gerekir. Aksi takdirde, alt sınıfınız düzgün çalışmaz.

Aşağıdaki kod, veri türünü barındıran bir sabiti, yapıcı yöntemini ve diziye öğe ekleyebilen dört yöntemi içeren TypedArray sınıfının tanımını gösterir. Her yöntemin kodu bu örnekte çıkarılmıştır ancak ilerleyen bölümlerde ele alınıp kapsamlı şekilde açıklanacaktır:

```
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 {}
}
```

Bu örnek, `-as3` derleyici seçeneğinin `true` değerine ve `-es` derleyici seçeneğinin `false` değerine ayarlandığını varsaydığından, geçersiz kılınan dört yöntem de `public` niteliği yerine AS3 ad alanını kullanır. Adobe Flash Builder ve Adobe® Flash® Professional için varsayılan ayarlar bunlardır.

Prototip miras kullanmayı tercih eden ileri düzey bir geliştiriciyseniz, `-es` derleyici seçeneği `true` değerine ayarlanmış şekilde TypedArray sınıfını derlemek için bu sınıf üzerinde iki küçük değişiklik yapabilirsiniz. İlk olarak, tüm `override` niteliklerini kaldırın ve AS3 ad alanını `public` niteliğiyle değiştirin. İkinci olarak, `Array.prototype` öğesini dört `super` öğesinin yerine yedek olarak kullanın.

TypedArray yapıcısı

Yapıcının rastgele uzunluktaki argümanlar listesini kabul etmesi gerektiğinden, alt sınıf yapıcısı ilginç bir zorluk ortaya çıkarır. Dizi oluşturmak için argümanların üst yapıcıya iletilme şekli zorluk yaratır. Argümanlar listesini dizi olarak iletirseniz, üst yapıcı bunu Array türünde tek bir argüman olarak değerlendirir ve sonuçta elde edilen sizi her zaman 1 öğe uzunluktadır. Doğrudan geçiş argümanı listeleri, geleneksel olarak `Function.apply()` yöntemi kullanılarak işlenir. Bu yöntem, argümanlar dizisini ikinci parametre olarak alır ancak işlevi çalıştırırken bunu bir argümanlar dizisine dönüştürür. Ne yazık ki, `Function.apply()` yöntemi yapıcılarla kullanılamaz.

Kalan tek seçenek, TypedArray yapıcısında Array yapıcısının mantığını yeniden oluşturmaktır. Aşağıdaki kod, Array sınıfı yapıcısında kullanılan algoritmayı gösterir; bu algoritmayı Array alt sınıfı yapıcınızda yeniden kullanabilirsiniz:

```
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]
            }
        }
    }
}
```

TypedArray yapıcısı, kod üzerinde yalnızca dört değişiklikle Array yapıcısındaki kodun büyük bir kısmını paylaşır. İlk olarak parametre listesi, dizinin veri türünün belirtimine olanak sağlayan yeni bir gerekli Class türü parametresini içerir. İkinci olarak, yapıcıya iletilen veri türü, `dataType` değişkenine atanır. Üçüncü olarak, `else` deyiminde, `for` döngüsünden sonra `length` özelliğinin değeri atanır, böylece `length` yalnızca uygun türdeki argümanları içerir. Dördüncü olarak, `for` döngüsünün gövdesi `push()` yönteminin geçersiz kılınmış sürümünü kullanır, böylece yalnızca doğru veri türündeki argümanlar diziye eklenir. Aşağıdaki örnek, TypedArray yapıcı işlevini gösterir:

```
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;
        }
    }
}
```

TypedArray geçersiz kılınmış yöntemler

TypedArray sınıfı, bir diziye öğe ekleyebilen Array sınıfının dört yöntemini geçersiz kılar. Her durumda, geçersiz kılınan yöntem, doğru veri türünde olmayan öğelerin eklenmesini önleyen bir tür denetimi ekler. Sonra da her yöntem kendisinin üst sınıf sürümünü çağırır.

`push()` yöntemi bir `for..in` döngüsüyle argümanlar listesini yineler ve her argümanda tür denetimi gerçekleştirir. Doğru veri türünde olmayan tüm argümanlar, `splice()` yöntemiyle `args` dizisinden kaldırılır. `for..in` döngüsü sona erdikten sonra, `args` dizisi yalnızca `dataType` türündeki değerleri içerir. Daha sonra, aşağıdaki kodun gösterdiği gibi, güncellenmiş `args` dizisiyle `push()` öğesinin üst sınıf sürümü çağrılır:

```
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));
}
```

`concat()` yöntemi, tür denetiminden geçen argümanları saklamak için `passArgs` adında geçici bir TypedArray oluşturur. Bu, `push()` yönteminde bulunan tür denetimi kodunun yeniden kullanılmasına olanak sağlar. `for..in` döngüsü, `args` dizisini yineler ve her argümanda `push()` öğesini çağırır. `passArgs` öğesi TypedArray olarak yazılmış olduğundan, `push()` öğesinin TypedArray sürümü çalıştırılır. Daha sonra `concat()` yöntemi, aşağıdaki kodun gösterdiği gibi, kendi üst sınıf sürümünü çağırır:

```
    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));
    }
```

`splice()` yöntemi, argümanların rastgele bir listesini alır ancak ilk iki argüman her zaman bir dizin sayısını ve silinecek öğe sayısını ifade eder. Geçersiz kılınmış `splice()` yönteminin yalnızca dizin konumu 2 veya daha yukarısındaki `args` dizi öğeleri için tür denetimi yapmasının nedeni de budur. Koddaki ilgi çekici bir nokta da, `for` döngüsü içinde `splice()` öğesine yinelemeli çağrı yapıldığı halde, `args` öğesi TypedArray değil de Array türünde olduğundan, başka bir deyişle, `args.splice()` öğesine yapılan çağrı, yöntemin üst sınıf sürümüne yapılan bir çağrı olduğundan, bunun yinelemeli bir çağrı olmamasıdır. `for..in` döngüsü tamamlandıktan sonra, aşağıdaki kodda gösterildiği gibi, `args` dizisi yalnızca 2 veya daha yüksek dizin konumlarındaki doğru türdeki değerleri içerir ve `splice()` öğesi kendi üst sınıf sürümünü çağırır:

```
    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));
    }
```

Dizinin başına öğe ekleyen `unshift()` yöntemi de argümanların rastgele bir listesini kabul eder. Geçersiz kılınmış `unshift()` yöntemi, aşağıdaki örnek kodda gösterildiği gibi, `push()` yönteminin kullandığına benzer bir algoritma kullanır:

```
    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));
    }
}
```

Diziler örneği: PlayList

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

PlayList örneği, şarkı listesini yöneten bir müzik çalma listesi uygulaması bağlamında dizilerle çalışma tekniklerini gösterir. Bu teknikler şunlardır:

- Dizinlenmiş bir dizi oluşturma
- Dizinlenmiş bir diziye öğeler ekleme
- Farklı sıralama seçeneklerini kullanarak nesne dizisini farklı özelliklerine göre sıralama
- Bir diziyi karakter sınırlı bir dizeye dönüştürme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. PlayList uygulama dosyası, Samples/PlayList klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
PlayList.mxml veya PlayList.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/playlist/PlayList.as	Bir şarkı listesini temsil eden sınıf. Listeyi saklamak için bir Array kullanır ve listenin öğelerinin sıralamasını yönetir.
com/example/programmingas3/playlist/Song.as	Tek bir şarkı hakkındaki bilgileri temsil eden bir değer nesnesi. PlayList sınıfı tarafından yönetilen öğeler Song örnekleridir.
com/example/programmingas3/playlist/SortProperty.as	Kullanılabilir değerleri, Song nesnelerinin listesinin sıralanmasında ölçüt olarak kullanılabilen Song sınıfının özelliklerini temsil eden sahte numaralandırma.

PlayList sınıfına genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

PlayList sınıfı, Song nesnelerinin kümesini yönetir. Oynatma listesine şarkı ekleme (`addSong()` yöntemi) ve listedeki şarkıları sıralama (`sortList()` yöntemi) işlevleriyle genel yöntemler içerir. Ayrıca sınıf, oynatma listesindeki gerçek şarkı kümesine erişilmesini sağlayan salt okunur bir erişimci özelliği (`songList`) içerir. Dahili olarak, PlayList sınıfı özel bir Array değişkenini kullanarak şarkılarını izler:

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

PlayList sınıfı tarafından şarkı listesini izlemek üzere kullanılan `_songs` Array değişkenine ek olarak, başka iki özel değişken de listenin sıralanması gerekip gerekmediğini (`_needToSort`) ve şarkı listesinin belirli bir zamanda hangi özelliğe göre sıralandığını (`_currentSort`) izler.

Tüm nesnelerde olduğu gibi, Array örneğinin bildirilmesi, Array oluşturulması işleminin yalnızca yarısıdır. Bir Array örneğinin özelliklerine veya yöntemlerine erişilmeden önce, PlayList sınıfının yapıcısında Array örneğinin başlatılması gerekir.

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

Yapıcının ilk satırı, kullanıma hazır olması için `_songs` değişkenini başlatır. Ayrıca, ilk sıralama özelliğini ayarlamak için `sortList()` yöntemi çağrılır.

Listeye şarkı ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı uygulamaya yeni bir şarkı girdiğinde, veri girişi formundaki kod, PlayList sınıfının `addSong()` yöntemini çağırır.

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

`addSong()` içinde, `_songs` dizisinin `push()` yöntemi çağrılır ve böylece `addSong()` öğesine iletilen Song nesnesi o diziye yeni bir öğe olarak eklenir. `push()` yöntemiyle, önceden uygulanmış olabilecek herhangi bir sıralama dikkate alınmaksızın, yeni öğe dizinin sonuna eklenir. Başka bir deyişle, `push()` yöntemi çağrıldıktan sonra, şarkı listesi artık doğru şekilde sıralanmaz, bu nedenle `_needToSort` değişkeni `true` değerine ayarlanır. Teoride, `sortList()` yöntemi hemen çağrılabilir, böylece listenin belirli bir zamanda sıralanıp sıralanmadığını izleme gereği ortadan kalkar. Pratikteyse, alınmadan hemen önce şarkı listesinin sıralanması gerekmez. Sıralama işlemi ertelendiğinde, uygulama gereksiz olan sıralama işlemini (örneğin, alınmadan önce listeye birçok şarkı eklendiğinde) gerçekleştirmez.

Şarkı listesini sıralama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Oynatma listesi tarafından yönetilen Song örnekleri karmaşık nesneler olduğundan, uygulama kullanıcıları, şarkı başlığı veya yayınlama yılı gibi farklı özelliklere göre oynatma listesini sıralamak isteyebilir. PlayList uygulamasında, şarkı listesini sıralama görevi üç bölüm içerir: listenin sıralanma ölçütü olan özelliği tanımlama, bu özelliğe göre sıralama yapılırken hangi sıralama seçeneklerinin kullanılması gerektiğini belirtme ve gerçek sıralama işlemini gerçekleştirme.

Sıralama özellikleri

Song nesnesi, şarkı başlığı, sanatçı, yayınlama yılı, dosya adı ve kullanıcı tarafından seçilen, şarkının ait olduğu bir tür kümesi gibi birçok özelliği izler. Bunlar arasından yalnızca ilk üçü sıralama için pratik ölçütlerdir. Geliştiriciler için kolaylık sağlaması açısından örnek, sıralama için kullanılabilir özellikleri temsil eden değerlerle numaralandırma olarak hareket eden SortProperty sınıfını içerir.

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

SortProperty sınıfı üç sabit içerir: `TITLE`, `ARTIST` ve `YEAR`; bunların her biri sıralama için kullanılabilen ilişkilendirilmiş Song sınıfı özelliğinin gerçek adını içeren bir String öğesini sağlar. Kodun geri kalanında, her sıralama özelliği belirtildiğinde, bu işlem numaralandırma üyesi kullanılarak yapılır. Örneğin, PlayList yapıcısında, aşağıdaki gibi başlangıçta `sortList()` yöntemi çağrılarak liste sıralanır:

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

Sıralama özelliği `SortProperty.TITLE` olarak belirtildiğinden, şarkılar başlığa göre sıralanır.

Özelliğe göre sıralama ve sıralama seçeneklerini belirtme

Şarkı listesini gerçekten sıralama çalışması, aşağıdaki gibi, `sortList()` yöntemindeki PlayList sınıfı tarafından gerçekleştirilir:

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

Başlığa veya sanatçıya göre sıralama yapılırken sıralamanın alfabetik olması akıllıcayken, yıla göre sıralama yapılırken sayısal sıralama yapılması akıllıcadır. `switch` deyimi, `sortProperty` parametresinde belirtilen değere göre `sortOptions` değişkeninde saklanan uygun sıralama seçeneğini tanımlamak için kullanılır. Burada da, sabit değerler yerine özellikleri birbirinden ayırmak için adlandırılmış numaralandırma üyeleri kullanılır.

Sıralama özelliği ve sıralama seçenekleri belirlenmiş durumdayken, `sortOn()` yöntemi çağrılıp bu iki değerin parametre olarak iletilmesiyle `_songs` dizisi gerçekten sıralanır. Şarkı listesi geçerli olarak sıralandığı gibi geçerli sıralama özelliği kaydedilir.

Dizi öğelerini karakter sınırlı bir dizeye birleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dizilerin, şarkı listesini PlayList sınıfında tutmak için kullanılmasına ek olarak, bu örnekte, belirli bir şarkının ait olduğu tür listesinin yönetilmesine yardımcı olmak için de Song sınıfında diziler kullanılır. Song sınıfının tanımındaki bu kod parçasını göz önünde bulundurun:

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

Yeni bir Song örneği oluşturulurken, şarkının ait olduğu türü (veya türleri) belirtmek için kullanılan `genres` parametresi, bir Array örneği olarak tanımlanır. Böylece, birden çok türün yapıcıya iletilebilen tek bir değişken şeklinde gruplanması yararlı hale gelir. Ancak, dahili olarak Song sınıfı, özel `_genres` değişkenindeki türleri noktalı virgülle ayrılmış bir String örneği olarak tutar. Belirtilen sınırlayıcı olarak `";"` değişmez dize değeriyle `join()` yöntemi çağrılarak, Array parametresi noktalı virgülle ayrılmış bir dizeye dönüştürülür.

Aynı işaretle, `genres` erişimcileri, türlerin Array olarak ayarlanmasına veya alınmasına olanak sağlar:

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

`genresset` erişimcileri yapıcıyla tamamen aynı şekilde hareket eder; bir Array öğesini kabul eder ve bunu noktalı virgülle ayrılmış bir String öğesine dönüştürmek için `join()` yöntemini çağırır. `get` erişimcisi tam tersi işlemi gerçekleştirir: `_genres` değişkeninin `split()` yöntemi çağrılır, böylece String öğesi, belirtilen sınırlayıcı (daha önceki gibi `";"` değişmez dize değeri) kullanılarak değerler dizesine ayrılır.

Bölüm 4: Hataları işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Hataları "işleme", bir hataya yanıt vermek ya da hatayı düzeltmek için uygulamanıza mantık eklemek demektir. Hatalar, bir uygulama derlediğinde veya derlenmiş bir uygulama yürütüldüğünde oluşturulur. Uygulamanız hataları işlerken, yanıt olmamasının tersine (hatayı oluşturan işlev, sessizce başarısız olduğu zaman) bir hatayla karşılaşıldığında bir yanıt olarak *birşey* gerçekleşir. Hata işleme doğru şekilde kullanıldığında, uygulamanızı ve kullanıcıları beklenmeyen davranışlara karşı korumanıza yardımcı olur.

Ancak hata işleme, derleme sırasında veya bir uygulama çalışırken atılan birçok türde hatayı yanıtlayan geniş bir kategoridir. Bu bölümde, çalışma zamanı hatalarının nasıl işlendiği, oluşturulabilecek farklı hata türleri ve ActionScript 3.0'daki hata işleme sisteminin avantajları ele alınmıştır.

Hata işlemenin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çalışma zamanı hatası, ActionScript içeriğinin istenen şekilde çalışmasını engelleyen ActionScript kodunda ters giden bir şeydir. ActionScript kodunuzun kullanıcılar için sorunsuz bir biçimde çalışmasını sağlamak için hatayı işleyen; yani hatayı gideren, geçici bir çözüm oluşturan veya en azından kullanıcının hata oluştuğunu bilmesini sağlayan bir kod yazın. Bu işleme *hata işleme* adı verilir.

Hata işleme, derleme sırasında veya bir uygulama çalışırken atılan birçok türde hatayı yanıtlayan geniş bir kategoridir. Derleme zamanında gerçekleşen hataların tanımlanması genellikle daha kolaydır. SWF dosyası oluşturma işlemini tamamlamak için bu hataları giderin.

Çalışma zamanı hatalarının gerçekleşmesi için kodun gerçekten çalışıyor olması gerektiğinden, çalışma zamanı hatalarının algılanması daha zor olabilir. Programınızın bir parçasında birçok kod dalı (bir `if..then..else` ifadesi varsa, kodunuzun hatasız olduğunu onaylamak için, gerçek kullanıcıların kullanabileceği her olası değeri, tüm olası girdi değerleriyle birlikte test edin.

Çalışma zamanı hataları iki kategoriye ayrılır: *program hataları*, ActionScript kodunuzdaki hatalardır, örn. bir yöntem parametresi için yanlış veri türünün belirtilmesi; *mantıksal hatalar*, programınızın mantığındaki (veri denetleme ve değer işleme) hatalardır, örn. bankacılık uygulamasında faiz oranlarını hesaplamak için yanlış formülün kullanılması. Bu iki hata türü de, genellikle uygulamanızın test edilmesiyle zaman içerisinde algılanıp düzeltilebilir.

İdeal olarak uygulamanızdaki tüm hataları son kullanıcılara yayınlanmadan tanımlayıp kaldırmak istersiniz. Ancak tüm hatalar önceden görülüp önlenemez. Örneğin, ActionScript uygulamanızın denetiminiz dışındaki belirli bir web sitesinden bilgi yüklediğini varsayın. Bir noktada bu web sitesi kullanılabilir olmazsa, uygulamanızın bu harici veriye dayanan kısmı doğru şekilde davranmaz. Hata işlemenin en önemli boyutu, bilinmeyen durumlar için hazırlanma ve sorunsuz bir biçimde işlenmeleridir. Kullanıcıların, uygulamanızı kullanmaya devam etmek veya en azından uygulamanın neden çalışmadığını açıklayan samimi bir hata mesajı almaya ihtiyacı vardır.

Çalışma zamanı hataları ActionScript'te iki şekilde temsil edilir:

- Hata sınıfları: Çoğu hatanın kendisiyle ilişkilendirilmiş bir hata sınıfı vardır. Bir hata oluştuğunda, Flash çalışma zamanı (Flash Player veya Adobe AIR gibi), söz konusu hatayla ilişkilendirilmiş belirli hata örneğini oluşturur. Kodunuz, hataya uygun bir yanıt oluşturmak için bu hata nesnesinde bulunan bilgileri kullanabilir.

- Hata olayları: Bazen, Flash çalışma zamanı normalde bir olay tetikleyecekken bir hata oluşur. Böyle durumlarda olay yerine bir hata olayı tetiklenir. Her hata olayının kendisiyle ilişkilendirilmiş bir sınıfı vardır ve Flash çalışma zamanı, hata olayına abone olan yöntemlere bu sınıfın bir örneğini iletir.

Belirli bir yöntemin bir hatayı mı yoksa hata olayını mı tetikleyeceğini belirlemek için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki yöntem girişine bakın.

Önemli kavramlar ve terimler

Aşağıdaki referans listesinde, hata işleme rutinlerinin programlanmasıyla ilgili önemli kavramlar yer alır:

Senkronize olmayan Hemen sonuç sağlamayan, bunun yerine olay şeklinde bir sonuç (veya hata) veren bir yöntem çağrısı gibi program komutu.

Yakalamak Bir istisna (çalışma zamanı hatası) gerçekleşip kodunuz istisnayı fark ettiğinde bu kod istisnayı *yakalamış* olur. Bir istisna yakalandığında, Flash çalışma zamanı istisnayı diğer ActionScript kodlarına bildirme işlemini durdurur.

Hata ayıklayıcı sürümü Flash çalışma zamanının çalışma zamanı hatalarını kullanıcılara bildiren Flash Player hata ayıklayıcı sürümü veya AIR Hata Ayıklama Başlatıcısı'na (ADL) benzeyen özel bir sürümü. Standart Flash Player veya Adobe AIR sürümünde (çoğu kullanıcının kullandığı), ActionScript kodunuz tarafından işlenmeyen hatalar yoksayılır. Hata ayıklayıcı sürümlerde (Adobe Flash CS4 Professional ve Adobe Flash Builder uygulamalarıyla dahil edilmiştir), işlenmemiş bir hata oluştuğunda bir uyarı mesajı görüntülenir.

İstisna Bir uygulama çalışırken oluşan ve Flash çalışma zamanının kendiliğinden çözemediği bir hata.

Yeniden atma Kodunuz bir istisna yakaladığında, Flash çalışma zamanı onu artık istisnaya ait diğer nesnelere bildirmez. İstisnanın diğer nesnelere bildirilmesi önemliyse, kodunuzun bildirim işlemini yeniden başlatmak için istisnayı *yeniden fırlatma*sı gerekir.

Senkronize Hemen sonuç sağlayan (veya hemen bir hata fırlatan), yani yanıtın aynı kod bloğunda kullanılabildiğini bir program komutudur (bir yöntem çağrısı gibi).

Atma Flash çalışma zamanı uygulamasına bir hata oluştuğunu bildirme (ve bunun sonucunda diğer nesnelere ve ActionScript koduna da bildirme) eylemi, hata *atma* olarak bilinir.

Hata türleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamaları geliştirip çalıştırdığınızda, farklı hata türleri ve hata terminolojisiyle karşılaşırsınız. Aşağıdaki listede önemli hata türleri ve terimler verilmiştir:

- *Derleme zamanı hataları*, kod derlemesi sırasında ActionScript derleyicisi tarafından verilir. Kodunuzdaki sözdizimi hataları uygulamanızın oluşturulmasını önlediğinde derleme zamanı hataları oluşur.
- *Çalışma zamanı hataları*, uygulamanızı derledikten sonra çalıştırdığınızda oluşur. Çalışma zamanı hataları, bir SWF dosyası Flash çalışma zamanında oynatılırken ortaya çıkan hataları temsil eder). Çoğu durumda çalışma zamanı hatalarını oluştuğunda işleyebilerek bunları kullanıcıya bildirebilir ve uygulamanızın devam etmesi için gerekli adımları uygulayabilirsiniz. Söz konusu hata, uzak bir web siteye bağlanılamama ya da verileri yükleyememe gibi kritik bir hataysa, uygulamanızın düzgün bir biçimde sonlandırılmasını sağlamak için hata işlemeyi kullanabilirsiniz.

- *Senkronize hatalar*, bir işlev çağrıldığında gerçekleşen çalışma zamanı hatalarıdır. Örneğin belirli bir yöntemi kullanmayı denediğinizde ve yönteme ilettiğiniz argüman geçersiz olduğunda Flash çalışma zamanı bir istisna atar. Hataların çoğu eşzamanlı—deyim çalıştırıldığında—oluşur ve denetim akışı hemen en uygun `catch` deyimine iletilir.

  Örneğin, program dosyayı karşıya yüklemeden önce `browse()` yöntemi çağrılmadığından, aşağıdaki kod alıntısı bir çalışma zamanı hatası atar:

  ```
  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.
  }
  ```

  Bu durumda, Flash Player dosya karşıya yükleme işleminden önce `browse()` yönteminin çağrılmadığını belirlediğinden, eşzamanlı olarak bir çalışma zamanı hatası atılır.

  Eşzamanlı hata işleme hakkında ayrıntılı bilgi için, bkz. "Bir uygulamada eşzamanlı hataları işleme" sayfa 58.

- *Senkronize olmayanhatalar* normal program akışı dışında görülen çalışma zamanı hatalarıdır. Bunlar olayları meydana getirir ve olay dinleyicileri bunları yakalar. Senkronizasyonsuz işlem, bir işlevin bir işlemi başlatıp işlemin tamamlanmasını beklemediği bir işlemdir. Uygulamanın ve kullanıcının bazı işlemleri denemesini beklemek için bir hata olayı dinleyicisi oluşturabilirsiniz. Böylece işlem başarısız olursa, hatayı olay dinleyicisi ile yakalayıp hata olayına yanıt verirsiniz. Daha sonra olay dinleyicisi hata olayına düzgün şekilde yanıt vermek için bir olay işleyicisi işlevini çağırır. Örneğin, olay işleyicisi, kullanıcıdan hatayı çözümlemesini isteyen bir iletişim kutusunu başlatabilir.

  Daha önce sunulan dosya karşıya yükleme eşzamanlı hatası örneğini göz önünde bulundurun. Dosya karşıya yüklemesine başlamadan önce `browse()` yöntemini başarıyla çağırırsanız, Flash Player birçok olay gönderir. Örneğin, karşıya yükleme başladığında `open` olayı gönderilir. Dosya karşıya yükleme işlemi başarıyla tamamlandığında, `complete` olayı gönderilir. Olay işleme, senkronize olmadığı (yani belirli, bilinen ve önceden belirlenmiş zamanlarda gerçekleşmeyen) için, aşağıdaki kodun da gösterdiği üzere `addEventListener()` yöntemini kullanın:

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

Eşzamansız hata işleme hakkında ayrıntılı bilgi almak için, bkz. "Hata olaylarını ve durumunu yanıtlama" sayfa 63.

- *Yakalanmamış istisnalar*, kendilerine yanıt verilmesi için karşılık gelen bir mantık olmadan atılan hatalardır (`catch` deyimi gibi). Uygulamanız bir hata atarsa ve hatayı işlemek için geçerli veya daha yüksek düzeyde uygun bir `catch` deyimi veya olay işleyicisi bulunamazsa, hata, yakalanmamış istisna olarak değerlendirilir.

  Yakalanmamış bir hata ortaya çıktığında, çalışma zamanı `uncaughtError` olayı gönderir. Bu olay "global hata işleyici" olarak bilinir. Bu olay SWF'nin `LoaderInfo.uncaughtErrorEvents` özelliği ile sağlanan UncaughtErrorEvents nesnesi tarafından gönderilir. `uncaughtError` olayı için kayıtlı bir dinleyici yoksa, çalışma zamanı yakalanmamış hataları göz ardı eder ve hata SWF'yi kapatmadığı sürece uygulamayı yürütmeye çalışır.

  `uncaughtError` olayının gönderilmesine ek olarak Flash çalışma zamanının hata ayıklama programı, sürümleri geçerli komut dosyasını sonlandırarak yakalanmamış hatalara yanıt verir. Daha sonra, yakalanmamış hatayı `trace` ifadesi çıktısında görüntüler veya hata mesajını günlük dosyasına yazar. Eğer istisna nesnesi Hata sınıfı veya herhangi bir alt sınıfının örneğiyse, çıktıda yığın izleme bilgisi de görüntülenir. Flash çalışma zamanlarının hata ayıklayıcı sürümünün kullanılmasıyla ilgili daha fazla bilgi için, bkz. "Flash çalışma zamanlarının hata ayıklayıcı sürümleri" sayfa 57.

  Not: *Bir uncaughtError olayını işlerken uncaughtError işleyicisinden bir hata olayı atılırsa olay işleyicisi bir kereden çok çağrılır. Bu sonsuz bir istisna döngüsüne neden olur. Bu tarz bir senaryodan kaçınmanız önerilir.*

ActionScript 3.0'da hata işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamaların çoğu hata işleme mantığı oluşturulmadan çalıştırılabildiğinden, geliştiriciler kendi uygulamalarında hata işleme oluşturulmasını erteleyebilir. Ancak, hata işleme olmazsa bir şeyler beklendiği gibi çalışmadığında bir uygulama kullanıcıya kolayca rahatsızlık verebilir. ActionScript 2.0'da belirli bir mesajla istisna atmak için özel işlevlerde mantık oluşturmanıza olanak sağlayan bir Error sınıfı vardır. Hata işleme, kullanıcı dostu bir uygulama oluşturulmasında kritik bir faktör olduğundan, ActionScript 3.0'da hataların yakalanması için genişletilmiş bir mimari vardır.

Not: *Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda, her yöntemdeki olası tüm istisnalar olmasa da birçok yöntem tarafından atılan istisnalar belgelenmiştir. Açıklamada, bir yöntemin fırlattığı istisnaların bazıları listense bile bir yöntem, sözdizimi hataları için veya yöntem açıklamasında açıkça bildirilmeyen başka sorunlar için bir istisna fırlatılabilir.*

ActionScript 3.0 hata işleme öğeleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, hata işleme için arasında şunların da yer aldığı birçok aracı içerir:

- Error sınıfları. ActionScript 3.0, hata nesneleri üretebilecek durumların kapsamını genişletmek için geniş bir Error sınıfı aralığına sahiptir. Her Error sınıfı, istem hatalarıyla (MemoryError koşulu gibi), kodlama hatalarıyla (ArgumentError koşulu gibi), ağ iletişimi ve iletişim hatalarıyla (URIError koşulu gibi) veya diğer durumlarla ilgili belirli hata koşullarının işlenmesine ve yanıtlanmasına yardımcı olur. Sınıfların her biriyle ilgili daha fazla bilgi almak için, bkz. "Error sınıflarını karşılaştırma" sayfa 66.
- Daha az sessiz arıza Önceki Flash Player sürümlerinde, yalnızca `throw` deyimini açıkça kullandığınızda hatalar oluşturulup bildirilirdi. Yerel ActionScript yöntemleri ve özellikleri, Flash Player 9 ve üstü Flash çalışma zamanları için çalışma zamanı hataları fırlatır. Bu hatalar, bu istisnaların işlenmelerini meydana geldikleri zaman daha etkili yapmanızı ve daha sonra her birine ayrı ayrı ulaşmanızı sağlar.
- Hata ayıklama sırasında görüntülenen hata mesajlarını temizleyin. Flash çalışma zamanının hata ayıklayıcı sürümünü kullandığınızda sorunlu kodlar veya durumlar dayanıklı hata mesajları oluşturur ve bu da belirli bir kod bloğunun başarısız olma nedenlerini kolayca tanımlamanıza yardımcı olur. Bu mesajlar, hataların daha etkili giderilmesini sağlar. Daha fazla bilgi için, bkz. "Flash çalışma zamanlarının hata ayıklayıcı sürümleri" sayfa 57.
- Kesin hatalar, kullanıcılara net hata mesajlarının görüntülenmesine olanak sağlar. Önceki Flash Player sürümlerinde `FileReference.upload()` yöntemi, `upload()` çağrısı başarısız olduğunda bir `false` Boolean değeri döndürerek beş olası hatadan birini belirtirdi. ActionScript 3.0'da `upload()` yöntemini çağırdığınızda bir hata ortaya çıkarsa, son kullanıcılara daha doğru hata mesajları göstermenizi sağlayan dört belirli hata size yardımcı olur.
- Daraltılmış hata işleme. Birçok yaygın durum için belirgin hatalar atılır. Örneğin ActionScript 2.0'de, bir FileReference nesnesi doldurulmadan önce, `name` özelliği `null` değerine sahip olur (böylece `name` özelliğini kullanabilmeniz veya görüntüleyebilmeniz için öncelikle bu değerin ayarlanmış olduğundan veya `bo` olmadığından emin olmanız gerekir). ActionScript 3.0'da, `name` özelliği doldurulmadan bu özelliğe erişmeyi denerseniz, Flash Player veya AIR uygulaması, değerin ayarlanmamış olduğunu size bildiren bir IllegalOperationError hatasını atar ve siz de `try..catch..finally` bloklarını kullanarak hatayı işleyebilirsiniz. Daha fazla bilgi için, bkz. "try..catch..finally deyimlerini kullanma" sayfa 58.
- Önemli performans sorunlarının olmaması Hataları işlemek için `try..catch..finally` bloklarının kullanılması, önceki ActionScript sürümlerine göre çok daha az ek kaynak tüketir veya hiç ek kaynak tüketmez.
- Belirli eşzamansız hata olayları için dinleyiciler oluşturmanıza olanak sağlayan bir ErrorEvent sınıfı. Daha fazla bilgi için, bkz. "Hata olaylarını ve durumunu yanıtlama" sayfa 63.

Hata işleme stratejileri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kodunuzda hata işleme mantığı oluşturmasanız bile, uygulamanız sorun oluşturan bir durumla karşılaşmadığı sürece başarılı şekilde çalışabilir. Ancak hataları etkin şekilde işlemezseniz ve uygulamanız bir sorunla karşılaşırsa, kullanıcılarınız uygulamanızın neden başarısız olduğunu asla bilmez.

Uygulamanızda hata işlemeye yönelik çeşitli yöntemler vardır. Aşağıdaki listede, üç önemli hata işleme seçeneği özetlenmektedir:

- `try..catch..finally` deyimlerini kullanma. Bu ifadeler ortaya çıktıkça senkronize hataları yakalar. Kod çalıştırmasının çeşitli düzeylerinde istisnaları yakalamak için, deyimlerinizi bir hiyerarşide yuvalayabilirsiniz. Daha fazla bilgi için, bkz. "try..catch..finally deyimlerini kullanma" sayfa 58.
- Kendi özel hata nesnelerinizi oluşturma. Uygulamanızda, yerleşik hata türlerinin kapsamadığı belirli işlemleri izlemek üzere kendi özel hata nesnelerinizi oluşturmak için Error sınıfını kullanabilirsiniz. Daha sonra özel hata nesnelerinizde `try..catch..finally` deyimlerini kullanabilirsiniz. Daha fazla bilgi için, bkz. "Özel hata sınıfları oluşturma" sayfa 62.
- Hata olaylarını yanıtlamak için olay dinleyicileri ve işleyicileri yazma. Bu stratejiyi kullanarak, try..'da fazla kod çoğaltmadan benzer olayları işlemenize olanak veren global hata işleyicileri oluşturabilirsiniz.`catch..finally` bloklarında çok sayıda kod kopyalamadan benzer olayları işlemenize olanak sağlayan genel hata işleyicileri oluşturabilirsiniz. Ayrıca bu yaklaşımı kullanarak eşzamansız hataları daha fazla yakalayabilirsiniz. Daha fazla bilgi için, bkz. "Hata olaylarını ve durumunu yanıtlama" sayfa 63.

Flash çalışma zamanlarının hata ayıklayıcı sürümleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe, hata ayıklama çalışmalarına yardımcı olmak için geliştiricilere Flash çalışma zamanlarının özel sürümlerini sağlar. Adobe Flash Professional veya Adobe Flash Builder uygulamasını yüklediğinizde, Flash Player'ın hata ayıklayıcı sürümünün bir kopyasını edinirsiniz. Bu araçlardan herhangi birini yüklediğinizde veya Adobe AIR SDK'nin parçası olarak yükleme yaptığınızda, ADL adı verilen Adobe AIR uygulamalarında hata ayıklamaya yarayan bir hizmet de edinirsiniz.

Flash Player ve Adobe AIR uygulamalarının hata ayıklayıcı sürümlerinin ve yayınlama sürümlerinin hata belirtme şekillerinde dikkate değer bir fark vardır. Hata ayıklayıcı sürümleri, hata türünü (örn. genel Error, IOError veya EOFError), hata sayısını ve okunabilir bir hata mesajını gösterir. Yayınlama sürümleri yalnızca hata türünü ve hata sayısını gösterir. Örneğin, şu kodu göz önünde bulundurun:

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

Flash Player uygulamasının hata ayıklayıcı sürümünde `readBoolean()` yöntemi bir EOFError hatası attıysa `tf` metin alanında şu mesaj görüntülenir: "EOFError: Hata No 2030: Dosya sonuna ulaşıldı."

Flash Player veya Adobe AIR uygulamasının yayınlama sürümünde aynı kod şu metni görüntüler: "EOFError: Error #2030."

Not: *Hata ayıklayıcı oynatıcıları "allComplete" adlı bir olay yayınlar; "allComplete" adlı özel olaylar oluşturmayın. Aksi takdirde, hata ayıklama sırasında beklenmeyen davranışlarla karşılaşırsınız.*

Yayınlama sürümlerinde kaynakları ve boyutu minimum tutmak için hata mesajı dizeleri verilmez. Belgelerde bir hata mesajıyla ilişkisini bulmak için (Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nun ekleri) hata sayısına bakabilirsiniz. Alternatif olarak, tam mesajı görmek için Flash Player ve AIR uygulamasının hata ayıklayıcı sürümlerini kullanarak hatayı yeniden oluşturabilirsiniz.

Bir uygulamada eşzamanlı hataları işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

En yaygın hata işleme, senkronize hata işleme mantığıdır, burada bir uygulama çalışırken senkronize hataları yakalamak için kodunuza deyimler eklersiniz. Bu hata işleme türü, işlevler başarısız olduğunda uygulamanızın çalışma zamanı hatalarını fark etmesine ve gidermesine olanak sağlar. Eşzamanlı hata yakalama mantığı, `try..catch..finally` deyimlerini içerir, bu ifadeler bir işlemi dener, Flash çalışma zamanından gelen herhangi bir hata yanıtını yakalar ve son olarak başarısız olan işlemi işlemek için başka bir işlemi çalıştırır.

try..catch..finally deyimlerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Eşzamanlı çalışma zamanı hatalarıyla çalışırken, hataları yakalamak için `try..catch..finally` deyimlerini kullanın. Bir çalışma zamanı hatası oluştuğunda, Flash çalışma zamanı bir istisna atar, başka bir deyişle, normal çalıştırmayı askıya alır ve Error türü için özel bir nesne oluşturur. Daha sonra Error nesnesi kullanılabilir birinci `catch` bloğuna atılır.

`try` deyimi, hata oluşturma olasılığına sahip deyimleri kapsar. Her zaman `try` deyimiyle `catch` deyimini kullanırsınız. `try` ifade bloğundaki ifadelerden birinde hata algılanırsa o `try` ifadesine eklenen `catch` deyimleri çalıştırılır.

`finally` deyimi, `try` bloğunda bir hata oluşsa da oluşmasa da yürütülecek olan deyimleri kapsar. Herhangi bir hata yoksa, `try` blok deyimleri tamamlandıktan sonra `finally` bloğundaki deyimler çalıştırılır. Bir hata varsa, ilk olarak `catch` deyimi sonra da `finally` bloğundaki deyimler çalıştırılır.

Aşağıdaki kod, `try..catch..finally` deyimlerinin kullanılmasına yönelik sözdizimini gösterir:

```
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.
}
```

Her `catch` deyimi, işlediği belirli bir istisna türünü tanımlar. `catch` deyimi yalnızca Error sınıfının alt sınıfları olan hata sınıflarını belirtebilir. Her `catch` deyimi sırayla denetlenir. Yalnızca fırlatılan hata türüyle eşleşen ilk `catch` ifadesi çalıştırılır. Başka bir deyişle, ilk olarak yüksek düzeydeki Error sınıfını ve sonra da Error sınıfının bir alt sınıfını denetlerseniz, yalnızca yüksek düzeydeki Error sınıfı eşleşir. Aşağıdaki kod bu noktayı gösterir:

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

Önceki kod, şu çıktıyı görüntüler:

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

ArgumentError hatasını doğru şekilde yakalamak için, aşağıdaki kodun gösterdiği gibi, en özel hata türlerinin ilk sırada ve daha genel hata türlerinin daha aşağıda listelendiğinden emin olmanız gerekir:

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

ActionScript API'sindeki birçok yöntem ve özellik, çalıştırılırken hatalarla karşılaşırsa çalışma zamanı hataları atar. Örneğin, aşağıdaki kodda gösterildiği gibi, yöntem ses akışını kapatamazsa, Sound sınıfındaki `close()` yöntemi bir IOError hatası atar:

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

Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nu öğrendikçe, her yöntemin açıklamasında ayrıntıyla anlatıldığı gibi hangi yöntemlerin istisna attığını fark edeceksiniz.

throw deyimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash çalışma zamanları çalışan uygulamalarınızda hatalarla karşılaştığı zaman istisna atar. Ayrıca, `throw` deyimini kullanarak kendiniz de açıkça istisnalar atabilirsiniz. Adobe, açıkça hata atarken Error sınıfının veya bu sınıfın alt sınıflarının örneklerini atmanızı önerir. Aşağıdaki kod, Error sınıfının bir örneğini (`MyErr`) atan ve hata atıldıktan sonra yanıt vermek için bir işlev (`myFunction()`) çağıran `throw` deyimini gösterir:

```
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.
}
```

`catch` deyimlerinin, en belirli veri türleri ilk sırada olacak şekilde sıralandığına dikkat edin. Sayı veri türü için `catch` ifadesi ilk sırada listelenmiş olsaydı, uint veri türü için catch ifadesi çalıştırılamazdı.

Not: *Java programlama dilinde, bir istisna atabilen her işlevin bu gerçeği bildirmesi ve atabildiği istisna sınıflarını işlev bildirimine eklenen bir `throws` deyiminde listelemesi gerekir. ActionScript bir işlev tarafından fırlatılan istisnaları belirtmenizi gerektirmez.*

Basit bir hata mesajını görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yeni istisna ve hata olayı modelinin en büyük avantajlarından biri, kullanıcılara eylemin ne zaman ve neden başarısız olduğunu bildirmenize olanak sağlamasıdır. Sizin göreviniz, mesajı görüntülemek ve yanıt olarak seçenekler sunmak için kod yazmaktır.

Aşağıdaki kod, hatayı bir metin alanında görüntülemek için basit bir `try..catch` deyimini gösterir:

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

ActionScript 3.0, daha çeşitli hata sınıflarını ve yerleşik derleyici hatalarını kullanarak, bir şeyin neden başarısız olduğuna dair önceki ActionScript sürümlerinden daha fazla bilgi sunar. Bu bilgi, iyi hata işleme özelliğine sahip daha kararlı uygulamalar oluşturmanıza olanak sağlar.

Hataları yeniden atma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulama oluşturduğunuzda, hatayı düzgün şekilde işleyemezseniz hatayı yeniden fırlatmanızı gerektirecek birçok durum vardır. Örneğin, aşağıdaki kod yuvalanmış bir `try..catch` bloğunu gösterir, bu, yuvalanmış blok `catch` bloğu hatayı işleyemezse özel bir ApplicationError hatasını yeniden atar:

```
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);
}
```

Önceki kod parçasındaki çıktı şöyle olur:

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

Yuvalanmış `try` bloğu, sonraki `catch` bloğu tarafından yakalanan özel bir ApplicationError hatası atar. Bu yuvalanmış `catch` bloğu, hatayı işlemeye çalışır ve başarısız olursa, ApplicationError nesnesini kapsayan `try..catch` bloğuna atar.

Özel hata sınıfları oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript'te kendi özelleştirilmiş hata sınıflarınızı oluşturmak için standart Error sınıflarından birini genişletebilirsiniz. Kendi hata sınıflarınızı oluşturmanızın birçok nedeni vardır:

- Uygulamanız için benzersiz olan belirli hataları veya hata gruplarını tanımlamak için.

  Örneğin, Flash çalışma zamanı tarafından yakalanan bu hatalara ek olarak kendi kodunuz tarafından verilen hatalar için farklı eylemler uygulayın. `try..catch` bloklarında yeni hata veri türünü izlemek için Error sınıfının bir alt sınıfını oluşturabilirsiniz.

- Uygulamanız tarafından oluşturulan hatalara yönelik benzersiz hata görüntüleme yetenekleri sağlamak için.

  Örneğin, hata mesajlarınızı belirli bir şekilde formatlayan yeni bir `toString()` yöntemi oluşturabilirsiniz. Ayrıca bir hata kodunu alıp kullanıcının dil tercihine göre uygun mesajı alan bir `lookupErrorString()` yöntemini de tanımlayabilirsiniz.

Özelleştirilmiş bir hata sınıfının çekirdek ActionScript Error sınıfını genişletmesi gerekir. Aşağıda, Error sınıfını genişleten özelleştirilmiş bir AppError sınıfı örneği verilmiştir:

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

Aşağıda, projenizde AppError öğesinin kullanılmasına bir örnek gösterilmektedir:

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

Not: *Alt sınıfınızda `Error.toString()` yöntemini geçersiz kılmak isterseniz bu yönteme bir `...(rest)` parametresi vermeniz gerekir. ActionScript 3.0'ın esas aldığı ECMAScript dil belirtimi, `Error.toString()` yöntemini bu şekilde tanımlar ve ActionScript 3.0, geriye doğru uyumluluk için bu yöntemi aynı şekilde tanımlar. Bu nedenle, `Error.toString()` yöntemini geçersiz kıldığınızda parametreleri tamamen aynı şekilde eşleştirmeniz gerekir. O parametreler yok sayıldığı için, çalışma zamanında `toString()` yönteminize herhangi bir parametre iletmek istemezsiniz.*

Hata olaylarını ve durumunu yanıtlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da hata işleme konusunda en dikkate değer iyileştirmelerden biri, bir uygulama çalışırken senkronize olmayan çalışma zamanı hatalarını yanıtlamak için hata olayı işleme desteğidir. (Eşzamansız hataların bir tanımı için, bkz. "Hata türleri" sayfa 53.)

Hata olaylarını yanıtlamak için olay dinleyicileri ve olay işleyicileri oluşturabilirsiniz. Sınıfların çoğu, diğer olayları gönderdikleri şekilde hata olaylarını da gönderir. Örneğin, XMLSocket sınıfının bir örneği normalde üç olay türü gönderir: `Event.CLOSE`, `Event.CONNECT` ve `DataEvent.DATA`. Ancak bir sorun oluştuğunda, XMLSocket sınıfı, `IOErrorEvent.IOError` veya `SecurityErrorEvent.SECURITY_ERROR` öğesini gönderebilir. Olay dinleyicileri ve olay işleyicileri hakkında daha fazla bilgi için, bkz. "Olayları işleme" sayfa 126.

Hata olayları, iki kategoriden birisine girer:

- ErrorEvent sınıfını genişleten hata olayları

  flash.events.ErrorEvent sınıfı, çalışan bir uygulamada ağ iletişimi ve iletişim işlemleriyle ilgili çalışma zamanı hatalarının yönetilmesine yönelik özellikleri ve yöntemleri içerir. AsyncErrorEvent, IOErrorEvent ve SecurityErrorEvent sınıfları, ErrorEvent sınıfını genişletir. Bir Flash çalışma zamanının hata ayıklayıcı sürümünü kullanıyorsanız, bir iletişim kutusu, oynatıcının karşılaştığı dinleyici işlevleri olmadan çalışma zamanında hata olaylarını size bildirir.

- Durum tabanlı hata olayları

Durum tabanlı hata olayları, ağ iletişimi ve iletişim sınıflarının `netStatus` ve `status` özellikleriyle ilgilidir. Bir Flash çalışma zamanı, veri okurken ya da yazarken bir sorunla karşılaşırsa, `netStatus.info.level` veya `status.level` özelliklerinin değeri (kullandığınız sınıf nesnesine bağlı olarak), `"error"` değerine ayarlanır. `level` özelliğinin, olay işleyicisi işlevinizde `"error"` değerini içerip içermediğini kontrol ederek bu hatayı yanıtlarsınız.

Hata olaylarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ErrorEvent sınıfı ve bu sınıfın alt sınıfları, veri okumaya veya yazmaya çalıştıkça, Flash çalışma zamanları tarafından gönderilen hataların işlenmesine yönelik hata türlerini içerir.

Aşağıdaki örnek, yerel bir dosya okunmaya çalışılırken algılanan hataları görüntülemek için, bir `try..catch` deyimini ve hata olayı işleyicilerini kullanır. Kullanıcıya seçenekler sağlamak için daha gelişmiş işleme kodu ekleyebilir veya "your error-handling code here": yorumunun belirttiği yerde otomatik olarak hatayı işleyebilirsiniz.

```
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
        }
    }
}
```

Durum değişikliği olaylarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash çalışma zamanları bir uygulama çalışırken `level` özelliğini destekleyen sınıflara ilişkin `netStatus.info.level` veya `status.level` özelliklerinin değerini dinamik olarak değiştirir. `netStatus.info.level` özelliğine sahip sınıflar, NetConnection, NetStream ve SharedObject sınıflarıdır. `status.level` özelliğine sahip sınıflar, HTTPStatusEvent, Camera, Microphone ve LocalConnection sınıflarıdır. `level` değerindeki değişikliğe yanıt vermek ve iletişim hatalarını izlemek için bir işleyici işlevi yazabilirsiniz.

Aşağıdaki örnek, `level` özelliğinin değerini test etmek için bir `netStatusHandler()` işlevini kullanır. `level` özelliği bir hatayla karşılaşıldığını belirtirse, kod “Video stream failed” mesajını izler.

```
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);
        }
    }
}
```

Error sınıflarını karşılaştırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript, önceden tanımlı birçok Error sınıfı sağlar. Ancak, kendi kodunuzda da aynı Error sınıflarını kullanabilirsiniz. ActionScript 3.0'da iki ana Error sınıfı türü vardır: ActionScript çekirdek Error sınıfları ve flash.error paketi Error sınıfları. flash.error paketinde, ActionScript 3.0 uygulama geliştirmesi ve hata ayıklamasına yardımcı olmak üzere sunulan ek sınıflar bulunur.

En önemli hata sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

En önemli hata sınıfları arasında Error, ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError ve VerifyError sınıfları bulunur. Bu sınıfların her biri üst düzey ad alanında bulunur.

Sınıf adı	Açıklama	Notlar
Error	Error sınıfı, istisnaları fırlatmak için kullanılır ve ECMAScript'te tanımlanmış diğer istisna sınıfları için temel sınıftır: EvalError, RangeError, ReferenceError, SyntaxError, TypeError ve URIError.	Error sınıfı, tüm çalışma zamanı hataları için temel sınıf görevi görür ve tüm özel hata sınıfları için önerilen temel sınıftır.
ArgumentError	ArgumentError sınıfı, bir işlev çağrısı sırasında sağlanan parametre değerleri, o işlev için tanımlanmış parametrelerle eşleşmediğinde gerçekleşen bir hatayı temsil eder.	Argüman hatalarının bazı örnekleri arasında şunlar yer alır: • Bir yönteme çok az veya çok fazla argüman sağlanması. • Bir argümanın bir numaralandırma üyesi olmasının beklenmesi ancak olmaması.
EvalError	Function sınıfının yapıcısına herhangi bir parametre iletilirse veya kullanıcı kodu `eval()` işlevini çağırırsa bir EvalError istisnası atılır.	ActionScript 3.0'da `eval()` işlevi desteği kaldırılmış olup bu işlevi kullanma girişimleri de bir hatanın verilmesine yol açar. Önceki Flash Player sürümleri, ada göre değişkenlere, özelliklere, nesnelere veya film kliplerine erişmek için `eval()` işlevini kullanırdı.
RangeError	Sayısal bir değer, kabul edilebilir aralığın dışında kalırsa bir RangeError istisnası fırlatılır.	Örneğin, bir gecikme negatif olunca veya sonlu olmayınca Timer sınıfı tarafından bir RangeError verilir. Ayrıca geçersiz derinlikte bir görüntüleme nesnesini eklemeyi denediğinizde de bir RangeError atılabilirdi.
ReferenceError	Mühürlü (dinamik olmayan) bir nesnede tanımsız bir özelliğe başvurulduğunda ReferenceError istisnası atılır. ActionScript 3.0'dan önceki ActionScript derleyici sürümleri, `undefined` bir özelliğe erişilmeye çalışıldığında bir hata atmazdı. Ancak ActionScript 3.0, bu koşulda ReferenceError istisnasını atar.	Tanımsız değişkenlerin istisnaları, olası hataları işaret ederek yazılım kalitesini yükseltmenize yardımcı olur. Ancak, değişkenlerinizi başlatmak zorunda olmaya alışkın değilseniz bu yeni ActionScript davranışı, kodlama alışkanlıklarınızda bazı değişiklikler yapmanızı gerektirebilir.
SecurityError	Bir güvenlik ihlali gerçekleşip erişim reddedildiğinde SecurityError istisnası atılır.	Güvenlik hatalarının bazı örnekleri arasında şunlar yer alır: • Sanal güvenlik alanı sınırından yetkisiz bir özellik erişimi veya yöntem çağrısı yapılması. • Sanal güvenlik alanının izin vermediği bir URL'ye erişim girişiminde bulunulması. • Bir bağlantı noktasına soket bağlantısı yapılmaya çalışılmış ancak gerekli soket ilke dosyası bulunmamıştır. • Kullanıcının kamerasına veya mikrofonuna bir girişimde bulunuldu ve kullanıcı cihaza erişim yapılmasını reddetti.
SyntaxError	ActionScript kodunuzda bir ayrıştırma hatası oluştuğunda SyntaxError istisnası atılır.	SyntaxError hatası şu koşullarda atılabilir: • RegEp sınıfı geçersiz bir düzenli ifade ayrıştırdığında ActionScript, SyntaxError istisnaları fırlatır. • XMLDocument sınıfı geçersiz bir XML ayrıştırdığında ActionScript, SyntaxError fırlatır.

Sınıf adı	Açıklama	Notlar
TypeError	Bir işlenenin gerçek türü beklenen türden farklı olduğunda TypeError istisnası atılır.	TypeError hatası şu koşullarda atılabilir: • Bir işlevin veya yöntemin gerçek parametresi, biçimsel parametre türüne zorlanamadığında. • Bir değişkene bir değer atandığında ve bu değer değişkenin türüne zorlanamadığında. • `is` veya `instanceof` operatörünün sağ tarafı geçerli bir tür olmadığında. • `super` anahtar sözcüğü kuraldışı olarak kullanıldığında. • Özellik arama birden çok bağlamaya yol açtığından belirsiz olduğunda. • Uyumsuz bir nesnede bir yöntem çağrılır. Örneğin, RegExp sınıfındaki bir yöntem genel bir nesneye "eklenip" sonra çağrıldığında bir TypeError istisnası fırlatılır.
URIError	Genel URI işleme işlevlerinden biri, tanımına uymayacak şekilde kullanıldığında URIError istisnası atılır.	URIError hatası şu koşullarda atılabilir: Geçerli bir URI bekleyen Flash Player API'si işlevi için, `Socket.connect()` gibi geçersiz bir URI belirtildiğinde.
VerifyError	Yanlış biçimlendirilmiş veya bozuk bir SWF dosyasıyla karşılaşıldığında VerifyError istisnası atılır.	Bir SWF dosyası başka bir SWF dosyasını yüklediğinde, üst SWF, yüklenen SWF tarafından oluşturulan bir VerifyError hatasını yakalayabilir.

flash.error paketi Error sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

flash.error paketi, Flash çalışma zamanı API'sinin parçası olarak değerlendirilen Error sınıflarını içerir. Açıklaması yapılan Error sınıflarının tersine, flash.error paketi, Flash çalışma zamanına özgü hata olaylarıyla iletişim kurar (Flash Player ve Adobe AIR gibi).

Sınıf adı	Açıklama	Notlar
EOFError	Kullanılabilir verilerin sonunun ötesini okumaya çalıştığınızda bir EOFError istisnası atılır.	Örneğin, IDataInput arabiriminde okuma yöntemlerinden biri çağrıldığında ve okuma isteğini karşılamak için yeterli veri olmadığında bir EOFError atılır.
IllegalOperationError	Bir yöntem uygulanmadığında veya uygulama geçerli kullanımı kapsamadığında, bir IllegalOperationError istisnası atılır.	Kuraldışı işlem hatası istisnalarının örnekleri arasında şunlar yer alır: • DisplayObjectContainer gibi bir temel sınıfın, Sahne Alanı'nın destekleyemeyeceği kadar çok işlevsellik sağlaması. Örneğin, Sahne alanı'nda (`stage.mask` öğesini kullanarak) bir maske almayı veya ayarlamayı denerseniz, Flash çalışma zamanı, "Stage sınıfı bu özellik veya yöntemi uygulamıyor" mesajıyla birlikte IllegalOperationError fırlatır. • Bir alt sınıfın gerektirmediği ve desteklemek istemediği bir yöntemi miras alması. • Erişilebilirlik desteği olmadan Flash Player derlendiğinde belirli yöntemlerin çağrılması. • Flash Player uygulamasının çalışma zamanı sürümünden yalnızca geliştirme özelliklerinin çağrılması. • Zaman çizelgesine yerleştirilmiş bir nesnenin adını ayarlamaya çalışmanız.
IOError	Bir G/Ç istisnası gerçekleştiğinde IOError istisnası atılır.	Örneğin, bağlanmamış veya bağlantısı kesilmiş bir sokette okuma-yazma işlemi girişiminde bulunulduğunda bu hatayı alırsınız.
MemoryError	Bellek ayırma isteği başarısız olduğunda bir MemoryError istisnası atılır.	Varsayılan olarak, ActionScript Virtual Machine 2, bir ActionScript programının ayırabileceği bellek miktarına bir sınırlama getirmez. Masaüstü bir sistemde bellek tahsisi hatalarına az rastlanır. Sistem bir işlem için gerekli belleği ayıramadığında bir hata atıldığını görürsünüz. Bu nedenle masaüstü bir bilgisayarda ayırma isteği aşırı yüksek miktarda olmadığı sürece bu istisna çok nadir fırlatılır; örneğin 32-bit Microsoft ® Windows® programı yalnızca adres alanının 2 GB'ına erişebildiğinden 3 milyar bayt isteği olanaksızdır.
ScriptTimeoutError	15 saniyelik komut dosyası zaman aşımı aralığına ulaşıldığında bir ScriptTimeoutError istisnası atılır. ScriptTimeoutError istisnasını yakalayarak komut dosyası zaman aşımını daha düzgün şekilde işleyebilirsiniz. Herhangi bir istisna işleyicisi yoksa, yakalanmayan istisna işleyicisi bir hata mesajı içeren bir iletişim kutusunu görüntüleyecektir.	Kötü amaçlı bir geliştiricinin istisnayı yakalayıp sonsuz döngüde kalmasını önlemek için, yalnızca belirli bir komut dosyasında atılan birinci ScriptTimeoutError istisnası yakalanabilir. Sonraki bir ScriptTimeoutError istisnası, kodunuz tarafından yakalanamaz; hemen yakalanmamış istisna işleyicisine gider.
StackOverflowError	Komut dosyası için kullanılabilir olan yığın tükendiğinde StackOverflowError istisnası atılır.	StackOverflowError istisnası, sonsuz bir yineleme oluştuğunu belirtebilir.

Hataları işleme örneği: CustomErrors uygulaması

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

CustomErrors uygulaması, bir uygulama oluşturulurken özel hatalarla çalışma tekniklerini gösterir. Bu teknikler şunlardır:

- XML paketini doğrulama
- Özel bir hata yazma
- Özel hatalar atma
- Bir hata atıldığında bunu kullanıcılara bildirme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. CustomErrors uygulama dosyaları, Samples/CustomError klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
CustomErrors.mxml veya CustomErrors.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası
com/example/programmingas3/errors/ApplicationError.as	Hem FatalError hem de WarningError sınıfları için temel hata sınıfı görevi gören bir sınıf.
com/example/programmingas3/errors/FatalError.as	Uygulama tarafından verilen bir FatalError hatasını tanımlayan bir sınıf. Bu sınıf özel ApplicationError sınıfını genişletir.
com/example/programmingas3/errors/Validator.as	Kullanıcı tarafından sağlanan bir çalışan XML paketini doğrulayan tek bir yöntemi tanımlayan sınıf.
com/example/programmingas3/errors/WarningError.as	Uygulama tarafından verilen bir WarningError hatasını tanımlayan bir sınıf. Bu sınıf özel ApplicationError sınıfını genişletir.

CustomErrors uygulamasına genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulama yüklendiğinde, Flex uygulamaları için `initApp()` yöntemi çağrılır veya Flash uygulamaları için zaman çizelgesi (işlev olmayan) kodu çalıştırılır. Bu kod, Doğrulayan sınıf tarafından doğrulanan örnek XML paketini tanımlar. Aşağıdaki kod çalıştırılır:

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

XML paketi daha sonra Sahne Alanı'nda bir TextArea bileşen örneğinde görüntülenir. Bu adım, XML paketini yeniden doğrulama girişiminde bulunmadan önce XML paketini değiştirmenizi sağlar.

Kullanıcı Doğrulama düğmesini tıklattığında, `validateData()` yöntemi çağrılır. Bu yöntem, Validator sınıfında `validateEmployeeXML()` yöntemini kullanarak çalışan XML paketini doğrular. Aşağıdaki kod, `validateData()` yöntemini gösterir:

```
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);
    }
}
```

İlk olarak, `xmlText` TextArea bileşen örneğinin içerikleri kullanılarak geçici bir XML nesnesi oluşturulur. Daha sonra, özel Doğrulayan sınıfındaki(com.example.programmingas3/errors/Validator.as)`validateEmployeeXML()` yöntemi çağırılır ve parametre olarak geçici XML nesnesine gönderilir. XML paketi geçerliyse, `status` Label bileşen örneği bir başarı mesajını görüntüler ve uygulamadan çıkılır. `validateEmployeeXML()` yöntemi özel bir hata atarsa (başka bir deyişle, bir FatalError, WarningError veya genel Error oluşursa), uygun `catch` deyimi çalıştırılır ve `showFatalError()`, `showWarningError()` veya `showGenericError()` yöntemlerinden birini çağırır. Bu yöntemlerden her biri, kullanıcıya belirli bir hatanın oluştuğunu bildirmek için `statusTex`t adındaki bir metin alanında uygun bir mesajı görüntüler. Ayrıca her yöntem, `status` Label bileşen örneğini belirli bir mesajla günceller.

Çalışan XML paketini doğrulama girişimi sırasında kritik bir hata oluşursa, aşağıdaki kodun gösterdiği gibi, hata mesajı `statusText` metin alanında görüntülenir ve `xmlText` TextArea bileşen örneği ile `validateBtn` Button bileşen örneği devre dışı bırakılır:

```
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();
}
```

Kritik hata yerine bir uyarı hatası oluşursa, hata mesajı `statusText` TextArea örneğinde görüntülenir ancak `xmlText` TextField ve Button bileşen örnekleri devre dışı bırakılmaz. `showWarningError()` yöntemi, özel hata mesajını `statusText` metin alanında görüntüler. Ayrıca mesaj, kullanıcıdan XML doğrulama işlemine devam etmeyi mi yoksa komut dosyasını durdurmayı mı istediğine karar vermesini ister. Aşağıdaki alıntı `showWarningError()` yöntemini gösterir:

```
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;
}
```

Kullanıcı Evet veya Hayır düğmesini tıklattığında, `closeHandler()` yöntemi çağrılır. Aşağıdaki alıntı `closeHandler()` yöntemini gösterir:

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

Kullanıcı Evet'i tıklatarak komut dosyasını durdurmayı seçerse bir FatalError verilerek uygulamanın sonlandırılmasına neden olur.

Özel bir doğrulayıcı oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Özel Validator sınıfı tek bir yöntemi içerir: `validateEmployeeXML()`. `validateEmployeeXML()` yöntemi, doğrulamak istediğiniz XML paketi olan tek bir argüman olan (`employee`) alır. `validateEmployeeXML()` yöntemi şöyledir:

```
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);
    }
}
```

Bir çalışanın doğrulanması için bir (ve tek) maliyet merkezine ait olması gerekir. Çalışan herhangi bir maliyet merkezine ait değilse, yöntem bir FatalError atar ve bu da ana uygulama dosyasında `validateData()` yöntemine köpürür. Çalışan birden çok maliyet merkezine aitse, bir WarningError atılır. XML doğrulayıcısındaki son denetim, kullanıcının tanımlanmış bir sosyal güvenlik numarasına sahip olmasıdır (XML paketinde `ssn` düğümü). Tam olarak bir `ssn` düğümü yoksa, bir FatalError hatası atılır.

`ssn` düğümünün geçerli bir sayı içerdiğinin veya çalışanın tanımlanmış en az bir telefon numarasının ve e-posta adresinin bulunduğundan ve her iki değerin de geçerli olduğundan emin olmak için `validateEmployeeXML()` yöntemine ek denetimler ekleyebilirsiniz. Ayrıca her çalışanın benzersiz olan ve yöneticilerinin kimliğini tanımlayan bir kimliği olacak şekilde XML'i değiştirebilirsiniz.

ApplicationError sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ApplicationError sınıfı hem FatalError hem de WarningError sınıfları için temel hata sınıfı görevini görür. ApplicationError sınıfı, Error sınıfını genişletir ve hata kimliği, önem derecesi ve özel hata kodlarını ve mesajlarını içeren bir XML nesnesi de dahil olmak üzere kendi özel yöntemlerini ve özelliklerini tanımlar. Bu sınıf aynı zamanda her hata türünün önem derecesini tanımlamak için kullanılan iki statik sabiti de tanımlar.

ApplicationError sınıfının yapıcı yöntemi şöyledir:

```
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>;
}
```

XML nesnesindeki her hata düğümü benzersiz bir sayısal kod ve hata mesajı içerir. Aşağıdaki `getMessageText()` yönteminde de görüldüğü gibi, hata mesajları E4X kullanılarak hata koduna göre kolayca aranabilir:

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

`getMessageText()` yöntemi tek bir tam sayı argümanını (`id`) alır ve bir dize döndürür. `id` argümanı, aranacak hatanın hata kodudur. Örneğin, `id` olarak 9001 değerinin iletilmesi, çalışanların yalnızca bir maliyet merkezine atanması gerektiğini belirten hatayı alır. Birden çok hata aynı hata koduna sahipse, ActionScript, yalnızca bulunan birinci sonucun hata mesajını (döndürülen XMLList nesnesinde `message[0]`) döndürür.

Bu sınıfta bulunan sonraki yöntem (`getTitle()`) herhangi bir parametre almaz ve bu belirli hatanın hata kimliğini içeren bir dize değerini döndürür. XML paketinin doğrulaması sırasında gerçekleşen tam hatayı kolayca tanımlamanıza yardımcı olmak için bu değer kullanılır. Aşağıdaki alıntı, `getTitle()` yöntemini gösterir:

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

ApplicationError sınıfındaki son yöntem şudur: `toString()`. Bu yöntem, Error sınıfında tanımlanan işlevi geçersiz kılar, böylece hata mesajının sunumunu özelleştirebilirsiniz. Söz konusu yöntem, oluşan belirli hata sayısını ve mesajını tanımlayan bir dize döndürür.

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

FatalError sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

FatalError sınıfı, özel ApplicationError sınıfını genişletir ve üç yöntemi tanımlar: FatalError yapıcısı, `getTitle()` ve `toString()`. Birinci yöntem olan FatalError yapıcısı, tek bir tam sayı argümanını (`errorID`) alır ve ApplicationError sınıfında tanımlanan statik sabit değerlerini kullanarak hatanın önem derecesini ayarlar, ardından ApplicationError sınıfında `getMessageText()` yöntemini çağırarak belirli hatanın hata mesajını alır. FatalError yapıcısı şöyledir:

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

FatalError sınıfındaki sonraki yöntem olan `getTitle()`, ApplicationError sınıfında önceden tanımlanan `getTitle()` yöntemini geçersiz kılar ve kullanıcıya kritik bir hata oluştuğunu bildirmek için başlığın sonuna "-- FATAL" metnini ekler. `getTitle()` yöntemi şöyledir:

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

Bu sınıftaki son yöntem olan `toString()`, ApplicationError sınıfında tanımlanan `toString()` yöntemini geçersiz kılar. `toString()` yöntemi

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

WarningError sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

WarningError sınıfı, ApplicationError sınıfını genişletir ve FatalError sınıfıyla neredeyse aynıdır. Tek farkı, aşağıdaki kodda görüldüğü gibi, birkaç küçük dize değişikliğinin yanı sıra hata önem derecesini ApplicationError.FATAL olarak değil, ApplicationError.WARNING olarak ayarlamasıdır:

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

Bölüm 5: Normal ifadeler kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normal ifade, dizelerde eşleşen metinleri bulup işlemek için kullanılan bir deseni açıklar. Normal ifadeler dizelere benzer ancak bunlar desenleri ve yinelemeyi açıklamak için özel kodlar içerebilir. Örneğin, aşağıdaki normal ifade, A karakteriyle başlayan ve arkasından bir veya birkaç sıralı rakam gelen bir dizeyle eşleşir:

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

Aşağıdaki başlıklar normal ifadeler yapmaya yönelik temel sözdizimini açıklar. Ancak normal ifadeler birçok karmaşıklık ve küçük farklar içerebilir. Normal ifadelerle ilgili ayrıntılı kaynakları web'de ve kitapçılarda bulabilirsiniz. Farklı programlama ortamlarının normal ifadeleri farklı şekillerde uyguladığını unutmayın. ActionScript 3.0, normal ifadeleri, ECMAScript sürüm 3 dil belirtiminde (ECMA-262) tanımlandığı şekilde uygular.

Daha fazla Yardım konusu

RegExp

Normal ifadelerin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normal ifade, karakterlerin bir desenini açıklar. Normal ifadeler genellikle bir metin değerinin belirli bir desene uyduğunu doğrulamak (örn. kullanıcının girdiği telefon numarasının uygun sayıda basamak içerdiğini doğrulamak) veya metin değerinin belirli bir desenle eşleşen kısımlarını değiştirmek için kullanılır.

Normal ifadeler basit olabilir. Örneğin, belirli bir dizenin "ABC" ile eşleştiğini doğrulamak istediğinizi veya bir dizede geçen tüm "ABC" metinlerinin yerine başka bir metin getirmek istediğinizi varsayın. Bu durumda, sırayla A, B ve C harflerini içeren deseni tanımlayan şu normal ifadeyi kullanabilirsiniz:

```
/ABC/
```

Normal ifade değişmezinin eğik çizgi (`/`) karakteriyle ayrıldığını unutmayın.

Normal ifade desenleri aşağıdaki geçerli e-posta adresiyle eşleşme ifadesi gibi karmaşık ve bazen de şifreli görünebilir:

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

Normal ifadeleri en çok dizelerde desenleri aramak ve karakterleri değiştirmek için kullanırsınız. Bu durumlarda, bir normal ifade nesnesi oluşturur ve bu nesneyi birçok String sınıfı yöntemlerinden birinin parametresi olarak kullanırsınız. Aşağıdaki String sınıfı yöntemleri, parametre olarak normal ifadeleri alır: `match()`, `replace()`, `search()` ve `split()`. Bu yöntemlerle ilgili daha fazla bilgi için, bkz. “Dizelerdeki desenleri bulma ve alt dizeleri değiştirme” sayfa 16.

RegExp sınıfı şu yöntemleri içerir: `test()` ve `exec()`. Daha fazla bilgi için, bkz. “Dizelerle normal ifadeleri kullanma yöntemleri” sayfa 90.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi bu özellikle ilgili önemli terimler içerir:

Kaçış karakteri Ardından gelecek karakterin, değişmez karakter olarak değil, meta karakter olarak değerlendirilmesi gerektiğini belirten bir karakter. Normal ifade sözdiziminde, ters eğik çizgi (\) kaçış karakteridir, bu nedenle, ardından başka bir karakter gelen ters eğik çizgi, yalnızca bir karakter değil, özel bir koddur.

Bayrak Normal ifade deseninin nasıl kullanılması gerektiği hakkında bazı seçenekleri (örn. büyük harfli ve küçük harfli karakterler arasında ayrım yapılıp yapılmayacağı) tanımlayan bir karakter.

Meta karakter Desende harfi harfine karakteri temsil etmeyip normal ifade deseninde özel bir anlam içeren bir karakter.

Nicelik belirteci Bir desen bölümünün kaç defa yinelenmesi gerektiğini belirten bir karakter (veya birkaç karakter). Örneğin, ABD posta kodunun beş veya dokuz sayı içermesi gerektiğini belirtmek için bir nicelik belirteci kullanılır.

Normal ifade Diğer dizelerin bir karakter deseniyle eşleştiğini doğrulamak veya dizenin belirli bölümlerini değiştirmek için kullanılabilecek ilgili karakter desenini tanımlayan bir program deyimi.

Normal ifade sözdizimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu bölümde, ActionScript normal ifade sözdiziminin tüm öğeleri açıklanmaktadır. Gördüğünüz gibi, normal ifadeler birçok karmaşıklık ve küçük farklar içerebilir. Normal ifadelerle ilgili ayrıntılı kaynakları web'de ve kitapçılarda bulabilirsiniz. Farklı programlama ortamlarının normal ifadeleri farklı şekillerde uyguladığını unutmayın. ActionScript 3.0, normal ifadeleri, ECMAScript sürüm 3 dil belirtiminde (ECMA-262) tanımlandığı şekilde uygular.

Genellikle, basit bir karakterler dizisi yerine daha karmaşık desenlerle eşleşen normal ifadeleri kullanırsınız. Örneğin, aşağıdaki normal ifade, sırayla A, B, ve C harflerini ve bunların ardından herhangi bir rakamı içeren deseni tanımlar:

```
/ABC\d/
```

`\d` kodu, "herhangi bir rakamı" temsil eder. Eğik çizgi (`\`) karakteri, kaçış karakteri olarak adlandırılır ve ardından gelen karakterle (bu durumda d) birlikte, normal ifadede özel bir anlam içerir.

Aşağıdaki normal ifade, ardından herhangi bir rakam (yıldız işaretine dikkat edin) gelen ABC harfleri desenini tanımlar:

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

Yıldız karakteri (`*`) *meta karakterdir*. Meta karakter, normal ifadelerde özel anlam içeren bir karakterdir. Yıldız, *nicelik belirteci* adı verilen özel bir meta karakter türü olup bir karakterin veya karakterler grubunun yineleme sayısını belirtmek için kullanılır. Daha fazla bilgi için, bkz. "Nicelik belirteçleri" sayfa 82.

Normal ifade, desenine ek olarak, normal ifadenin nasıl eşleşeceğini belirten bayraklar da içerebilir. Örneğin, aşağıdaki normal ifade, normal ifadenin eşleşen dizelerde büyük/küçük duyarlılığını yoksaydığını belirten `i` bayrağını kullanır:

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

Daha fazla bilgi için bkz. "Bayraklar ve özellikler" sayfa 87.

Normal ifadeleri şu String sınıfı yöntemleriyle kullanabilirsiniz: `match()`, `replace()` ve `search()`. Bu yöntemlerle ilgili daha fazla bilgi için bkz. "Dizelerdeki desenleri bulma ve alt dizeleri değiştirme" sayfa 16.

Normal ifadenin bir örneğini oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normal ifade örneği oluşturmanın iki yolu vardır. Birinci yolda, normal ifadeleri ayırmak için eğik çizgi karakterleri (`/`) kullanılır; ikinci yolda ise `new` yapıcısı kullanılır. Örneğin, şu normal ifadeler birbirine eşdeğerdir:

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

Tırnak işaretlerinin bir dize değişmezini ayırdığı gibi, eğik çizgiler de normal ifade değişmezini ayırır. Normal ifadenin eğik çizgi içinde kalan kısmı *deseni tanımlar*. Normal ifade, son ayırıcı eğik çizgiden sonra *bayraklar* da içerebilir. Bu bayraklar, normal ifadenin bir parçası olarak değerlendirilir ancak bunlar normal ifadenin deseninden bağımsızdır.

`new` yapıcısını kullanırken, normal ifadeyi tanımlamak için iki dize kullanırsınız. Aşağıdaki örnekte gösterildiği gibi, birinci dize deseni tanımlar ve ikinci dize de bayrakları tanımlar:

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

Eğik çizgi ayırıcıları kullanılarak tanımlanan bir normal ifade *içine* eğik çizgi dahil ederken, eğik çizginin önüne ters eğik çizgi (`\`) kaçış karakteri getirmeniz gerekir. Örneğin, aşağıdaki normal ifade, `1/2` deseniyle eşleşir:

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

`new` yapıcısıyla tanımlanan bir normal ifade *içine* tırnak işaretleri dahil etmek için, tırnak işaretlerinden önce ters eğik çizgi (`\`) kaçış karakteri eklemeniz gerekir (herhangi bir String değişmezi tanımlanırken olduğu gibi). Örneğin, aşağıdaki normal ifadeler, `eat at "joe's"` deseniyle eşleşir:

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

Eğik çizgi ayırıcıları kullanılarak tanımlanan normal ifadelerde, tırnak işaretleriyle ters eğik çizgi kaçış karakterini kullanmayın. Aynı şekilde, `new` yapıcısıyla tanımlanan normal ifadelerde de eğik çizgilerle kaçış karakterini kullanmayın. Aşağıdaki normal ifadeler eşdeğerdir ve ikisi de `1/2 "joe's"` desenini tanımlar:

```
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"', '');
```

Ayrıca, `new` yapıcısıyla tanımlanan bir normal ifadede, `\d` (herhangi bir rakamla eşleşir) gibi ters eğik çizgi (`\`) karakteriyle başlayan bir meta sırası kullanmak için, ters eğik çizgiyi iki defa yazın:

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

`RegExp()` yapıcı yönteminin birinci parametresi bir dize olduğundan ve dize değişmezinde tek bir ters eğik çizgi karakteri olarak tanınması için iki defa ters eğik çizgi yazmanız gerektiğinden, bu durumda ters eğik çizgi karakterini iki defa yazmanız gerekir.

İlerleyen bölümlerde, normal ifade desenlerinin tanımlanmasına yönelik sözdizimi açıklanmaktadır.

Bayraklar hakkında daha fazla bilgi almak için, bkz. "Bayraklar ve özellikler" sayfa 87.

Karakterler, meta karakterler ve meta sıralar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

En basit normal ifade, aşağıdaki örnekte olduğu gibi, bir karakterler sırasıyla eşleşen ifadedir:

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

Ancak meta karakterler olarak bilinen şu karakterler, normal ifadelerde özel anlam içerir:

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

Örneğin, aşağıdaki normal ifade, sırayla A harfi, ardından B harfinin sıfır veya birkaç örneği (yıldız meta karakteri bu yinelemeyi belirtir) ve sonra da C harfi gelen bir desenle eşleşir:

```
/AB*C/
```

Bir normal ifade desenine, özel anlamı olmadan bir meta karakter dahil etmek için, ters eğik çizgi (`\`) kaçış karakterini kullanmanız gerekir. Örneğin, aşağıdaki normal ifade, sırayla A harfi, B harfi, yıldız işareti ve C harfini içeren bir desenle eşleşir:

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

Meta sıra da meta karakter gibi, normal ifadede özel anlam içerir. Meta sıra bir veya daha çok karakterden oluşur. İlerleyen bölümlerde, meta karakterlerin ve meta sıraların kullanılmasıyla ilgili ayrıntılar sağlanmaktadır.

Meta karakterler hakkında

Aşağıdaki tabloda, normal ifadelerde kullanabildiğiniz meta karakterler özetlenmektedir:

Meta karakter	Açıklama
^ (şapka)	Dizenin başında eşleşir. `m` (`multiline`) bayrağı ayarlanmış durumdayken, şapka karakteri satırın başlangıcıyla da eşleştirilir (bkz. "Bayraklar ve özellikler" sayfa 87). Karakter sınıfının başında kullanıldığında, şapka karakterinin dize başlangıcını değil, değilleme belirttiğini unutmayın. Daha fazla bilgi için, bkz. "Karakter sınıfları" sayfa 81.
$(dolar işareti)	Dizenin sonunda eşleşir. `m` (`multiline`) bayrağı ayarlıyken, `$` işareti, yeni satır (`\n`) karakterinden önceki konumla da eşleşir. Daha fazla bilgi için bkz. "Bayraklar ve özellikler" sayfa 87.
\ (ters eğik çizgi)	Özel karakterlerin meta karakter anlamından kaçınılmasını sağlar. Ayrıca, normal ifade değişmezinde eğik çizgi kullanmak istiyorsanız ters eğik çizgi karakterini kullanın, örn. `/1\/2/` (sırayla 1 karakteri, eğik çizgi karakteri ve 2 karakterinden oluşan bir desenle eşleşmek için).
. (nokta)	Herhangi bir tek karakterle eşleşir. Nokta yalnızca `s` (`dotall`) bayrağı ayarlandığında yeni satır karakteriyle (`\n`) eşleşir. Daha fazla bilgi için, bkz. "Bayraklar ve özellikler" sayfa 87.
* (yıldız)	Sıfır veya birkaç kere yinelenen önceki öğeyle eşleşir. Daha fazla bilgi için bkz. "Nicelik belirteçleri" sayfa 82.
+ (artı)	Bir veya birkaç kere yinelenen önceki öğeyle eşleşir. Daha fazla bilgi için bkz. "Nicelik belirteçleri" sayfa 82.
? (soru işareti)	Sıfır veya bir kere yinelenen önceki öğeyle eşleşir. Daha fazla bilgi için bkz. "Nicelik belirteçleri" sayfa 82.

Meta karakter	Açıklama
`(` ve `)`	Normal ifade içindeki grupları tanımlar. Şunlar için grupları kullanın: • `|` değiştiricisinin kapsamını sınırlamak için: `/(a|b|c)d/` • Nicelik belirtecinin kapsamını tanımlamak için: `/(walla.){1,2}/` • Geribaşvurularda. Örneğin, aşağıdaki normal ifadede bulunan `\1` öğesi, desenin birinci parantez grubunda eşleşen her şeyle eşleşir: • `/(\w*) is repeated: \1/` Daha fazla bilgi için bkz. "Gruplar" sayfa 84.
`[` ve `]`	Tek bir karakter için olası eşleşmeleri tanımlayacak şekilde bir karakter sınıfını tanımlar: `/[aeiou]/`, belirtilen karakterlerden herhangi biriyle eşleşir. Karakter sınıfları içinde, karakterlerin aralığını belirlemek için kısa çizgi (-) kullanın: `/[A-Z0-9]/`, büyük harfli A - Z veya 0 - 9 ile eşleşir. Karakter sınıfları içinde,] ve - karakterlerinin özel anlamından kaçınmak için ters eğik çizgi ekleyin. `/[+\-]\d+/`, bir veya daha çok rakamdan önce gelen `+` ya da `-` öğesiyle eşleşir. Karakter sınıfları için, normalde meta karakter olan diğer karakterler, ters eğik çizgiye gerek kalmadan normal karakterler (meta karakter olmayan) olarak değerlendirilir: `/[$]/£`, `$` veya `£` öğesiyle eşleşir. Daha fazla bilgi için bkz. "Karakter sınıfları" sayfa 81.
`|` *(çubuk)*	Sol taraftaki bölümle veya sağ taraftaki bölümle eşleşmek üzere değişim için kullanılır: `/abc|xyz/`, `abc` veya `xyz` öğesiyle eşleşir.

Meta sıralar hakkında

Meta sıralar, normal bir ifade deseninde özel anlam içeren karakterler sırasıdır. Aşağıdaki tabloda, bu meta sıralar açıklanmaktadır:

Meta sıra	Açıklama

`{n}`
`{n,}`
ve
`{n,n}`
	Önceki öğe için sayısal nicelik belirtecini veya nicelik belirteci aralığını belirtir:
	`/A{27}/`, `27` kere yinelenen `A` karakteriyle eşleşir.
	`/A{3,}/`, en az `3` kere yinelenen `A` karakteriyle eşleşir.
	`/A{3,5}/`, `3` ile `5` defa yinelenen `A` karakteriyle eşleşir.
	Daha fazla bilgi için bkz. "Nicelik belirteçleri" sayfa 82.

`\b`	Sözcük karakteri ile sözcük olmayan bir karakter arasındaki konumla eşleşir. Dizedeki birinci ve sonuncu karakter bir sözcükse, ayrıca dizenin başıyla veya sonuyla da eşleşir.
`\B`	İki sözcük karakteri arasındaki konumda eşleşir. Ayrıca sözcük olmayan iki karakter arasındaki konumla da eşleşir.
`\d`	Ondalık bir basamakla eşleşir.
`\D`	Rakam dışında bir karakterle eşleşir.
`\f`	Sonraki sayfaya geçme karakteriyle eşleşir.
`\n`	Yeni satır karakteriyle eşleşir.

Meta sıra	Açıklama
`\r`	Döndürme karakteriyle eşleşir.
`\s`	Herhangi bir boşluk karakteriyle (boşluk, sekme, yeni satır veya döndürme karakteri) eşleşir.
`\S`	Boşluk karakteri dışındaki herhangi bir karakterle eşleşir.
`\t`	Sekme karakteriyle eşleşir.
`\u`*nnnn*	*nnnn* onaltılık sayısıyla karakter kodu belirtilmiş Unicode karakteriyle eşleşir. Örneğin, `\u263a`, gülen yüz karakteridir.
`\v`	Dikey besleme karakteriyle eşleşir.
`\w`	Bir sözcük karakteriyle (`AZ–`, `az–`, `0-9` veya `_`) eşleşir. `\w` öğesinin, `é` , `ñ` veya `ç` gibi İngilizce olmayan karakterlerle eşleşmediğini unutmayın.
`\W`	Sözcük karakteri dışında herhangi bir karakterle eşleşir.
`\\x`*nn*	*nn* onaltılık sayısıyla tanımlandığı şekilde ASCII değeri belirtilmiş karakterle eşleşir.

Karakter sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normal ifadedeki tek bir konumla eşleşecek karakterlerin bir listesini belirtmek için karakter sınıflarını kullanırsınız. Karakter sınıflarını köşeli ayraçlarla (`[` ve `]`) tanımlarsınız. Örneğin, aşağıdaki normal ifade, `bag`, `beg`, `big`, `bog` veya `bug` ile eşleşen bir karakter sınıfını tanımlar:

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

Karakter sınıflarında kaçış sıraları

Normalde bir normal ifadede özel anlamlar içeren meta karakterlerin ve meta sıraların çoğu, karakter sınıfında bu aynı anlamları *içermez*. Örneğin, normal bir ifadede, yineleme için yıldız işareti kullanılır, ancak karakter sınıfında yıldız görüntülendiğinde bu kural geçerli değildir. Aşağıdaki karakter sınıfı, listelenen diğer karakterlerle birlikte harfi harfine yıldız ile eşleşir:

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

Ancak, aşağıdaki tabloda listelenen üç karakter, karakter sınıflarında da özel anlamlarını içerecek şekilde meta karakterler olarak çalışır:

Meta karakter	Karakter sınıflarındaki anlamı
`]`	Karakter sınıfının sonunu tanımlar.
`-`	Karakter aralığını tanımlar. (İlerleyen bölümlerdeki "Karakter sınıflarında karakter aralıkları" konusuna bakın.)
`\`	Meta sıraları tanımlar ve meta karakterlerin özel anlamını geri alır.

Bu karakterlerden herhangi birinin değişmez karakter olarak (özel meta karakter anlamı olmadan) tanınması için, karakterin önüne ters eğik çizgi kaçış karakterini getirmeniz gerekir. Örneğin, aşağıdaki normal ifade, dört sembolden herhangi biriyle (`$`, `\`, `]` veya `-`) eşleşen bir karakter sınıfını içerir:

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

Özel anlamlarını koruyan meta karakterlere ek olarak, aşağıdaki meta sıralar da karakter sınıfları içinde meta sıralar olarak çalışır:

Meta sıra	Karakter sınıflarındaki anlamı
`\n`	Yeni satır karakteriyle eşleşir.
`\r`	Döndürme karakteriyle eşleşir.
`\t`	Sekme karakteriyle eşleşir.
`\u`***nnnn***	Unicode kod punto değeri (*nnnn* onaltılık sayısıyla tanımlandığı şekilde) belirtilmiş karakterle eşleşir.
`\\x`***nn***	ASCII değeri (*nn* onaltılık sayısıyla tanımlandığı şekilde) belirtilmiş karakterle eşleşir.

Diğer normal ifade meta sıraları ve meta karakterleri, karakter sınıfı içinde normal karakterler olarak değerlendirilir.

Karakter sınıflarında karakter aralıkları

`A-Z`, `a-z` veya `0-9` gibi bir karakter aralığı belirtmek için kısa çizgi kullanın. Bu karakterlerin, karakter kümesi içinde geçerli bir aralık oluşturması gerekir. Örneğin, aşağıdaki karakter sınıfı, `a-z` aralığındaki karakterlerden herhangi biriyle veya bir rakamla eşleşir:

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

ASCII değerine göre bir aralık belirtmek için, `\\x`*nn* ASCII karakter kodunu da kullanabilirsiniz. Örneğin, aşağıdaki karakter sınıfı, genişletilmiş ASCII karakterleri kümesinden herhangi bir karakterle (örn. `é` ve `ê`) eşleşir:

```
\\x
```

Değilleme uygulanmış karakter sınıfları

Karakter sınıfının başında bir başlık (`^`) karakteri kullandığınızda bu, sınıfa değilleme işlemi uygular—listelenmeyen tüm karakterler bir eşleşme olarak değerlendirilir. Aşağıdaki karakter sınıfı, küçük harfler (`az–`) veya rakamlar *dışında* herhangi bir karakterle eşleşir:

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

Değillemeyi belirtmek için karakter sınıfının *başına* şapka (`^`) karakterini yazmanız gerekir. Aksi takdirde, yalnızca karakter sınıfındaki karakterlere şapka karakteri eklemiş olursunuz. Örneğin, aşağıdaki karakter sınıfı, şapka da dahil olmak üzere, sembol karakterlerinden herhangi biriyle eşleşir:

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

Nicelik belirteçleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki gibi, desenlerde karakterlerin veya sıraların yinelemesini belirtmek için nicelik belirteçlerini kullanırsınız:

Nicelik belirteci meta karakteri	Açıklama
`*` (yıldız)	Sıfır veya birkaç kere yinelenen önceki öğeyle eşleşir.

Nicelik belirteci meta karakteri	Açıklama
+ (artı)	Bir veya birkaç kere yinelenen önceki öğeyle eşleşir.
? (soru işareti)	Sıfır veya bir kere yinelenen önceki öğeyle eşleşir.
{*n*} {*n*,} ve {*n*,*n*}	Önceki öğe için sayısal nicelik belirtecini veya nicelik belirteci aralığını belirtir: /A{27}/, 27 kere yinelenen A karakteriyle eşleşir. /A{3,}/, en az 3 kere yinelenen A karakteriyle eşleşir. /A{3,5}/, 3 ile 5 defa yinelenen A karakteriyle eşleşir.

Tek bir karaktere, karakter sınıfına veya bir gruba nicelik belirteci uygulayabilirsiniz:

- /a+/, en az bir kere yinelenen a karakteriyle eşleşir.
- /\d+/, bir veya daha fazla rakamla eşleşir.
- /[abc]+/, her biri a, b veya c olacak şekilde bir veya birkaç karakterin yinelemesiyle eşleşir.
- /(very,)*/ , ardından sıfır veya birkaç kere virgül ve boşluk gelen very sözcüğüyle eşleşir.

Nicelik belirteci uygulanmış parantez grupları içinde nicelik belirteçleri kullanabilirsiniz. Örneğin, aşağıdaki nitelik belirteci, word ve word-word-word öğeleriyle eşleşir:

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

Varsayılan olarak, normal ifadeler *doyumsuz eşleşme olarak bilinen bir eşleşme gerçekleştirir.* Normal ifadedeki herhangi bir alt desen (örn. .*) normal ifadenin diğer bölümüne ilerlemeden önce dizede olabildiğince çok karakterle eşleşmeye çalışır. Örneğin, şu normal ifadeyi ve dizeyi göz önünde bulundurun:

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

Normal ifade, dizenin tamamıyla eşleşir:

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

Ancak yalnızca tek bir ... grubuyla eşleşme yapmak istediğinizi varsayın. Bunu şu şekilde yapabilirsiniz:

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

Herhangi bir nicelik belirtecinden sonra soru işareti (?) ekleyerek, bu nicelik belirtecini *tembel nicelik belirteci* olarak değiştirin. Örneğin, tembel *? nicelik belirtecini kullanan şu normal ifade, sırayla ve olabildiğince az sayıda karakter (tembel) ve sonra içeren bir dizeyle eşleşir:

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

Nicelik belirteçleriyle ilgili olarak şu noktaları unutmayın:

- {0} ve {0,0} nicelik belirteçleri, bir öğeyi eşleşmeden hariç tutmaz.
- /abc+*/ öğesinde olduğu gibi, birden çok nicelik belirtecini birleştirmeyin.
- Nokta (.), ardından * nicelik belirteci gelse de, s (dotall) bayrağı ayarlanmadan satırlara yayılmaz. Örneğin, şu kodu göz önünde bulundurun:

```
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>
```

Daha fazla bilgi için bkz. "Bayraklar ve özellikler" sayfa 87.

Değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normal ifade motorunun eşleşme alternatiflerini göz önünde bulundurmasını sağlamak için, normal ifadede `|` (boru) karakterini kullanın. Örneğin, aşağıdaki normal ifade, `cat, dog, pig, rat` sözcüklerinden herhangi biriyle eşleşir:

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

`|` değiştiricisinin kapsamını kısıtlamak üzere gruplar tanımlamak için parantezler kullanabilirsiniz. Aşağıdaki normal ifade, ardından `nap` veya `nip` gelen `cat` öğesiyle eşleşir:

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

Daha fazla bilgi için, bkz. "Gruplar" sayfa 84.

Biri `|` değiştiricisini ve diğeri bir karakter sınıfını kullanan (`[` ve `]` ile tanımlı) şu iki normal ifade eşdeğerdir:

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

Daha fazla bilgi için bkz. "Karakter sınıfları" sayfa 81.

Gruplar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki gibi, parantezleri kullanarak normal ifadede bir grup belirtebilirsiniz:

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

Grup, bir desenin alt bölümüdür. Şunları yapmak için grupları kullanabilirsiniz:

- Birden çok karaktere nicelik belirteci uygulama.
- Uygulanacak alt desenleri, değiştirme yoluyla (`|` karakterini kullanarak) ayırma.
- Alt dize eşleşmelerini yakalama (örneğin, önceden eşleşen bir grupla eşleşme sağlamak için normal ifadede `\1` öğesini kullanarak veya String sınıfının `replace()` yönteminde aynı şekilde `$1` öğesini kullanarak).

İlerleyen bölümlerde, grupların bu kullanımlarıyla ilgili ayrıntılar sağlanmaktadır.

Nicelik belirteçleriyle grupları kullanma

Bir grup kullanmazsanız, aşağıda gösterildiği gibi, nicelik belirteci kendisinden önceki karakter veya karakter sınıfı için geçerli olur:

```
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
```

Ancak, birden çok karakter veya karakter sınıfına nicelik belirteci uygulamak için bir grup kullanabilirsiniz:

```
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
```

Nicelik belirteçleri hakkında daha fazla bilgi için, bkz. "Nicelik belirteçleri" sayfa 82.

Değiştirici (|) karakteriyle grupları kullanma

Aşağıdaki gibi, değiştirici (`|`) karakteri uygulamak istediğiniz karakterler grubunu tanımlamak için grupları kullanabilirsiniz:

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

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

Alt dize eşleşmelerini yakalamak için grupları kullanma

Bir desende standart bir parantez grubu tanımladığınızda, daha sonra normal ifadede bu gruba başvurabilirsiniz. Bu, *geribaşvuru* olarak bilinir ve bu tür gruplar da *yakalama grupları* olarak adlandırılır. Örneğin, aşağıdaki normal ifadede `\1` sırası, yakalama parantez grubunun eşleştiği tüm alt dizelerle eşleşir:

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

`\1`, `\2`, ... , `\99` yazarak normal ifadede bu geribaşvuruların en fazla 99 tanesini belirtebilirsiniz.

Aynı şekilde, String sınıfının `replace()` yönteminde, yerini alan dizeye yakalanan grup alt dizesi eşleşmelerini eklemek için `$1$99` öğesini kullanabilirsiniz:

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

Ayrıca, yakalama gruplarını kullanırsanız, RegExp sınıfının `exec()` yöntemi ve String sınıfının `match()` yöntemi, yakalama gruplarıyla eşleşen alt dizeleri döndürür:

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

Yakalama yapmayan grupları ve ileri yönlü grupları kullanma

Yakalama yapmayan grup, yalnızca gruplandırma için kullanılır; "toplanmaz" ve numaralandırılmış geribaşvurularla eşleşmez. Yakalama yapmayan grupları tanımlamak için şu şekilde `(?:` ve `)` öğelerini kullanın:

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

Örneğin, bir yakalama grubuna ve yakalama yapmayan gruba `(com|org)` öğesinin koyulması arasındaki farka dikkat edin (`exec()` yöntemi, tam eşleşmeden sonra yakalama gruplarını listeler):

```
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
```

Yakalama yapmayan grubun özel bir türü *ileri yönlü grup* olup bunun da iki türü vardır: *pozitif ileri yönlü grup* ve *negatif ileri yönlü grup*.

Gruptaki alt desenin konumda eşleşmesi gerektiğini belirten bir pozitif ileri yönlü grup tanımlamak için `(?=` ve `)` öğelerini kullanın. Ancak, dizenin pozitif ileri yönlü grupla eşleşen kısmı, normal ifadede kalan diğer desenlerle eşleşebilir. Örneğin, aşağıdaki kodda `(?=e)` bir pozitif ileri yönlü grup olduğundan, bunun eşleştiği `e` karakteri, normal ifadenin sonraki kısmıyla da (bu durumda yakalama grubu, `\w*)`) eşleşebilir:

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

Gruptaki alt desenin konumda *eşleşmemesi* gerektiğini belirten bir negatif ileri yönlü grup tanımlamak için, `(?!` ve `)` öğelerini kullanın. Örneğin:

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

Adlandırılmış grupları kullanma

Adlandırılmış grup, normal ifadede, adlandırılmış bir tanımlayıcı verilmiş grup türüdür. Adlandırılmış grubu tanımlamak için, `(?P<name>` ve `)` öğelerini kullanın. Örneğin, aşağıdaki normal ifade, `digits` adında tanımlayıcıyla adlandırılmış bir grup içerir:

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

`exec()` yöntemini kullandığınızda, eşleşen ada sahip bir grup `result` dizisinin bir özelliği olarak eklenir:

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

Aşağıda, `name` ve `dom` tanımlayıcılarıyla, adlandırılmış iki grup kullanan başka bir örnek verilmiştir:

```
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
```

Not: *Adlandırılmış gruplar, ECMAScript dil belirtiminin parçası değildir. Bunlar ActionScript 3.0'a eklenen bir özelliktir.*

Bayraklar ve özellikler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki tabloda, normal ifadeler için ayarlayabildiğiniz beş bayrak listelenmektedir. Normal ifade nesnesinin bir özelliği olarak her bayrağa erişilebilir.

Bayrak	Özellik	Açıklama
`g`	`global`	Birden çok eşleşme yapar.
`i`	`ignoreCase`	Büyük/küçük harf duyarlı eşleşme. `A`—`Z` ve `a`—`z` karakterleri için geçerlidir ancak `É` ve `é` gibi genişletilmiş karakterler için geçerli değildir.
`m`	`multiline`	Bu bayrak ayarlıyken, `$` ve `^` öğeleri sırayla bir satırın başıyla ve bir satırın sonuyla eşleşebilir.
`s`	`dotall`	Bu bayrak ayarlıyken, `.` (nokta), yeni satır karakteriyle (`\n`) eşleşebilir.
`x`	`extended`	Genişletilmiş normal ifadelere izin verir. Desenin parçası olarak yoksayılan boşlukları normal ifadeye yazabilirsiniz. Bu, normal ifade kodunuzu daha okunaklı yazmanıza olanak sağlar.

Bu özelliklerin salt okunur olduğunu unutmayın. Aşağıdaki gibi, normal ifade değişkeni ayarladığınızda, bayrakları (`g`, `i`, `m`, `s`, `x`) ayarlayabilirsiniz:

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

Ancak, adlandırılmış özellikleri doğrudan ayarlayamazsınız. Örneğin, aşağıdaki kod bir hataya yol açar:

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

Varsayılan olarak, bayrakları normal ifade bildiriminde belirtmemeniz durumunda, bayraklar ayarlanmaz ve karşılık gelen özellikler de `false` değerine ayarlanır.

Ayrıca, normal ifadenin iki başka özelliği daha vardır:

- `lastIndex` özelliği, normal ifadenin `exec()` veya `test()` yöntemine yapılan sonraki çağrı için dizede kullanılacak dizin konumunu belirtir.
- `source` özelliği, normal ifadenin desen bölümünü tanımlayan dizeyi belirtir.

g (global) bayrağı

`g` (`global`) bayrağı dahil *edilmediğinde*, normal ifade birden çok eşleşme yapmaz. Örneğin, normal ifadeye `g` bayrağı dahil edilmediğinde, `String.match()` yöntemi yalnızca bir eşleşen alt dize döndürür:

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

`g` bayrağı ayarlandığında, `Sting.match()` yöntemi aşağıdaki gibi birden çok eşleşme döndürür:

```
var str:String = "she sells seashells by the seashore.";
var pattern:RegExp = /sh\w*/g;
// The same pattern, but this time the g flag IS set.
trace(str.match(pattern)); // output: she,shells,shore
```

i (ignoreCase) bayrağı

Varsayılan olarak, normal ifade eşleşmeleri büyük/küçük harf duyarlıdır. `i` (`ignoreCase`) bayrağını ayarladığınızda, büyük/küçük harf duyarlılığı yoksayılır. Örneğin, normal ifadedeki küçük harfli `s` öğesi, dizenin birinci karakteri olan büyük harfli `S` öğesiyle eşleşmez:

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

Ancak `i` bayrağı ayarlıyken, normal ifade, baş harf olan `S` öğesiyle eşleşir:

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

`i` bayrağı yalnızca `A–Z` ve `a–z` karakterleri için büyük/küçük harf duyarlılığını yok sayar ancak `É` ve `é` gibi genişletilmiş karakterler için büyük/küçük duyarlılığını uygulamaya devam eder.

m (multiline) bayrağı

`m` (`multiline`) bayrağı ayarlanmazsa, `^` öğesi dizenin başıyla ve `$` öğesi de dizenin sonuyla eşleşir. `m` bayrağı ayarlanırsa, bu karakterler sırayla satırın başıyla ve satırın sonuyla eşleşir. Bir yeni satır karakteri içeren şu dizeyi göz önünde bulundurun:

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

Normal ifadede `g` (`global`) bayrağı ayarlansa da, `^` için yalnızca bir eşleşme (dizenin başı) bulunduğundan, `match()` yöntemi yalnızca bir alt dizeyle eşleşir. Çıktı şudur:

```
Test
```

Burada, aynı kod `m` bayrağı ayarlanmış şekilde verilmiştir:

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

Bu defa, çıktı her iki satırın başındaki sözcükleri içerir:

```
Test,Multiline
```

Yalnızca `\n` karakterinin satır sonunu işaret ettiğini unutmayın. Aşağıdaki karakterler satır sonunu işaret etmez:

- Döndürme (`\r`) karakteri
- Unicode satır ayırıcı (`\u2028`) karakteri

- Unicode paragraf ayırıcı (`\u2029`) karakteri

s (dotall) bayrağı

`s` (`dotall` veya "dot all") bayrağı ayarlanmazsa, normal ifadedeki bir nokta (`.`) yeni satır karakteriyle (`\n`) eşleşmez. Bu nedenle de aşağıdaki örnekte bir eşleşme yoktur:

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

Ancak, `s` bayrağı ayarlanırsa, nokta yeni satır karakteriyle eşleşir:

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

Bu durumda eşleşme, yeni satır karakteri de dahil olacak şekilde `<p>` etiketleri içindeki alt dizenin tamamı olur:

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

x (extended) bayrağı

Özellikle de çok fazla meta sembol ve meta sıralar içeriyorsa, normal ifadelerin okunması zor olabilir. Örneğin:

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

Normal ifadede `x` (`extended`) bayrağını kullandığınızda, desene yazdığınız tüm boşluklar yoksayılır. Örneğin, aşağıdaki normal ifade bir önceki örnekle aynıdır:

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

`x` bayrağı ayarlanmışsa ve boşluk karakteriyle eşleşme olmasını istiyorsanız, boşluğun önüne ters eğik çizgi getirin. Örneğin, şu iki normal ifade birbirine eşdeğerdir:

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

lastIndex özelliği

`lastIndex` özelliği, dizede, bir sonraki aramanın başlayacağı dizin konumunu belirtir. Bu özellik, `g` bayrağı `true` değerine ayarlanmış bir normal ifadede çağrılan `exec()` ve `test()` yöntemlerini etkiler. Örneğin, şu kodu göz önünde bulundurun:

```
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);
}
```

`lastIndex` özelliği varsayılan olarak 0 değerine ayarlanır (aramayı dizenin başından başlatmak için). Her eşleşmeden sonra, bu özellik eşleşmeden sonraki dizin konumuna ayarlanır. Bu nedenle de önceki kodun çıktısı şöyledir:

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

`global` bayrağı `false` değerine ayarlanırsa, `exec()` ve `test()` yöntemleri `lastIndex` özelliğini kullanmaz veya ayarlamaz.

String sınıfının `match()`, `replace()` ve `search()` yöntemlerinin tümü, yöntem çağrısında kullanılan normal ifadenin `lastIndex` özelliğinin ayarına bakılmaksızın, tüm aramaları dizenin başından başlatır. (Ancak, `match()` yöntemi `lastIndex` öğesini 0 değerine ayarlar.)

`lastIndex` özelliğini normal ifade eşlemesi için dizedeki başlangıç konumunu belirlemek üzere ayarlayabilirsiniz.

source özelliği

`source` özelliği, normal ifadenin desen bölümünü tanımlayan dizeyi belirtir. Örneğin:

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

Dizelerle normal ifadeleri kullanma yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RegExp sınıfı iki yöntem içerir: `exec()` ve `test()`.

String sınıfı, RegExp sınıfının `exec()` ve `test()` yöntemlerine ek olarak, dizelerde normal ifadelerle eşleşme yapmanıza olanak sağlayan şu yöntemleri de içerir: `match()`, `replace()`, `search()` ve `splice()`.

test() yöntemi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RegExp sınıfının `test()` yöntemi, aşağıdaki örnekte gösterildiği gibi, sağlanan dizede normal ifadeyle eşleşme olup olmadığını kontrol eder:

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

exec() yöntemi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RegExp sınıfının `exec()` yöntemi, sağlanan dizede normal ifadeyle eşleşme olup olmadığını kontrol eder ve şunlarla bir dizi döndürür:

- Eşleşen alt dize
- Normal ifadede herhangi bir parantez grubuyla alt dize eşleşmeleri

Dizi ayrıca, alt dize eşleşmesinin başının dizin konumunu belirten bir `index` özelliğini de içerir.

Örneğin, şu kodu göz önünde bulundurun:

```
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
```

Normal ifade için `g` (`global`) bayrağı ayarlandığında, birden çok alt dizeyle eşleşme yapmak için birçok defa `exec()` yöntemini kullanın:

```
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
```

RegExp parametrelerini kullanan String yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki String sınıfı yöntemleri, parametre olarak normal ifadeleri alır: `match()`, `replace()`, `search()` ve `split()`. Bu yöntemlerle ilgili daha fazla bilgi için, bkz. "Dizelerdeki desenleri bulma ve alt dizeleri değiştirme" sayfa 16.

Normal ifade örneği: Wiki ayrıştırıcısı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu basit Wiki metin dönüştürme örneğinde, birçok normal ifade kullanımı gösterilmektedir:

- Kaynak Wiki deseniyle eşleşen metin satırlarını uygun HTML çıktı dizelerine dönüştürme.
- URL desenlerini HTML `<a>` köprüsü etiketlerine dönüştürmek için normal ifade kullanma.
- ABD doları dizelerini (örn. `"$9.95"`) euro dizelerine (örn. `"8.24 €"`) dönüştürmek için normal ifade kullanma.

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. WikiEditor uygulama dosyalarını Samples/WikiEditor klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
WikiEditor.mxml veya WikiEditor.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/regExpExamples/WikiParser.as	Wiki girdi metni desenlerini eşdeğer HTML çıktısına dönüştürmek için normal ifadeleri kullanan yöntemleri içeren bir sınıf.
com/example/programmingas3/regExpExamples/URLParser.as	URL dizelerini HTML `<a>` köprüsü etiketlerine dönüştürmek için normal ifadeleri kullanan yöntemleri içeren bir sınıf.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	ABD doları dizelerini euro dizelerine dönüştürmek için normal ifadeleri kullanan yöntemleri içeren bir sınıf.

WikiParser sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

WikiParser sınıfı, Wiki girdi metnini eşdeğer HTML çıktısına dönüştüren yöntemleri içerir. Bu çok güçlü bir Wiki dönüştürme uygulaması değildir ancak desen eşleştirme ve dize dönüştürmesi için normal ifadelerin bazı kullanımlarını gösterir.

`setWikiData()` yöntemiyle birlikte yapıcı işlevi yalnızca aşağıdaki gibi Wiki girdi metninin örnek bir dizesini başlatır:

```
public function WikiParser()
{
    wikiData = setWikiData();
}
```

Kullanıcı örnek uygulamada Test Et düğmesini tıklattığında, uygulama, WikiParser nesnesinin `parseWikiString()` yöntemini çağırır. Bu yöntem, başka birçok yöntemi çağırır ve bu yöntemler de daha sonra sonuçta elde edilen HTML dizesini oluşturur.

```
public function parseWikiString(wikiString:String):String
{
    var result:String = parseBold(wikiString);
    result = parseItalic(result);
    result = linesToParagraphs(result);
    result = parseBullets(result);
    return result;
}
```

Çağrılan yöntemlerin her biri (`parseBold()`, `parseItalic()`, `linesToParagraphs()` ve `parseBullets()`), girdi Wiki metnini HTML olarak formatlanmış metne dönüştürmek üzere bir normal ifade tarafından tanımlandığı şekilde, eşleşen desenleri değiştirmek için dizenin `replace()` yöntemini kullanır.

Kalın ve italik desenleri dönüştürme

`parseBold()` yöntemi bir Wiki kalın metin desenini (örn. `'''foo'''`) arar ve bu metin desenini şu şekilde HTML eşdeğerine (örn. `<b>foo</b>`) dönüştürür:

```
private function parseBold(input:String):String
{
    var pattern:RegExp = /'''(.*?)'''/g;
    return input.replace(pattern, "<b>$1</b>");
}
```

Normal ifadenin `(.?*)` bölümünün, iki tanımlayıcı `'''` deseni arasındaki herhangi sayıda karakterle (`*`) eşleştiğini unutmayı. Şu `?` nicelik belirteci eşleşmeyi tembel durumuna getirir, bu nedenle, `'''aaa''' bbb '''ccc'''` gibi bir dize için eşleşen birinci dize `'''aaa'''` olup dizenin tamamı olmaz (`'''` deseniyle başlayıp sona eren).

Normal ifadedeki parantezler bir yakalama grubunu ifade eder ve `replace()` yöntemi, yerini alan dizede `$1` kodunu kullanarak bu gruba başvurur. Normal ifadedeki `g` (`global`) bayrağı, `replace()` yönteminin dizedeki tüm eşleşmeleri (yalnızca birinciyi değil) değiştirmesini sağlar.

`parseItalic()` yöntemi `parseBold()` yöntemine benzer şekilde çalışır, tek farkı, italik metin sınırlayıcısı olarak iki kesme işaretini (`''`) (üç değil) kontrol etmesidir:

```
private function parseItalic(input:String):String
{
    var pattern:RegExp = /''(.*?)''/g;
    return input.replace(pattern, "<i>$1</i>");
}
```

Madde imi desenlerini dönüştürme

Aşağıdaki örnekte gösterildiği gibi, `parseBullet()` yöntemi, bir Wiki madde imi satır desenini (örn. `* foo`) arar ve bunu HTML eşdeğerine (örn. `<li>foo</li>`) dönüştürür:

```
private function parseBullets(input:String):String
{
    var pattern:RegExp = /^\*(.*)/gm;
    return input.replace(pattern, "<li>$1</li>");
}
```

Normal ifadenin başındaki `^` sembolü, satırın başıyla eşleşir. Normal ifadedeki `m` (`multiline`) bayrağı, normal ifadenin `^` sembolünü yalnızca dizenin başıyla değil, satırın başıyla da eşleştirmesini sağlar.

`\*` deseni, bir yıldız karakteriyle eşleşir. (`*` nicelik belirteci yerine değişmez yıldızı işaret etmek için ters eğik çizgi kullanılmıştır.)

Normal ifadedeki parantezler bir yakalama grubunu tanımlar ve `replace()` yöntemi, yerini alan dizede `$1` kodunu kullanarak bu gruba başvurur. Normal ifadedeki `g` (`global`) bayrağı, `replace()` yönteminin dizedeki tüm eşleşmeleri (yalnızca birinciyi değil) değiştirmesini sağlar.

Paragraf Wiki desenlerini dönüştürme

`linesToParagraphs()` yöntemi, girdi Wiki dizesindeki her satırı bir HTML `<p>` paragraf etiketine dönüştürür. Yöntemdeki bu satırlar, girdi Wiki dizesinden boş satırları çıkarır:

```
var pattern:RegExp = /^$/gm;
var result:String = input.replace(pattern, "");
```

Normal ifadenin, satırın başını ve sonunu eşleştirdiği `^` ve `$` sembolleri. Normal ifadedeki `m` (`multiline`) bayrağı, normal ifadenin ^ sembolünü yalnızca dizenin başıyla değil satırın başıyla da eşleştirmesini sağlar.

`replace()` yöntemi, eşleşen tüm alt dizelerin yerine (boş satırlar) boş bir dize (`""`) getirir. Normal ifadedeki `g` (`global`) bayrağı, `replace()` yönteminin dizedeki tüm eşleşmeleri (yalnızca birinciyi değil) değiştirmesini sağlar.

URL'leri HTML <a> etiketlerine dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı, `urlToATag` onay kutusunu seçmişse, örnek uygulamada Test Et düğmesini tıklattığında, uygulama, URL dizelerini girdi Wiki dizesinden HTML `<a>` etiketlerine dönüştürmek için `URLParser.urlToATag()` statik yöntemini çağırır

```
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>");
```

`RegExp()` yapıcı işlevi, çok sayıda bileşenden tek bir normal ifade (`urlPattern`) oluşturmak için kullanılır. Bu bileşenlerin her biri, normal ifade deseninin parçasını tanımlayan dizelerdir.

`protocol` dizesi tarafından tanımlanan normal ifade deseninin birinci bölümü, `http://` veya `ftp://` olmak üzere bir URL protokolünü tanımlar. Parantezler, `?` sembolüyle belirtilen, yakalama yapmayan bir grubu tanımlar. Başka bir deyişle, parantezler yalnızca `|` değiştirme desenine yönelik bir grubu tanımlamak için kullanılır; bu grup, `replace()` yönteminin yerini alan dizesinde geribaşvuru kodlarıyla (`$1`, `$2`, `$3`) eşleşmez.

Normal ifadeyi oluşturan diğer bileşenlerin her biri, daha sonra `replace()` yönteminin yerini alan dizesindeki geribaşvuru kodlarında (`$1`, `$2`, `$3`) kullanılan yakalama gruplarını (desende parantezlerle belirtilir) kullanır.

Desenin, `urlPart` dizesi tarafından tanımlanan bölümü, şu karakterlerden *en az* biriyle eşleşir: `a-z`, `0-9`, `_` veya `-`. `+` nicelik belirteci, en az bir karakterin eşleştiğini belirtir. Şu `\.` öğesi, zorunlu bir nokta (`.`) karakterini belirtir. Ve geri kalan kısım da bu karakterlerden en az birini içeren bir dizeyle eşleşir: `a-z`, `0-9`, `_` veya `-`.

Desenin, `optionalUrlPart` dizesiyle tanımlanan bölümü, şunlardan *hiçbiriyle veya bir ya da birkaç tanesiyle* eşleşir: nokta (`.`) karakteri ve onu takip eden herhangi bir sayıda alfasayısal karakter (`_` ve `-` dahil). `*` nicelik belirteci, bu karakterlerden hiçbirinin eşleşmediğini veya bir ya da birkaçının eşleştiğini belirtir.

`replace()` yöntemine yapılan çağrı, normal ifadeyi kullanır ve geribaşvuruları kullanarak, yerini alan HTML dizesini oluşturur.

Daha sonra `urlToATag()` yöntemi, e-posta desenlerini, HTML `<a>` köprü dizeleriyle değiştirme tekniğine benzer teknikler kullanan `emailToATag()` yöntemini çağırır. Bu örnek dosyada HTTP, FTP ve e-posta URL'leriyle eşleşme yapmak için kullanılan normal ifadeler, örneklendirme amacıyla nispeten basit tutulmuştur; bu URL'lerin daha doğru şekilde eşleştirilmesi için daha karmaşık normal ifadeler vardır.

ABD doları dizelerini euro dizelerine dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı `dollarToEuro` onay kutusunu seçmişse, örnek uygulamadaki Test Et düğmesini tıklattığında, aşağıdaki gibi uygulama, ABD doları dizelerini (örn. `"$9.95"`) euro dizelerine (örn. `"8.24 €"`) dönüştürmek için, `CurrencyConverter.usdToEuro()` statik yöntemini çağırır:

```
var usdPrice:RegExp = /\$([\d,]+.\d+)+/g;
return input.replace(usdPrice, usdStrToEuroStr);
```

Birinci satır, ABD doları dizelerinin eşleşmesi için basit bir deseni tanımlar. `$` karakterinin önüne ters eğik çizgi (`\`) kaçış karakterinin getirildiğine dikkat edin.

`replace()` yöntemi, desen eşleştirici olarak normal ifadeyi kullanır ve yerini alacak dizeyi (euro cinsinden bir değer) belirlemek için `usdStrToEuroStr()` işlevini çağırır.

`replace()` yönteminin ikinci parametresi olarak bir işlev adı kullanıldığında, çağrılan işleve parametre olarak şunlar iletilir:

- Dizenin eşleşen bölümü.
- Herhangi bir yakalanmış parantez grubu eşleşmesi. Bu şekilde iletilen argüman sayısı, yakalanan parantez grubu eşleşmelerinin sayısına bağlı olarak değişiklik gösterir. İşlev kodu içinde `arguments.length - 3` öğesini kontrol ederek, yakalanmış parantez grubu eşleşmelerinin sayısını belirleyebilirsiniz.
- Eşlemenin başladığı dize içindeki dizin konumu.
- Tam dize.

`usdStrToEuroStr()` yöntemi şu şekilde ABD doları dize desenlerini euro dizelerine dönüştürür:

```
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;
}
```

`args[1]` öğesinin, `usdPrice` normal ifadesinin eşleştiği yakalanmış parantez grubunu temsil ettiğini unutmayın. Bu, ABD doları dizesinin sayısal bölümüdür: başka bir deyişle, `$` işareti olmadan dolar miktarıdır. Bu yöntem, döviz kuru dönüştürmesi uygular ve bir sonuç dizesi (başında $ sembolü yerine, sonunda € sembolü bulunan) döndürür.

Bölüm 6: XML ile çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, XML için ECMAScript (E4X) belirtimini (ECMA-357 sürüm 2) esas alan bir sınıflar grubunu içerir. Bu sınıflar, XML verileriyle çalışılmasına yönelik güçlü ve kullanımı kolay işlevler sağlar. E4X'i kullanarak, önceki programlama teknikleriyle mümkün olandan daha hızlı şekilde XML verileriyle kod geliştirebilirsiniz. Ek bir avantaj olarak, oluşturduğunuz kodun okunması daha da kolay olur.

Daha fazla Yardım konusu

XML sınıfı

ECMA-357 tanımlaması

XML temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML, yapılandırılmış bilgilerin bilgisayarların birlikte çalışmasını ve insanların yazmasını ve anlamasını kolaylaştıracak şekilde temsil edilmesinin standart bir yoludur. XML, eXtensible Markup Language'in kısaltmasıdır. XML standardı, www.w3.org/XML/ adresinde bulunabilir.

XML, verinin okunmasını, işlenmesini ve veriye erişilmesini kolaylaştırmak için verilerin sınıflandırılmasına yönelik standart ve kullanışlı bir yol sunar. XML, bir ağaç yapısını ve HTML'e benzeyen bir etiket yapısını kullanır. Aşağıda, XML verisinin basit bir örneği verilmiştir:

```
<song>
    <title>What you know?</title>
    <artist>Steve and the flubberblubs</artist>
    <year>1989</year>
    <lastplayed>2006-10-17-08:31</lastplayed>
</song>
```

XML verileri, diğer etiketler içine yuvalanmış etiketler, nitelikler ve diğer yapısal bileşenlerle daha karmaşık da olabilir. Aşağıda, XML verisinin daha karmaşık bir örneği verilmiştir:

```
<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>
```

Bu XML belgesinin, içinde başka XML yapıları da (örn. alt öğeleri içinde `song` etiketleri) barındırdığına dikkat edin. Bu ayrıca veri yerine başka etiketler içeren nitelikler (`song` etiketlerindeki `tracknumber` ve `length` nitelikleri) ve etiketler (örn. `tracks` etiketi) gibi diğer XML yapılarını da gösterir.

XML ile çalışmaya başlama

XML ile ilgili çok az deneyiminiz varsa veya hiç deneyiminiz yoksa, XML verilerinin en yaygın yönleri burada kısaca açıklanmıştır. XML verileri, bilgilerin yapılandırılmış bir biçimde organize edilmesi için belirli bir sözdizimi kullanılarak düz metin biçiminde yazılır. Genellikle, tek bir XML verisi kümesi, *XML belgesi* olarak bilinir. XML biçiminde, veriler hiyerarşik bir yapı kullanılarak *öğeler* (bunlar tek veri öğeleri veya diğer öğeleri içeren kaplar olabilir) halinde organize edilir. Her XML belgesi, üst düzey veya ana öğe olarak tek bir öğeye sahiptir; bu kök öğenin içinde diğer öğeleri içeren başka öğeler bulunabileceği gibi tek bir bilgi de bulunabilir. Örneğin, bu XML belgesi, bir müzik albümüyle ilgili bilgileri içerir:

```
<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>
```

Her öğe, bir *etiketler* kümesiyle ayrılır—öğenin adı, açılı ayraçlar (küçüktür ve büyüktür işaretleri) içine sarılır. Öğenin başlangıcını belirten açma etiketi, öğe adını içerir:

```
<title>
```

Öğenin sonunu belirten kapatma etiketi, öğe adından önce eğik çizgi içerir:

```
</title>
```

Bir öğe içerik barındırmıyorsa, boş bir öğe olarak yazılabilir. (Bazen kendiliğinden kapanan öğe olarak adlandırılır.) XML'de bu öğe:

```
<lastplayed/>
```

bu öğeyle aynıdır:

```
<lastplayed></lastplayed>
```

Bir öğe, açma ve kapatma etiketleri arasında bulunan içeriğine ek olarak, öğenin açma etiketinde tanımlanan ve *nitelikler* olarak bilinen başka değerler de içerebilir. Örneğin, bu XML öğesi, `"4:19"` değerine sahip `length` adındaki tek bir niteliği tanımlar:

```
<song length="4:19"></song>
```

Her XML öğesi, tek bir değer veya bir ya da birkaç XML öğesi barındıran ya da herhangi bir şey barındırmayan (boş bir öğe için) bir içeriğe sahiptir.

XML hakkında daha fazla bilgi

XML ile çalışma hakkında daha fazla bilgi almak için, aralarında şu web sitelerinin de bulunduğu birçok ek kitap ve kaynak vardır:

- W3Schools XML Öğreticisi: http://w3schools.com/xml/
- XMLpitstop öğreticileri, tartışma listeleri ve daha fazlası: http://xmlpitstop.com/

XML ile çalışmaya yönelik ActionScript sınıfları

ActionScript 3.0, XML yapısındaki bilgilerle çalışmak için kullanılan birçok sınıf içerir. İki ana sınıf şunlardır:

- XML: Birden çok alt öğe içeren tek bir XML belgesi olabileceği gibi bir belge içindeki tek değerli bir öğe de olabilen tek bir XML öğesini temsil eder.
- XMLList: XML öğelerinin bir kümesini temsil eder. “eşdüzey” (XML belgesinin hiyerarşisinde, aynı düzeyde ve aynı üst öğede bulunan) niteliğinde birden çok XML öğesi olduğunda, XMLList nesnesi kullanılır. Örneğin, bu XML öğeleri kümesiyle (büyük olasılıkla bir XML belgesinde) çalışmanın en kolay yolu, XMLList örneğidir:

  ```
  <artist type="composer">Fred Wilson</artist>
  <artist type="conductor">James Schmidt</artist>
  <artist type="soloist">Susan Harriet Thurndon</artist>
  ```

XML ad alanlarının da dahil olduğu daha ileri düzey kullanımlar için ActionScript ayrıca Namespace ve QName sınıflarını da içerir. Daha fazla bilgi için, bkz. “XML ad alanlarını kullanma” sayfa 111.

ActionScript 3.0, XML ile çalışılmasına yönelik yerleşik sınıflara ek olarak, XML verilerine erişilmesi ve XML verilerinin işlenmesi için de belirli işlevler sağlayan birçok operatör içerir. Bu sınıf ve operatörler kullanılarak XML ile çalışma yaklaşımı, ECMA-357 sürüm 2 belirtimi tarafından tanımlandığı şekilde, XML için ECMAScript (E4X) olarak bilinir.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi XML işleme yordamlarını programlarken karşılaşacağınız önemli terimleri içerir:

Öğe Başlatma etiketi ile kapatma etiketi arasında barındırılan içerik (etiketler de dahil) olarak tanımlanan, bir XML belgesindeki tek bir öğe. XML öğeleri, metin verileri veya başka öğeler içerebilir veya boş olabilir.

Boş öğe Herhangi bir alt öğe içermeyen XML öğesi. Boş öğeler genellikle kendiliğinden kapanan etiketler olarak (örn. `<element/>`) yazılır.

Belge Tek bir XML yapısı. Bir XML belgesi herhangi bir sayıda öğe içerebilir (veya tek bir boş öğeden oluşabilir); ancak XML belgesinin, belgedeki diğer tüm öğeleri içeren tek bir üst düzey öğeye sahip olması gerekir.

Düğüm XML öğesinin başka bir adı.

Nitelik Öğenin içinde yuvalanmış ayrı bir alt öğe olarak yazılmayıp `attributename="value"` biçiminde açma etiketinin içine yazılan, bir öğeyle ilişkilendirilmiş adlandırılmış bir değer.

XML işlemeye yönelik E4X yaklaşımı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML için ECMAScript belirtimi, XML verileriyle çalışılmasına yönelik sınıflar ve işlevlerin bir kümesini tanımlar. Bu sınıflar ve işlevler topluca *E4X olarak bilinir.* ActionScript 3.0, şu E4X sınıflarını içerir: XML, XMLList, QName ve Namespace.

E4X sınıflarının yöntemleri, özellikleri ve operatörleri, şu hedefler göz önünde bulundurularak tasarlanmıştır:

- Basitlik—E4X, XML verileriyle çalışılmasına yönelik kodların yazılmasını ve anlaşılmasını olabildiğince kolaylaştırır.
- Tutarlılık—E4X'in ardındaki mantık ve yöntemler, ActionScript'in diğer bölümlerindekilerle dahili olarak tutarlılık gösterir.
- Bilindik Olması—XML verilerini, nokta (.) operatörü gibi bilindik operatörlerle işlersiniz.

Not: *ActionScript 2.0'da farklı bir XML sınıfı vardı. ActionScript 3.0'da bu sınıf, E4X'in bir parçası olan ActionScript 3.0 XML sınıfı ile çakışmaması için XMLDocument olarak yeniden adlandırılmıştır. ActionScript 3.0'da, XMLDocument, XMLNode, XMLParser ve XMLTag gibi eski sınıflar, eski sürümlerin desteklenmesi için flash.xml paketine dahil edilmiştir. Yeni E4X sınıfları çekirdek sınıflardır; bunları kullanmak için bir paketi içe aktarmanız gerekmez. ActionScript 2.0 XML sınıflarıyla ilgili ayrıntılı bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki flash.xml paketine bakın.*

Aşağıda, E4X ile veri işlenmesine örnek verilmiştir:

```
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>
```

Genellikle uygulamanız bir web hizmeti veya RSS beslemesi gibi harici bir kaynaktan XML verilerini yükler. Ancak, daha belirgin olmaları için, burada sağlanan kod örnekleri XML verisini değişmez değerler olarak atar.

Aşağıdaki kodda gösterildiği gibi, E4X, XML'deki özellik ve niteliklere erişilmesi için, nokta (.) ve nitelik tanımlayıcısı (@) operatörleri gibi bazı sezgisel operatörleri içerir:

```
trace(myXML.item[0].menuName); // Output: burger
trace(myXML.item.(@id==2).menuName); // Output: fries
trace(myXML.item.(menuName=="burger").price); // Output: 3.95
```

Aşağıdaki kod parçasında gösterildiği gibi, XML'e yeni bir alt düğüm atamak için `appendChild()` yöntemini kullanın:

```
var newItem:XML =
    <item id="3">
        <menuName>medium cola</menuName>
        <price>1.25</price>
    </item>

myXML.appendChild(newItem);
```

Aşağıda gösterildiği gibi, verileri okumak ve atamak için `@` ve `.` operatörlerini kullanın:

```
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";
```

Aşağıdaki gibi, XML düğümlerini yinelemek için bir `for` döngüsü kullanın:

```
var total:Number = 0;
for each (var property:XML in myXML.item)
{
    var q:int = Number(property.@quantity);
    var p:Number = Number(property.price);
    var itemTotal:Number = q * p;
    total += itemTotal;
    trace(q + " " + property.menuName + " $" + itemTotal.toFixed(2))
}
trace("Total: $", total.toFixed(2));
```

XML nesneleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML nesnesi, bir XML öğesini, niteliği, yorumu, işleme talimatını veya metin öğesini temsil edebilir.

XML nesnesi, *basit içerik* veya *karmaşık içerik* barındırmasına göre sınıflandırılır. Alt düğümler içeren bir XML nesnesi, karmaşık içerik barındırıyor olarak sınıflandırılır. XML nesnesi şunlardan herhangi biriyse, basit içerik barındırıyor olarak sınıflandırılır: bir nitelik, bir yorum, bir işleme talimatı veya bir metin düğümü.

Örneğin, aşağıdaki XML nesnesi, bir yorum ve işleme talimatı olmak üzere karmaşık içerik barındırır:

```
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>
```

Aşağıdaki örnekte gösterildiği gibi, yeni XML nesneleri, yorum ve işleme talimatı oluşturmak için şimdi `comments()` ve `processingInstructions()` yöntemlerini kullanabilirsiniz:

```
var x2:XML = x1.comments()[0];
var x3:XML = x1.processingInstructions()[0];
```

XML özellikleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML sınıfı beş statik özellik içerir:

- `ignoreComments` ve `ignoreProcessingInstructions` özellikleri, XML nesnesi ayrıştırılırken yorumların veya işleme talimatlarının yok sayılıp yok sayılmayacağını belirler.
- `ignoreWhitespace` özelliği, öğe etiketlerinde ve yalnızca boşluk karakterleriyle ayrılan gömülü ifadelerde boşluk karakterlerinin yok sayılıp yok sayılmayacağını belirler.
- `prettyIndent`ve `prettyPrinting` özellikleri, XML sınıfının `toString()` ve `toXMLString()` yöntemleri tarafından döndürülen metni biçimlendirmek için kullanılır.

Bu özelliklerle ilgili ayrıntılar için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

XML yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki yöntemler, XML nesnelerinin hiyerarşik yapısıyla çalışmanıza olanak sağlar:

- `appendChild()`
- `child()`
- `childIndex()`
- `children()`
- `descendants()`
- `elements()`
- `insertChildAfter()`
- `insertChildBefore()`

- `parent()`
- `prependChild()`

Aşağıdaki yöntemler, XML nesnesi nitelikleriyle çalışmanıza olanak sağlar:

- `attribute()`
- `attributes()`

Aşağıdaki yöntemler, XML nesnesi özellikleriyle çalışmanıza olanak sağlar:

- `hasOwnProperty()`
- `propertyIsEnumerable()`
- `replace()`
- `setChildren()`

Aşağıdaki yöntemler, nitelendirilmiş adlarla ve ad alanlarıyla çalışılmasına yöneliktir:

- `addNamespace()`
- `inScopeNamespaces()`
- `localName()`
- `name()`
- `namespace()`
- `namespaceDeclarations()`
- `removeNamespace()`
- `setLocalName()`
- `setName()`
- `setNamespace()`

Aşağıdaki yöntemler, belirli türdeki XML içerikleriyle çalışılmasına ve bu XML içeriklerinin belirlenmesine yöneliktir:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `nodeKind()`
- `processingInstructions()`
- `text()`

Aşağıdaki yöntemler, dizelere dönüştürmeye ve XML nesnelerinin biçimlendirmesine yöneliktir:

- `defaultSettings()`
- `setSettings()`
- `settings()`
- `normalize()`
- `toString()`
- `toXMLString()`

Birkaç ek yöntem daha vardır:

- `contains()`
- `copy()`
- `valueOf()`
- `length()`

Bu yöntemlerle ilgili ayrıntılı bilgi için bkz. Adobe Flash Platform için ActionScript 3.0 Başvurusu.

XMLList nesneleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XMLList örneği, XML nesnelerinin rastgele bir koleksiyonunu temsil eder. Bunlar arasında tam XML belgeleri, XML parçaları veya XML sorgusunun sonuçları yer alabilir.

Aşağıdaki yöntemler, XMLList nesnelerinin hiyerarşik yapısıyla çalışmanıza olanak sağlar:

- `child()`
- `children()`
- `descendants()`
- `elements()`
- `parent()`

Aşağıdaki yöntemler, XMLList nesnesi nitelikleriyle çalışmanıza olanak sağlar:

- `attribute()`
- `attributes()`

Aşağıdaki yöntemler, XMLList özellikleriyle çalışmanıza olanak sağlar:

- `hasOwnProperty()`
- `propertyIsEnumerable()`

Aşağıdaki yöntemler, belirli türdeki XML içerikleriyle çalışılmasına ve bu XML içeriklerinin belirlenmesine yöneliktir:

- `comments()`
- `hasComplexContent()`
- `hasSimpleContent()`
- `processingInstructions()`
- `text()`

Aşağıdaki yöntemler, dizelere dönüştürmeye ve XMLList nesnelerinin biçimlendirmesine yöneliktir:

- `normalize()`
- `toString()`
- `toXMLString()`

Birkaç ek yöntem daha vardır:

- `contains()`

- `copy()`
- `length()`
- `valueOf()`

Bu yöntemlerle ilgili ayrıntılı bilgi için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Tek bir XML öğesi içeren bir XMLList nesnesi, XML nesnesiyle aynı şekilde değerlendirildiğinden, tam olarak bir tane XML öğesi içeren bir XMLList nesnesi için, XML sınıfının tüm özelliklerini ve yöntemlerini kullanabilirsiniz. Örneğin, aşağıdaki kodda, `doc.div` tek bir öğe içeren bir XMLList nesnesi olduğundan, XML sınıfındaki `appendChild()` yöntemini kullanabilirsiniz:

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

XML özelliklerinin ve yöntemlerinin listesi için, bkz. "XML nesneleri" sayfa 100.

XML değişkenlerini başlatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki gibi, XML nesnesine bir XML değişmezi atayabilirsiniz:

```
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>
```

Aşağıdaki kod parçasında gösterildiği gibi, XML verisi içeren bir dizeden bir XML nesnesi örneği oluşturmak için, `new` yapıcısını da kullanabilirsiniz:

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                            + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

Dizedeki XML verileri düzgün biçimlendirilmemişse (örneğin, bir kapatma etiketi eksikse), çalışma zamanı hatasıyla karşılaşırsınız.

Ayrıca aşağıdaki örnekte gösterildiği gibi, verileri başvuruya göre (diğer değişkenlerden) bir XML nesnesine iletebilirsiniz:

```
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>
```

Bir URL'den XML verileri yüklemek için, aşağıdaki örnekte gösterildiği gibi, URLLoader sınıfını kullanın:

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

Bir soket bağlantısından XML verilerini okumak için, XMLSocket sınıfını kullanın. Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki XMLSocket sınıfına bakın.

XML nesnelerini birleştirme ve dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML nesnesinin özellikler listesinin başına veya sonuna bir özellik eklemek için, aşağıdaki örnekte gösterildiği gibi, `prependChild()` yöntemini veya `appendChild()` yöntemini kullanın:

```
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>
```

Belirtilen bir özellikten önce veya sonra bir özellik eklemek için, aşağıda gösterildiği gibi, `insertChildBefore()` yöntemini ya da `insertChildAfter()` yöntemini kullanın:

```
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>)
```

Aşağıdaki örnekte gösterildiği gibi, XML nesneleri oluştururken başvuruya göre (diğer değişkenlerden) veri iletmek için küme ayracı operatörlerini de ({ ve }) kullanabilirsiniz:

```
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)
}
```

Aşağıdaki gibi, = operatörünü kullanarak bir XML nesnesine özellikler ve nitelikler atayabilirsiniz:

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

Böylece x XML nesnesi şunlara ayarlanır:

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

XMLList nesnelerini bitiştirmek için + ve += operatörlerini kullanabilirsiniz:

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

Böylece `xList` XMLList nesnesi şunlara ayarlanır:

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

XML yapılarında geçiş yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML'in güçlü özelliklerinden biri, doğrusal metin karakterleri dizesi yoluyla karmaşık, yuvalanmış veriler sağlama yeteneğidir. Bir XML nesnesine veri yüklediğinizde, ActionScript veriyi ayrıştırır ve verinin hiyerarşik yapısını belleğe gönderir (veya XML verisi düzgün biçimlendirilmemişse bir çalışma zamanı hatası gönderir).

XML ve XMLList nesnelerinin operatörleri ve yöntemleri, XML verilerinde geçiş yapılmasını kolaylaştırır.

Bir XML nesnesinin alt özelliklerine erişmek için, nokta (.) operatörünü ve alt öğe erişimcisi (..) operatörünü kullanın. Şu XML nesnesini göz önünde bulundurun:

```
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>
```

`myXML.book` nesnesi, `myXML` nesnesinin, `book` adına sahip olan alt özelliklerini içeren bir XMLList nesnesidir. Bunlar, `myXML` nesnesinin iki `book` özelliğiyle eşleşen iki XML nesnesidir.

`myXML..lastName` nesnesi, `lastName` adına sahip olan tüm alt öğe özelliklerini içeren bir XMLList nesnesidir. Bunlar, `myXML` nesnesinin iki `lastName` özelliğiyle eşleşen iki XML nesnesidir.

`myXML.book.editor.lastName` nesnesi, `myXML` nesnesinin `book` adına sahip alt öğelerinin `editor` adına sahip alt öğelerinin `lastName` adına sahip herhangi bir alt öğesini içeren XMLList nesnesidir: bu durumda, yalnızca bir XML nesnesi içeren bir XMLList nesnesidir ("`Case`" değerine sahip `lastName` özelliği).

Üst ve alt düğümlere erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`parent()` yöntemi, XML nesnesinin üst öğesini döndürür.

Belirli alt nesnelere erişmek için, bir alt öğe listesinin sıra belirten dizin değerlerini kullanabilirsiniz. Örneğin, `book` adına sahip iki alt özellik içeren bir `myXML` XML nesnesini göz önünde bulundurun. `book` olarak adlandırılan her alt özellik, bununla ilişkilendirilmiş bir dizin sayısına sahiptir:

```
myXML.book[0]
myXML.book[1]
```

Belirli bir alt alt öğeye erişmek için, hem alt öğe hem de alt alt öğe adları için dizin sayıları belirtebilirsiniz:

```
myXML.book[0].title[0]
```

Ancak, `x.book[0]` öğesinin, `title` başlık adına sahip yalnızca bir alt öğesi varsa, şu şekilde dizin başvurusunu çıkarabilirsiniz:

```
myXML.book[0].title
```

Aynı şekilde, `x` nesnesinin yalnızca bir kitap alt öğesi varsa ve bu alt nesne yalnızca bir başlık nesnesine sahipse, şu şekilde her iki dizin başvurusunu çıkarabilirsiniz:

```
myXML.book.title
```

Aşağıdaki örnekte gösterildiği gibi, bir değişken veya ifadeyi esas alan adlara sahip alt öğelere gitmek için `child()` yöntemini kullanabilirsiniz:

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

Niteliklere erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki kodda gösterildiği gibi, bir XML veya XMLList nesnesindeki niteliklere erişmek için `@` sembolünü (nitelik tanımlayıcısı operatörü) kullanın:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

Aşağıdaki kodda gösterildiği gibi, bir XML veya XMLList nesnesinin tüm niteliklerine erişmek için, `@` sembolüyle birlikte `*` joker karakter sembolünü kullanabilirsiniz:

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

Aşağıdaki kodda olduğu gibi, bir XML veya XMLList nesnesinin belirli bir niteliğine veya tüm niteliklerine erişmek için, `attribute()` ya da `attributes()` yöntemini kullanabilirsiniz:

```
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
```

Aşağıdaki örnekte gösterildiği gibi, niteliklere erişmek için şu sözdizimini de kullanabildiğinizi unutmayın:

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

Bunların her biri şuna eşdeğerdir: `employee.@id`. Ancak tercih edilen yaklaşım, `employee.@id` sözdizimidir.

Niteliğe veya öğe değerine göre filtre uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Belirli bir öğe adına veya nitelik değerine sahip öğelere filtre uygulamak için, parantez operatörlerini— `(` ve `)` — kullanabilirsiniz. Şu XML nesnesini göz önünde bulundurun:

```
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>
```

Aşağıdaki ifadelerin tümü geçerlidir:

- `x.employee.(lastName == "McGee")`—Bu, ikinci `employee` düğümüdür.
- `x.employee.(lastName == "McGee").firstName`—Bu, ikinci `employee` düğümünün `firstName` özelliğidir.
- `x.employee.(lastName == "McGee").@id`—Bu, ikinci `employee` düğümünün `id` niteliğidir.
- `x.employee.(@id == 347)`—Birinci `employee` düğümü.
- `x.employee.(@id == 347).lastName`—Bu, birinci `employee` düğümünün `lastName` özelliğidir.
- `x.employee.(@id > 300)`—Bu, her iki `employee` özelliğini içeren bir XMLList öğesidir.
- `x.employee.(position.toString().search("analyst") > -1)`—Bu, her iki `position` özelliğini içeren bir XMLList öğesidir.

Var olmayan nitelikler veya öğeleri filtrelemeye çalışırsanız, bir istisna atılır. Örneğin, ikinci `p` öğesinde `id` niteliği bulunmadığından, aşağıdaki kodun son satırı bir hata oluşturur:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

Aynı şekilde, ikinci `p` öğesinin `b` özelliği bulunmadığından, aşağıdaki kodun son satırı bir hata oluşturur:

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

Bu hataları önlemek için, aşağıdaki kodda olduğu gibi, `attribute()` ve `elements()` yöntemlerini kullanarak, eşleşen niteliklere veya öğelere sahip özellikleri tanımlayabilirsiniz:

```
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'));
```

Ayrıca aşağıdaki kodda olduğu gibi, `hasOwnProperty()` yöntemini de kullanabilirsiniz:

```
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'));
```

for..in ve for each..in deyimlerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, XMLList nesnelerinin yinelenmesi için `for..in` deyimini ve `for each..in` deyimini içerir. Örneğin, şu `myXML` XML nesnesini ve `myXML.item` XMLList nesnesini göz önünde bulundurun. `myXML.item` XMLList nesnesi, XML nesnesinin iki `item` düğümünü içerir.

```
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>;
```

`for..in` deyimi, bir XMLList öğesindeki özellik adları kümesini yinelemenize olanak sağlar:

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

`for each..in` deyimi, XMLList öğesindeki özellikleri yinelemenize olanak sağlar:

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

XML ad alanlarını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML nesnesindeki (veya belgesindeki) ad alanları, nesnenin içerdiği veri türünü tanımlar. Örneğin, SOAP mesajlaşma protokolünü kullanan bir web hizmetine XML verilerini gönderip teslim ederken, XML'in açma etiketinde ad alanını bildirirsiniz:

```
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>;
```

Ad alanı, bir önek (`soap`) ve ad alanını tanımlayan bir URI (`http://schemas.xmlsoap.org/soap/envelope/`) içerir.

ActionScript 3.0, XML ad alanlarıyla çalışılması için Namespace sınıfını içerir. Önceki örnekte bulunan XML nesnesi için, şu şekilde Namespace sınıfını kullanabilirsiniz:

```
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";
```

XML sınıfı, ad alanlarıyla çalışılması için şu yöntemleri içerir: `addNamespace()`, `inScopeNamespaces()`, `localName()`, `name()`, `namespace()`, `namespaceDeclarations()`, `removeNamespace()`, `setLocalName()`, `setName()` ve `setNamespace()`.

`default xml namespace` direktifi, XML nesneleri için varsayılan bir ad alanı atamanıza olanak sağlar. Örneğin, aşağıda, hem `x1` hem de `x2` aynı varsayılan ad alanına sahiptir:

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

XML tür dönüştürmesi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML nesnelerini ve XMLList nesnelerini String değerlerine dönüştürebilirsiniz. Aynı şekilde, dizeleri de XML nesnelerine ve XMLList nesnelerine dönüştürebilirsiniz. Ayrıca, tüm XML nitelik değerlerinin, adlarının ve metin değerlerinin dize olduğunu unutmayın. İlerleyen bölümlerde, bu XML tür dönüştürmesi biçimlerinin tümü ele alınmıştır.

XML ve XMLList nesnelerini dizelere dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML ve XMLList sınıfları, bir `toString()` yöntemini ve bir `toXMLString()` yöntemini içerir. `toXMLString()` yöntemi, XML nesnesinin tüm etiketlerini, niteliklerini, ad alanı bildirimlerini ve içeriklerini barındıran bir dize döndürür. Karmaşık içerik (alt öğeler) barındıran XML nesneleri için, `toString()` yöntemi `toXMLString()` yöntemiyle tamamen aynı şeyi yapar. Basit içerik barındıran XML nesneleri (yalnızca bit metin öğesi barındıran) için, `toString()` yöntemi aşağıdaki örnekte gösterildiği gibi, yalnızca öğenin metin içeriğini döndürür:

```
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
```

`toString()` veya `toXMLString()` öğesini belirtmeden `trace()` yöntemini kullanırsanız, bu kodda gösterildiği gibi, varsayılan olarak `toString()` yöntemi kullanılarak veriler dönüştürülür:

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

Kodun hatalarını ayıklamak için `trace()` yöntemini kullanırken genellikle `trace()` yönteminin daha tam veri vermesi için `toXMLString()` yöntemini kullanmayı istersiniz.

Dizeleri XML nesnelerine dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki gibi, bir dizeden XML nesnesi oluşturmak için, `new XML()` yapıcısını kullanabilirsiniz:

```
var x:XML = new XML("<a>test</a>");
```

Aşağıdaki gibi, bir dizeyi, geçersiz XML'i veya düzgün biçimlendirilmemiş XML'i temsil eden bir dizeden XML'e dönüştürmeyi denerseniz, bir çalışma zamanı hatası atılır:

```
var x:XML = new XML("<a>test"); // throws an error
```

Dizelerden nitelik değerlerini, adları ve metin değerlerini dönüştürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tüm XML nitelik değerleri, adları ve metin değerleri, String veri türündedir ve bunları başka veri türlerine dönüştürmeniz gerekebilir. Örneğin, aşağıdaki kod, metin değerlerini sayılara dönüştürmek için `Number()` işlevini kullanır:

```
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;
```

Bu kod `Number()` işlevini kullanmasaydı, kod, + operatörünü dize bitiştirme operatörü olarak yorumlardı ve son satırdaki `trace()` yöntemi çıktı olarak şunu verirdi:

```
01.003.95
```

Harici XML belgelerini okuma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir URL'den XML verilerini yüklemek için URLLoader sınıfını kullanabilirsiniz. Uygulamalarınızda aşağıdaki kodu kullanmak için, örnekteki `XML_URL` değerinin yerine geçerli bir URL getirin:

```
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.");
}
```

Ayrıca bir sunucuyla senkronize olmayan XML soket bağlantısı kurmak için XMLSocket sınıfını kullanabilirsiniz. Daha fazla bilgi için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

ActionScript'te XML örneği: internetten RSS verilerini yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RSSViewer örnek uygulaması, ActionScript'te XML ile çalışılmasının birçok özelliğini gösterir, örn:

- XML verilerinde RSS beslemesi biçiminde geçiş yapmak için XML yöntemlerini kullanma.
- XML verilerini bir metin alanında kullanmak üzere HTML biçiminde birleştirmek için XML yöntemlerini kullanma.

RSS biçimi, XML yoluyla haberleri yayınlamak için yaygın olarak kullanılır. Basit bir RSS veri dosyası şöyle görünebilir:

```
<?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>
```

SimpleRSS uygulaması, Internet'ten RSS verilerini okur, başlık, bağlantı ve açıklamalar için verileri ayrıştırır ve bu verileri döndürür. SimpleRSSUI sınıfı kullanıcı arabirimini sağlar ve tüm XML işlemeyi gerçekleştiren SimpleRSS sınıfını çağırır.

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. RSSViewer uygulama dosyalarını Samples/RSSViewer klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
RSSViewer.mxml veya RSSViewer.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/rssViewer/RSSParser.as	RSS (XML) verilerinde geçiş yapmak ve karşılık gelen HTML temsilini oluşturmak için E4X kullanan yöntemleri içeren bir sınıf.
RSSData/ak.rss	Örnek bir RSS dosyası. Bu uygulama, web'deki RSS verileri, Adobe tarafından barındırılan bir Flex RSS beslemesinde okunacak şekilde ayarlanmıştır. Ancak uygulamayı, Flex RSS beslemesinden daha farklı bir şema kullanan bu belgedeki RSS verilerini okuyacak şekilde kolayca değiştirebilirsiniz.

XML verilerini okuma ve ayrıştırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RSSParser sınıfı, `rssXML` değişkeninde saklanan girdi RSS verilerini HTML olarak biçimlendirilmiş çıktı (`rssOutput`) içeren bir dizeye dönüştüren `xmlLoaded()` yöntemini içerir.

Kaynak RSS verileri varsayılan bir ad alanı içeriyorsa, kod, yöntemin başına yakın konumda varsayılan XML ad alanını ayarlar:

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

Sonraki satırlar, kaynak XML verisinin içerikleri üzerinde döngü gerçekleştirerek `item` adındaki alt öğe özelliklerinin her birini inceler:

```
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);
}
```

İlk üç satır, dize değişkenlerini, XML verisinin `item` özelliğinin başlık, açıklama ve bağlantı özelliklerini temsil edecek şekilde ayarlar. Sonraki satır, parametre olarak üç yeni dize değişkeni kullanarak, HTML verilerini bir XMLList nesnesi biçiminde almak için `buildItemHTML()` yöntemini çağırır.

XMLList verilerini birleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

HTML verileri (bir XMLList nesnesi) şu biçimde olabilir:

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

Yöntemin birinci satırları, varsayılan xml ad alanını temizler:

```
default xml namespace = new Namespace();
```

`default xml namespace` direktifi, blok düzeyinde kapsam işlevine sahiptir. Başka bir deyişle, bu bildirimin kapsamı `buildItemHTML()` yöntemidir.

Bunu izleyen satırlar, işleve iletilen dize argümanlarını esas alarak XMLList öğesini oluşturur:

```
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;
```

Bu XMLList nesnesi, ActionScript HTML metin alanı için uygun olan dize verisini temsil eder.

`xmlLoaded()` yöntemi, `buildItemHTML()` yönteminin döndürme değerini kullanır ve bunu bir dizeye dönüştürür:

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

RSS beslemesinin başlığını ayıklama ve özel bir olay gönderme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`xmlLoaded()` yöntemi, kaynak RSS XML verisindeki bilgileri esas alarak bir `rssTitle` dize değişkeni ayarlar:

```
rssTitle = rssXML.channel.title.toString();
```

Son olarak, `xmlLoaded()` yöntemi, verilerin ayrıştırıldığını ve kullanılabilir olduğunu uygulamaya bildiren bir olay oluşturur:

```
            dataWritten = new Event("dataWritten", true);
```

Bölüm 7: Yerel JSON işlevini kullanma

ActionScript 3.0, ActionScript nesnelerini JavaScript Nesne Gösterimi (JSON) biçimi kullanarak kodlamak ve bunların kodunu çözmek için yerel bir API sağlar. JSON sınıfı ve destekleyici öğe işlevleri, çok az farkla ECMA-262 5. sürüm belirtimini izler.

Topluluk üyesi Todd Anderson, yerel JSON API ve üçüncü taraf as3corelib JSON sınıfının bir karşılaştırmasını sağlamaktadır. Bkz. Working with Native JSON in Flash Player 11 (Flash Player 11'de Yerel JSON ile Çalışma).

Daha fazla Yardım konusu

JSON

JSON API'sine genel bakış

ActionScript JSON API'si, JSON sınıfı ve birkaç yerel sınıftaki `toJSON()` üye işlevlerinden oluşur. Herhangi bir sınıfa yönelik özel JSON kodlaması gerektiren uygulamalar için ActionScript çerçevesi, varsayılan kodlamayı geçersiz kılmaya yönelik yollar sağlar.

JSON sınıfı, `toJSON()` üyesi sağlamayan herhangi bir ActionScript sınıfı için içe aktarmayı ve dışa aktarmayı dahili olarak işler. Bu durumlar için JSON, karşılaştığı her nesnenin ortak özelliklerine çapraz geçiş uygular. Bir nesne başka nesneler içeriyorsa JSON, yuvalanmış nesnelerde yinelenir ve aynı çapraz geçişi gerçekleştirir. Herhangi bir nesne `toJSON()` yöntemi sunuyorsa JSON, iç algoritmaları yerine bu özel yöntemi kullanır.

JSON arabirimi `stringify()` kodlama yöntemi ve `parse()` kod çözme yönteminden oluşur. Bu yöntemlerin her biri, JSON kodlama veya kod çözme iş akışına kendi mantığınızı eklemenize olanak tanıyan bir parametre sağlar. Bu parametre, `stringify()` için `replacer`, `parse()` için ise `reviver` olarak adlandırılır. Bu parametreler, aşağıdaki imzayı kullanarak iki bağımsız değişkenli bir işlev tanımı alır:

```
function(k, v):*
```

toJSON() yöntemleri

`toJSON()` yönteminin imzası şudur:

```
public function toJSON(k:String):*
```

Varsa, `JSON.stringify()` öğesi, nesne çapraz geçişi sırasında karşılaştığı her bir ortak özellik için `toJSON()` öğesini çağırır. Özellik anahtar-değer çiftinden oluşur. `stringify(`, `toJSON()` öğesini çağırdığında o anda incelemekte olduğu özelliğin anahtarını (`k`) iletir. Tipik bir `toJSON()` uygulaması, her bir özellik adını inceler ve istenen kodlama değerini getirir.

`toJSON()` yöntemi, yalnızca bir String öğesini değil herhangi bir türün değerini (* olarak belirtilen) döndürebilir. Bu değişken döndürme türü `toJSON()` öğesinin bir nesneyi, uygunsa, döndürmesine olanak tanır. Örneğin, özel sınıfınızın bir özelliği üçüncü taraf bir kütüphaneden nesne içeriyorsa, bu nesneyi `toJSON()` özelliğinizle karşılaştığında döndürebilirsiniz. Sonrasında JSON üçüncü taraf nesnesinde yinelenir. Kodlama işlem akışı aşağıdaki şekilde davranır:

- `toJSON()` öğesi, dizeyi değerlendirmeyen bir nesne döndürürse `stringify()` öğesi bu nesnede yinelenir.
- `toJSON()` bir dize döndürürse, `stringify()` bu değeri başka bir dizede sarar, sarılmış dizeyi döndürür ve ardından bir sonraki değere geçer.

Birçok durumda, bir nesne döndürmek uygulamanız tarafından oluşturulan JSON dizesini döndürmeye tercih edilir. Bir nesne döndürmek, yerleşik JSON kodlaması algoritmasından faydalanır ve ayrıca JSON öğesinin yuvalanmış nesnelerde yinelenmesine izin verir.

`toJSON()` yöntemi, Object sınıfı veya diğer birçok yerel sınıfta tanımlanmaz. Yokluğu, JSON'a nesnenin ortak özellikleri üzerinde standart çapraz geçişini gerçekleştirmesini belirtir. İsterseniz, nesnenizin özel özelliklerini göstermek için `toJSON()` öğesini de kullanabilirsiniz.

Birkaç yerel sınıf, ActionScript kütüphanelerinin tüm kullanma durumları için etkin bir şekilde çözemediği güçlüklere neden olur. ActionScript, bu sınıflar için, istemcilerin ihtiyaçlarına uyması için yeniden uygulayabileceği küçük bir uygulama sağlar. Önemsiz `toJSON()` üyeleri sağlayan sınıflar şunlardır:

- ByteArray
- Date
- Dictionary
- XML

ByteArray sınıfını, `toJSON()` yönteminin üzerine yazmak için alt sınıflandırabilirsiniz veya sınıfın prototipini yeniden tanımlayabilirsiniz. Son olarak belirtilen Date ve XML sınıfları, `toJSON()` öğesini yeniden tanımlamak için sınıf prototipini kullanmanızı gerektirir. Dictionary sınıfı dinamik olarak belirtilir ve bu size `toJSON()` öğesinin üzerine yazmanız için fazladan özgürlük verir.

Özel JSON davranışını tanımlama

Yerel sınıflar için kendi JSON kodlamanızı ve kod çözme işleminizi uygulamak için birkaç seçenek arasından seçim yapabilirsiniz:

- Nihai olmayan yerel bir sınıfın özel alt sınıfında `toJSON()` öğesini tanımlama veya geçersiz kılma
- Sınıf prototipinde `toJSON()` öğesini tanımlama veya yeniden tanımlama
- Dinamik bir sınıfta `toJSON` özelliğini tanımlama
- `JSON.stringify() replacer` ve `JSON.parser() reviver` parametrelerini kullanma

Daha fazla Yardım konusu

ECMA-262, 5. Sürüm

toJSON() öğesini yerleşik bir sınıf prototipinde tanımlama

ActionScript'teki yerel JSON uygulaması, ECMA-262, 5. sürümde tanımlanan ECMAScript JSON mekanizmasını yansıtır. ECMAScript, sınıfları desteklemediğinden ActionScript, JSON davranışını prototip bazlı gönderme bakımından tanımlar. Prototipler benzetilmiş devralma işleminin yanı sıra üye eklentilerine ve yeniden tanımlamalarına izin veren ActionScript 3.0 sınıfı habercileridir.

ActionScript, `toJSON()` öğesini herhangi bir sınıf prototipinde tanımlamanıza ve yeniden tanımlamanıza olanak tanır. Bu ayrıcalık, son olarak işaretlenmiş sınıflar için bile geçerlidir. `toJSON()` öğesini bir sınıf prototipinde tanımladığınızda tanımınız, uygulama alanınızda bu sınıfın tüm örnekleri için geçerli olur. Örneğin burada bir `toJSON()` yöntemini MovieClip prototipinde nasıl tanımlayabileceğiniz bulunmaktadır:

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

Sonrasında uygulamanız `stringify()` öğesini herhangi bir MovieClip örneğinde çağırdığında `stringify()` öğesi, `toJSON()` yönteminizin çıktısını döndürür:

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

Ayrıca yöntemi tanımlayan yerel sınıflarda `toJSON()` öğesini geçersiz kılabilirsiniz. Örneğin, aşağıdaki kod `Date.toJSON()` öğesini geçersiz kılar:

```
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"
```

Sınıf düzeyinde toJSON() öğesini tanımlama veya geçersiz kılma

`toJSON()` öğesini yeniden tanımlamak üzere prototipleri kullanmak için uygulamalara her zaman gerek yoktur. Üst sınıf nihai olarak işaretlenmemişse `toJSON()` öğesini bir alt sınıfın üyesi olarak tanımlayabilirsiniz. Örneğin, ByteArray sınıfını genişletebilir ve genel bir `toJSON()` işlevi belirtebilirsiniz:

```
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"
```

Bir sınıf dinamikse bu sınıfın nesnesine `toJSON` özelliğini ekleyebilirsiniz ve buna aşağıda olduğu gibi bir işlev atayabilirsiniz:

```
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."}
```

`toJSON()` öğesini herhangi bir ActionScript sınıfında geçersiz kılabilir, tanımlayabilir veya yeniden tanımlayabilirsiniz. Ancak, çoğu yerleşik ActionScript sınıfı `toJSON()` öğesini tanımlamaz. Object sınıfı `toJSON` öğesini kendi varsayılan prototipinde tanımlamaz veya onu bir sınıf üyesi olarak bildirmez. Yerel sınıfların yalnızca çok azı yöntemi bir prototip işlevi olarak tanımlar. Bu nedenle `toJSON()` öğesini çoğu sınıfta geleneksel anlamda geçersiz kılamazsınız.

`toJSON()` öğesini tanımlamayan yerel sınıflar dahili JSON uygulaması tarafından JSON öğesine serileştirilirler. Mümkünse bu yerleşik işlevi değiştirmekten kaçının. Bir `toJSON()` öğesi tanımlarsanız, JSON sınıfı kendi işlevselliği yerine sizin mantığınızı kullanır.

JSON.stringify() değiştirici parametresini kullanma

Prototip üstünde `toJSON()` öğesinin üzerine yazma, uygulamada bir sınıfın JSON dışa aktarma davranışını değiştirmek için yararlıdır. Ancak bazı durumlarda, dışa aktarma mantığınız yalnızca geçici koşullar altındaki özel durumlar için geçerli olabilir. Böyle küçük kapsam değişikliklerini düzenlemek için `JSON.stringify()` yönteminin `replacer` parametresini kullanabilirsiniz.

`stringify()` yöntemi, `replacer` parametresinden kodlanmakta olan nesneye geçirilen işlevi uygular. Bu işlevin imzası `toJSON()` öğesininkine benzer:

```
function (k,v):*
```

`toJSON()` öğesinden farklı olarak, `replacer` işlevi anahtarın (`k`) yanı sıra değeri de (`v`) gerektirir. Bu farklılık, `stringify()` öğesi, kodlanmakta olan nesne yerine statik JSON nesnesinde arandığı için gereklidir. `JSON.stringify()` öğesi, `replacer(k,v)` öğesini çağırdığında orijinal girdi nesnesini çapraz geçiyordur. `replacer` işlevine iletilen örtük `this` parametresi, anahtarı ve değeri tutan nesne ile ilgilidir. `JSON.stringify()` öğesi, orijinal girdi nesnesini değiştirmediği için bu nesne kapsayıcıda çapraz geçirilerek değiştirilmemiş olarak kalır. Bu nedenle, orijinal nesnedeki anahtarı sorgulamak için `this[k]` kodunu kullanabilirsiniz. `v` parametresi, `toJSON()` öğesinin dönüştürdüğü değeri tutar.

`toJSON()` gibi, `replacer` işlevi de herhangi bir değer türünü döndürebilir. `replacer` öğesi bir dize döndürürse JSON motoru, tırnak içindeki içeriklerden kurtulur ve ardından kurtulan içerikleri tırnak içine kaydırır. Bu kaydırma işlemi, `stringify()` öğesinin, `JSON.parse()` öğesini sonraki çağırışında bir dize olarak kalan geçerli bir JSON dizesi nesnesini almasını garantiler.

Aşağıdaki kod, `replacer` parametresi ile örtük `this` parametresini, Date nesnesinin `time` ve `hours` değerlerini döndürmek için kullanır:

```
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;
});
```

JSON.parse() reviver parametresini kullanma

`JSON.parse()` yönteminin `reviver` parametresi, `replacer` işlevinin tersini yapar: JSON dizesini kullanılabilir bir ActionScript nesnesine dönüştürür. `reviver` bağımsız değişkeni, iki parametre alan ve herhangi bir tür döndüren bir işlevdir:

```
function (k,v):*
```

Bu işlevde, `k` öğesi bir anahtardır ve `v` öğesi, `k` öğesinin değeridir. `stringify()` öğesi gibi `parse()` öğesi, JSON anahtar-değer çiftlerini çapraz geçer ve varsa, `reviver` işlevini her bir çifte uygular. Olası bir problem, JSON sınıfının, bir nesnenin ActionScript sınıfı adının çıktısını almamasıdır. Bu nedenle, ne tür bir nesnenin kullanıma geçirileceğini bilmek zorlu olabilir. Bu problem özellikle nesneler iç içe geçmiş olduğu zaman sorun olabilir. `toJSON()`, `replacer` ve `reviver` işlevlerinin tasarlanması sırasında, orijinal nesneleri bozulmadan tutarken dışa aktarılan ActionScript nesnelerini tanımlamak için yollar bulabilirsiniz.

Ayrıştırma örneği

Aşağıdaki örnek, JSON dizelerinden ayrıştırılan nesneleri kullanıma geçirmek için bir strateji göstermektedir. Bu örnek iki sınıfı tanımlar: JSONGenericDictExample ve JSONDictionaryExtnExample. JSONGenericDictExample sınıfı özel bir sözlük sınıfıdır. Her kayıt, bir kişinin eşsiz kimliğinin yanı sıra adını ve doğum tarihini içerir. JSONGenericDictExample yapıcısı her çağrıldığında yeni oluşturulan nesneyi, kimlik olarak statik olarak artan bir tamsayı ile dahili bir statik dizeye ekler. JSONGenericDictExample sınıfı, daha uzun olan `id` üyesinden sadece tamsayı kısmını ayıklayan `revive()` yöntemini tanımlar. `revive()` yöntemi bu tamsayıyı, kullanıma geçirilebilir doğru nesneyi arayıp bulması ve döndürmesi için kullanır.

JSONGenericDictExample sınıfı, ActionScript Dictionary sınıfını genişletir. Kayıtları küme yapısına sahip değildir ve herhangi bir veriyi içerebilir. Veri, sınıf tanımlı özellikler olarak atanmak yerine, JSONDictionaryExtnExample nesnesi oluşturulduktan sonra atanır. JSONDictionaryExtnExample kayıtları anahtar olarak JSONGenericDictExample nesnelerini kullanır. Bir JSONDictionaryExtnExample nesnesi kullanıma geçirildiğinde `JSONGenericDictExample.revive()` işlevi, doğru anahtar nesnesini almak için JSONDictionaryExtnExample ile ilişkili kimliği kullanır.

En önemlisi, `JSONDictionaryExtnExample.toJSON()` yönteminin, JSONDictionaryExtnExample nesnesine ek olarak bir işaretleyici dize döndürmesidir. Bu dize, JSON çıktısını JSONDictionaryExtnExample sınıfına aitmiş gibi tanımlar. Bu işaretleyici, `JSON.parse()` sırasında hangi nesne türünün işlendiği konusunda hiçbir şüphe bırakmaz.

```
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;
        }
    }
}
```

Aşağıdaki çalışma zamanı komut dosyası JSONDictionaryExtnExample nesnesinde `JSON.parse()` öğesini çağırdığında, `reviver` işlevi JSONDictionaryExtnExample öğesindeki her nesnede `JSONGenericDictExample.revive()` öğesini çağırır. Bu çağrı nesne anahtarını temsil eden kimliği ayıklar. `JSONGenericDictExample.revive()` işlevi saklanan JSONDictionaryExtnExample nesnesini özel statik bir diziden almak ve döndürmek için bu kimliği kullanır.

```
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]>]
```

Bölüm 8: Olayları işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Olay işleme sistemi, programcıların kullanıcı girişlerine ve sistem olaylarına pratik bir şekilde yanıt vermesini sağlar. ActionScript 3.0 olay modeli yalnızca pratik olmakla kalmayıp, aynı zamanda standart uyumlu ve görüntüleme listesiyle entegredir. Yeni olay modeli, endüstri standardı olay işleme mimarisi niteliğindeki Belge Nesnesi Modeli (DOM) Düzey 3 Olaylar Belirtimi'ni esas alarak ActionScript programcıları için güçlü ve aynı zamanda sezgisel bir olay işleme aracı sağlar.

ActionScript 3.0 olay işleme sistemi görüntüleme listesiyle yakından etkileşim içindedir. Görüntüleme listesiyle ilgili temel bir fikir edinmek için, "Görüntü programlama" sayfa 153 bölümünü okuyun.

Daha fazla Yardım konusu

flash.events paketi

Belge Nesnesi Modeli (DOM) Düzey 3 Olaylar Tanımlaması

Olay işleme temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Olayları, SWF dosyanızdaki, bir programcı olarak ilginizi çeken herhangi türde bir oluşum olarak değerlendirebilirsiniz. Örneğin, çoğu SWF dosyası, ister fare tıklatmasına yanıt vermek gibi basit, ister bir forma girilen verileri kabul etme ve işleme gibi karmaşık olsun, bir şekilde kullanıcı etkileşimini destekler. Bu şekilde SWF dosyanızla gerçekleştirilen herhangi bir kullanıcı etkileşimi olay olarak değerlendirilir. Olaylar herhangi bir doğrudan kullanıcı etkileşimi olmadan da gerçekleşebilir, örn. bir sunucudan veri yüklemesi sona erdiğinde veya takılan bir kamera etkin olduğunda.

ActionScript 3.0'da her olay, Event sınıfının veya alt sınıflarından birinin örneği olan bir olay nesnesi tarafından temsil edilir. Bir olay nesnesi hem belirli bir olayla ilgili bilgileri saklar, hem de olay nesnesinin işlenmesini kolaylaştıran yöntemler içerir. Örneğin, Flash Player veya AIR bir fare tıklatması algıladığında, bu belirli fare tıklatması olayını temsil etmek için bir olay nesnesi (MouseEvent sınıfının bir örneği) oluşturur.

Flash Player veya AIR, bir olay nesnesi oluşturduktan sonra bu olay nesnesini *gönderir*, başka bir deyişle, olay nesnesi, olayın hedefi olan nesneye iletilir. Gönderilen bir olay nesnesinin hedefi görevini gören bir nesne *olay hedefi* olarak adlandırılır. Örneğin, takılı bir kamera etkin olduğunda, Flash Player doğrudan olay hedefine (bu durumda kamerayı temsil eden nesne) bir olay nesnesi gönderir. Ancak olay hedefi, görüntüleme listesindeyse, olay nesnesi olay hedefine ulaşıncaya kadar görüntüleme listesi hiyerarşisinden aşağı doğru iletilir. Bazı durumlarda, olay nesnesi görüntüleme listesi hiyerarşisinde aynı rota boyunca geri yukarı doğru "köpürür". Görüntü listesi hiyerarşisindeki bu çapraz geçiş, *olay akışı* olarak adlandırılır.

Olay dinleyicilerini kullanarak kodunuzda olay nesnelerini "dinleyebilirsiniz". *Olay dinleyicileri*, belirli olaylara yanıt olarak yazdığınız işlevler veya yöntemlerdir. Programınızın olaylara yanıt vermesini sağlamak için, olay hedefine veya olay nesnesinin olay akışının bir parçası olan herhangi bir görüntüleme listesi nesnesine olay dinleyicileri eklemeniz gerekir.

Her olay dinleyicisi kodu yazdığınızda, bu kod temel yapıyı takip eder (kalın yazı karakterine sahip öğeler, kendi özel durumunuz için dolduracağınız yer tutuculardır):

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

Bu kod iki işlem gerçekleştirir. İlk olarak, olaya yanıt olarak gerçekleştirilecek eylemleri belirtmenin bir yolu olan bir işlevi tanımlar. Daha sonra, işlevin belirtilen olaya abone olmasının sonucu olarak bu, kaynak nesnenin `addEventListener()` yöntemini çağırır, böylece olay gerçekleştiğinde işlevin eylemleri gerçekleştirilir. Gerçekten olay gerçekleştiğinde, olay hedefi, olay dinleyicileri olarak kaydedilen tüm işlevlerin ve yöntemlerin listesini kontrol eder. Ardından, olay nesnesini bir parametre olarak ileterek sırayla her birini çağırır.

Kendi olay dinleyicinizi oluşturmak için bu kodda dört şeyi değiştirmeniz gerekir. İlk olarak, işlevin adını, kullanmak istediğiniz adla değiştirmeniz gerekir. (Kodda **eventResponse** geçen iki yerde bunun değiştirilmesi gerekir.) İkinci olarak, dinlemek istediğiniz olay (kodda **EventType**) tarafından gönderilen olay nesnesinin uygun sınıf adını belirtmeniz ve bu belirli olay (listede **EVENT_NAME**) için uygun sabiti belirtmeniz gerekir. Üçüncü olarak, olayı gönderen (bu kodda **eventTarget**) nesnede, `addEventListener()` yöntemini çağırmanız gerekir. İsteğe bağlı olarak, işlevin parametresi (bu kodda **eventObject**) olarak kullanılan değişkenin adını değiştirebilirsiniz.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi olay işleme grafiklerini programlarken karşılaşacağınız önemli terimleri içerir:

Köpürme Köpürme, bir üst görüntüleme nesnesinin alt nesneleri tarafından gönderilen olaylara yanıt verebilmesi için meydana gelir.

Köpürme aşaması Olay akışında bir olayın üst görüntüleme nesnelerine yayıldığı kısım. Köpürme aşaması yakalama ve hedef aşamalarından sonra meydana gelir.

Yakalama aşaması Olay akışında olayın en genel hedeften en belirli hedef nesneye yayıldığı kısım. Yakalama aşaması hedef ve köpürme aşamalarından önce meydana gelir.

Varsayılan davranış Bazı olaylar, normalde olayla birlikte gerçekleşen ve varsayılan davranış adı verilen bir davranış içerir. Örneğin, kullanıcı bir metin alanına yazı yazdığında, bir metin girdisi olayı ortaya çıkar. Bu olay için varsayılan davranış, metin alanına yazılan karakterin gerçekten görüntülenmesidir—ancak bu varsayılan davranışı geçersiz kılabilirsiniz (herhangi bir nedenle yazılan karakterin görüntülenmesini istemiyorsanız).

Gönderme Olay dinleyicilerine bir olayın gerçekleştiğini bildirme.

Olay Nesnede gerçekleşen ve nesnenin diğer nesnelere hakkında bilgi verebileceği bir şey.

Olay akışı Görüntüleme listesindeki (ekranda görüntülenen bir nesne) bir nesnede olaylar gerçekleştiğinde, söz konusu nesneyi içeren tüm nesnelere olay bildirilir ve bu nesneler de karşılık olarak olay dinleyicilerine bildirim gönderir. Bu işlem Sahne Alanı ile başlar ve görüntüleme listesi üzerinden, olayın gerçekleştiği gerçek nesneye doğru ilerler, ardından tekrar Sahne Alanı'na doğru geri ilerler. Bu işlem olay akışı olarak bilinir.

Olay nesnesi Belirli bir olayın meydana gelmesi hakkında bilgi içeren bir nesne olup olay gönderildiğinde tüm dinleyicilere gönderilir.

Olay hedefi Gerçekten bir olay gönderen nesne. Örneğin, kullanıcı Sprite içinde ve dolayısıyla Sahne Alanı içinde de bulunan bir düğmeyi tıklatırsa, bu nesnelerin tümü olaylar gönderir ancak olay hedefi (bu durumda tıklatılan düğme), olayın gerçekten gerçekleştiği yerdir.

Dinleyici Belirli bir olay gerçekleştiğinde bunun bildirilmesi gerektiğini belirtmek için bir nesneyle kendini kaydeden bir nesne veya işlev.

Hedef aşaması Olay akışında olayın olabilecek en belirli hedefe ulaştığı nokta. Hedef aşaması yakalama ve köpürme aşamalarının arasında meydana gelir.

ActionScript 3.0 olay işleme, önceki sürümlerden nasıl farklılık gösterir

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da olay işleme ile önceki ActionScript sürümlerinde olay işleme arasındaki en belirgin farklılık, önceki ActionScript sürümlerinde birçok farklı olay işleme sistemleri bulunurken, ActionScript 3.0'da olay işleme için yalnızca bir sistem bulunmasıdır. Bu bölüm, önceki ActionScript sürümlerinde olay işlemenin nasıl çalıştığına genel bakış ile başlar ve sonra ActionScript 3.0 için olay işlemenin nasıl değiştiğini ele alır.

Önceki ActionScript sürümlerinde olay işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0 öncesindeki ActionScript sürümleri, olayların işlenmesine yönelik birçok farklı yöntem sağlamıştır:

- Doğrudan Button ve MovieClip örnekleri üzerinde yerleştirilebilen `on()` olay işleyicileri
- Doğrudan MovieClip örnekleri üzerine yerleştirilebilen `onClipEvent()` işleyicileri
- `XML.onload` ve `Camera.onActivity` gibi geri çağrı işlevi özellikleri
- `addListener()` yöntemi kullanarak kaydettiğiniz olay dinleyicileri
- Kısmı olarak DOM olay modelini uygulayan UIEventDispatcher sınıfı.

Bu mekanizmaların her biri, kendi avantaj ve sınırlama kümesini sunar. `on()` ve `onClipEvent()` işleyicilerinin kullanımı kolaydır ancak doğrudan düğmelerin ve film kliplerinin üzerine yerleştirilen kodun bulunması güç olduğundan, bu işleyiciler projelerin sonraki bakımını daha güç hale getirir. Geri çağrı işlevleri kolay uygulanabilir ancak sizi herhangi bir belirli olay için yalnızca bir geri çağrı işleviyle sınırlandırır. Olay dinleyicileri daha zor uygulanabilir, yalnızca bir dinleyici nesnesi ve işlevinin oluşturulmasını değil aynı zamanda olayı oluşturan nesneyle dinleyicinin kaydını da gerektirir. Ancak bu yüksek yük, birçok dinleyici nesnesi oluşturmanıza ve bunların tümünü aynı olay için kaydetmenize olanak sağlar.

ActionScript 2.0 bileşenlerinin geliştirilmesi de başka bir olay modeli oluşturmuştur. UIEventDispatcher sınıfına dahil edilen bu yeni model, DOM Olayları Belirtimi'nin bir alt kümesini esas almıştır. Bileşen olayı işleme konusunda bilgi sahibi olan geliştiriciler, yeni ActionScript 3.0 olay modeline geçiş sürecinde daha az zorluk yaşayacaktır.

Ne yazık ki, çeşitli olay modelleri tarafından kullanılan sözdizimi çeşitli şekillerde örtüşür ve çeşitli şekillerde farklılık gösterir. Örneğin, ActionScript 2.0'da `TextField.onChanged` gibi bazı özellikler, bir geri çağrı işlevi veya olay dinleyicisi olarak kullanılabilir. Ancak, dinleyici nesnelerini kaydetmeye yönelik sözdizimi, altı sınıf arasından, dinleyicileri mi yoksa UIEventDispatcher sınıfını mı destekleyeni kullandığınıza bağlı olarak farklılık gösterir. Key, Mouse, MovieClipLoader, Selection, Stage ve TextField sınıfları için, `addListener()` yöntemini kullanırsınız ancak bileşen olay işlemesi için `addEventListener()` adında bir yöntem kullanırsınız.

Farklı olay işleme modellerinin sunduğu başka bir karmaşıklık da, kullanılan mekanizmaya bağlı olarak geniş ölçüde değişiklik gösteren olay işleyici işlevinin kapsamıydı. başka bir deyişle, `this` anahtar sözcüğünün anlamı olay işleme sistemleri arasında tutarlı değildi.

ActionScript 3.0'da olay işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, önceki dil sürümlerinde varolan birçok farklı olay işleme mekanizmalarının yerini alan tek bir olay işleme modelini sunar. Yeni olay modeli, Belge Nesnesi Modeli (DOM) Düzey 3 Olaylar Belirtimi'ni esas alır. SWF dosya biçimi özellikle Belge Nesnesi Modeli standardına bağlı olmasa da, DOM olay modelinin uygulanmasının mümkün olması için görüntüleme listesi ile DOM yapısı arasında yeterince benzerlik vardır. Görüntüleme listesindeki bir nesne, DOM hiyerarşik yapısındaki bir düğüme benzetilebilir ve *görüntüleme listesi nesnesi* ile *düğüm* terimleri de bu açıklama içinde birbirinin yerine kullanılabilir.

DOM olay modelinin Flash Player ve AIR uygulaması, varsayılan davranışlar adında bir kavram içerir. *Varsayılan davranış*, Flash Player veya AIR uygulamasının, belirli olayların normal sonucu olarak çalıştırdığı bir eylemdir.

Varsayılan davranışlar

Geliştiriciler genellikle olaylara yanıt veren kodlar yazmaktan sorumludur. Ancak bazı durumlarda bir davranış bir olayla o kadar yaygın şekilde ilişkilendirilmiştir ki, geliştirici davranışı iptal etmek için kod eklemediği sürece Flash Player veya AIR uygulaması davranışı otomatik olarak çalıştırır. Flash Player veya AIR otomatik olarak davranış sergilediğinden, bu davranışlar varsayılan davranışlar olarak bilinir.

Örneğin, kullanıcı bir TextField nesnesine metin girdiğinde, metnin TextField nesnesinde görüntülenmesi o kadar yaygındır ki, Flash Player ve AIR uygulamasında bu davranış yerleşik olarak bulunur. Bu varsayılan davranışın gerçekleşmesini istemiyorsanız, yeni olay işleme sistemini kullanarak bu davranışı iptal edebilirsiniz. Kullanıcı bir TextField nesnesine metin girdiğinde, Flash Player veya AIR, kullanıcı girdisini temsil etmek için TextEvent sınıfının bir örneğini oluşturur. Flash Player veya AIR uygulamasının, TextField nesnesinde metni görüntülemesini önlemek için, bu belirli TextEvent örneğine erişmeniz ve örneğin `preventDefault()` yöntemini çağırmanız gerekir.

Tüm varsayılan davranışlar engellenemez. Örneğin, kullanıcı TextField nesnesinde bir sözcüğü çift tıklattığında, Flash Player ve AIR bir MouseEvent nesnesi oluşturur. Bu önlenemeyen varsayılan davranış, imlecin altındaki sözcüğün vurgulanmasıdır.

Olay nesnelerinin birçok türü, ilişkilendirilmiş varsayılan davranışlara sahip değildir. Örneğin, bir ağ bağlantısı kurulduğunda Flash Player bir bağlanma olay nesnesi gönderir ancak bununla ilişkilendirilmiş bir varsayılan davranış yoktur. Event sınıfı ve alt sınıflarının API belgeleri her olay türünü listeler ve engellenebilir olsun veya olmasın tüm ilişkilendirilmiş davranışları açıklar.

Varsayılan davranışların yalnızca Flash Player veya AIR tarafından gönderilen olay nesneleriyle ilişkilendirildiğinin ve ActionScript üzerinden programlama yoluyla gönderilen olay nesneleri için var olmadığının anlaşılması önemlidir. Örneğin, `textInput` türünde bir olay nesnesi göndermek için EventDispatcher sınıfının yöntemlerini kullanabilirsiniz ancak bu olay nesnesi kendisiyle ilişkilendirilmiş varsayılan bir davranışa sahip olmaz. Başka bir deyişle, Flash Player ve AIR, programlama yoluyla gönderdiğiniz `textInput` olayının sonucunda TextField nesnesinde bir karakter görüntülemez.

ActionScript 3.0'daki olay dinleyicilerine yönelik yenilikler

ActionScript 2.0 `addListener()` yöntemini kullanma konusunda deneyimli geliştiriciler için, ActionScript 2.0 olay dinleyicisi modeli ile ActionScript 3.0 olay modeli arasındaki farklılıkların belirtilmesi yardımcı olacaktır. Aşağıdaki listede, iki olay modeli arasındaki önemli farklılıklar açıklanmaktadır:

- ActionScript 2.0'da olay dinleyicileri eklemek için, bazı durumlarda `addListener()` yöntemini ve bazı durumlarda da `addEventListener()` yöntemini kullanırken, ActionScript 3.0'da tüm durumlarda `addEventListener()` yöntemini kullanırsınız.
- ActionScript 2.0'da herhangi bir olay akışı yoktur, başka bir deyişle, `addListener()` yöntemi yalnızca olayı yayınlayan nesnede çağrılabilir, ActionScript 3.0'da ise olay akışının parçası olan herhangi bir nesnede `addEventListener()` yöntemi çağrılabilir.
- ActionScript 2.0'da, işlevler, yöntemler veya nesneler olay dinleyicileri olabilirken, ActionScript 3.0'da yalnızca işlevler ya da yöntemler olay dinleyicileri olabilir.

Olay akışı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

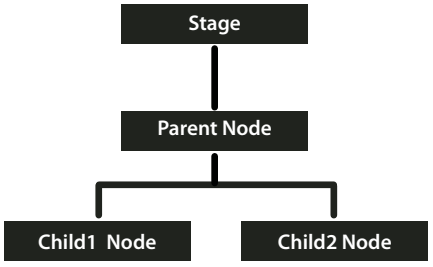
Flash Player veya AIR, her olay gerçekleştiğinde olay nesneleri gönderir. Olay hedefi görüntüleme listesinde bulunmuyorsa, Flash Player veya AIR doğrudan olay hedefine olay nesnesini gönderir. Örneğin, Flash Player doğrudan bir URLStream nesnesine ilerleme olay nesnesini gönderir. Ancak olay hedefi görüntüleme listesinde bulunuyorsa, Flash Player, görüntüleme listesine olay nesnesini gönderir ve olay nesnesi, görüntüleme listesi üzerinden olay hedefine doğru seyahat eder.

Olay akışı, bir olay nesnesinin görüntüleme listesi üzerinden nasıl hareket ettiğini açıklar. Görüntüleme listesi, ağaç olarak tanımlanabilen bir hiyerarşide organize edilir. Görüntüleme listesi hiyerarşisinin en üstünde Sahne Alanı yer alır, bu, görüntüleme listesinin kökü görevini gören özel bir görüntüleme nesnesi kabıdır. Sahne Alanı, flash.display.Stage sınıfı tarafından temsil edilir ve yalnızca bir görüntüleme nesnesi üzerinden Sahne Alanı'na erişilebilir. Her görüntüleme nesnesi, söz konusu uygulamanın Sahne Alanı'nı ifade eden `stage` adında bir özelliğe sahiptir

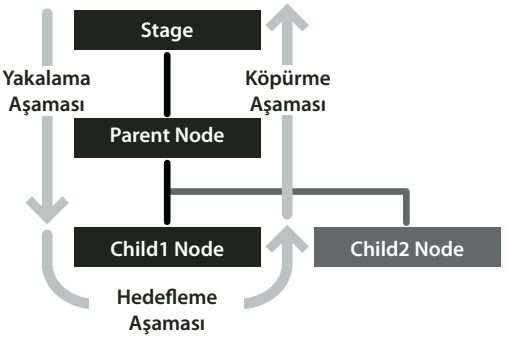
Flash Player veya AIR, görüntüleme listesiyle ilgili bir olay için bir olay nesnesi gönderir, bu olay nesnesi, Sahne Alanı'ndan *hedef düğüme* gidişli dönüşlü bir seyahatte bulunur. DOM Olayları Belirtimi, hedef düğümü olay hedefini temsil eden düğüm olarak tanımlar. Başka bir deyişle, hedef düğüm, olayın gerçekleştiği görüntüleme listesi nesnesidir. Örneğin, kullanıcı `child1` adında bir görüntüleme listesi nesnesini tıklatırsa, Flash Player veya AIR uygulaması hedef düğüm olarak `child1` öğesini kullanarak bir olay nesnesi gönderir.

Olay akışı kavramsal olarak üç bölüme ayrılmıştır. Birinci bölüm yakalama aşaması olarak adlandırılır; bu aşama Sahne Alanı'ndan hedef düğümün üst öğesine kadar tüm düğümleri kapsar. İkinci bölüm hedef aşaması olarak adlandırılıp yalnızca hedef düğümü içerir. Üçüncü bölüm ise köpürme aşaması olarak adlandırılır. Köpürme aşaması, hedef düğümün üst öğesinden geri Sahne Alanı'na doğru dönüş yolculuğunda karşılaşılan düğümleri kapsar.

Aşağıdaki diyagramda gösterildiği gibi, görüntüleme listesini, Sahne Alanı'nın en üstte bulunduğu dikey bir hiyerarşi olarak algılarsanız, aşamaların adları daha anlamlı gelecektir:



Kullanıcı `Child1 Node` üzerini tıklatırsa, Flash Player veya AIR uygulaması olay akışına bir olay nesnesi gönderir. Aşağıdaki görüntüde gösterildiği gibi, nesnenin seyahati `Stage` öğesinde başlar, `Parent Node` öğesinden aşağı devam ederek `Child1 Node` öğesine ilerler ve sonra geri `Stage` öğesine doğru "köpürerek" `Stage` öğesine doğru seyahati sırasında tekrar `Parent Node` üzerinden hareket eder.



Bu örnekte, yakalama aşaması, ilk aşağı doğru seyahat sırasında `Stage` ve `Parent Node` öğelerini kapsar. Hedef aşaması, `Child1 Node` üzerinde harcanan süreyi kapsar. Köpürme aşaması, geri kök düğüme doğru yukarı yönde yapılan seyahat sırasında karşılaşılan `Parent Node` ve `Stage` öğelerini kapsar.

Olay akışı, ActionScript programcıları için önceden kullanılabilir olan olay işleme sisteminden daha güçlü bir olay işleme sistemi elde edilmesine katkıda bulunur. Önceki ActionScript sürümlerinde olay akışı yoktur, başka bir deyişle, olay dinleyicileri yalnızca olayı oluşturan nesneye eklenebilir. ActionScript 3.0'da, yalnızca hedef düğüme değil, aynı zamanda olay akışındaki herhangi bir düğüme de olay dinleyicileri ekleyebilirsiniz.

Bir kullanıcı arabirimi bileşeni birden çok nesneyi kapsadığında, olay akışı sırasında olay dinleyicileri ekleme yeteneği yararlı olur. Örneğin, bir düğme nesnesi genellikle düğmenin etiketi görevini gören bir metin nesnesi içerir. Olay akışına dinleyici ekleme yeteneği olmadığında, düğmenin herhangi bir yerinde oluşan tıklatma olayları hakkında bildirim almak için hem düğme nesnesine hem de metin nesnesine dinleyici eklemeniz gerekir. Olay akışının olması ise, düğme nesnesine, hem metin nesnesi üzerinde hem de düğme nesnesinin metin nesnesinin kaplamadığı alanları üzerinde gerçekleşen tıklatma olaylarını işleyen tek bir olay dinleyicisi yerleştirmenize olanak sağlar.

Ancak her olay nesnesi, olay akışının üç aşamasına da katılmaz. `enterFrame` ve `init` gibi bazı olay türleri, doğrudan hedef düğüme gönderilir ve yakalama aşamasına ve köpürme aşamasına katılmaz. Başka olaylar da, Socket sınıfının bir örneğine gönderilen olaylar gibi görüntüleme listesinde bulunmayan nesneleri hedefleyebilir. Bu olay nesneleri de yakalama ve köpürme aşamalarına katılmadan doğrudan hedef nesneye akar.

Belirli olay türlerinin nasıl davrandığını öğrenmek için, API belgelerini kontrol edebilir veya olay nesnesinin özelliklerini inceleyebilirsiniz. Olay nesnesinin özelliklerinin incelenmesi, aşağıdaki bölümde açıklanmıştır.

Olay nesneleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Olay nesneleri, yeni olay işleme sisteminde iki ana amaca hizmet eder. Birincisi, olay nesneleri bir özellik kümesindeki belirli özellikler hakkındaki bilgileri saklayarak gerçek olayları temsil eder. İkincisi, olay nesneleri, olay nesnelerini işlemenize izin veren ve olay işleme sisteminin davranışını etkileyen bir özellik kümesi içerir.

Bu özellik ve yöntemlere erişilmesini kolaylaştırmak için, Flash Player API'si, tüm olay nesneleri için temel sınıf görevi gören bir Event sınıfını tanımlar. Event sınıfı, tüm olay nesneleri için ortak olan ana bir özellik ve yöntem kümesi tanımlar.

Bu bölüm Event sınıfı özelliklerinin ele alınmasıyla başlar, Event sınıfı yöntemlerinin açıklamasıyla devam eder ve neden Event sınıfının alt sınıflarının var olduğuna dair bir açıklamayla sona erer.

Event sınıfının özelliklerini anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Event sınıfı, bir olay nesnesi hakkında önemli bilgiler sağlayan birçok salt okunur özellik ve sabitleri tanımlar. Özellikle şunlar çok önemlidir:

- Olay nesnesi türleri, sabitler tarafından temsil edilir ve `Event.type` özelliğinde saklanır.
- Bir olayın varsayılan davranışının önlenip önlenemeyeceği, bir Boolean değeriyle temsil edilir ve `Event.cancelable` özelliğinde saklanır.
- Olay akışı bilgileri, kalan özelliklerde bulunur.

Olay nesnesi türleri

Her olay nesnesinin ilişkilendirilmiş bir olay türü vardır. Olay türleri dize değerleri olarak `Event.type` özelliğinde saklanır. Bir olay nesnesinin türünü bilmekte yarar vardır, böylece kodunuz farklı türlerdeki nesneleri ayırt edebilir. Örneğin, aşağıdaki kod, `clickHandler()` dinleyici işlevinin, `myDisplayObject` öğesine iletilen herhangi bir fare tıklatma olay nesnelerini yanıtlaması gerektiğini belirtir:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

Event sınıfı ile yirmi olay türü ilişkilendirilmiş olup bunlar Event sınıfı sabitleri tarafından temsil edilir ve bazıları aşağıdaki Event sınıfı tanımının alıntısında gösterilmektedir:

```
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
    }
}
```

Bu sabitler, belirli olay türlerini ifade etmenin kolay bir yolunu sağlar. Temsil ettikleri dizeleri kullanmak yerine bu sabitleri kullanmanız gerekir. Kodunuzda bir sabit adını yanlış yazarsanız, derleyici yanlışı yakalar ancak bunun yerine dizeleri kullanırsanız, derleme zamanında yazım hatası bildirilmeyebilir ve bu da hata ayıklaması yapılması zor olan beklenmeyen bir davranışa yol açabilir. Örneğin, bir olay dinleyicisi eklerken şu kodu kullanın:

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

bunun yerine:

```
myDisplayObject.addEventListener("click", clickHandler);
```

Varsayılan davranış bilgileri

Kodunuz belirli bir olay nesnesi için varsayılan davranışın engellenip engellenemeyeceğini `cancelable` özelliğine erişerek kontrol edebilir. `cancelable` özelliği, varsayılan bir davranışın önlenip önlenemeyeceğini belirten bir Boolean değerini içerir. `preventDefault()` yöntemini kullanarak az sayıda olayla ilişkilendirilmiş varsayılan bir davranışı engelleyebilir veya iptal edebilirsiniz. Daha fazla bilgi için "Event sınıfının yöntemlerini anlama" sayfa 134 altındaki Varsayılan olay davranışını iptal etme konusuna bakın.

Olay akışı bilgileri

Kalan Event sınıfı özellikleri, aşağıdaki listede açıklandığı gibi, olay nesnesi ve olay nesnesinin olay akışıyla ilişkisi hakkında önemli bilgiler içerir:

- `bubbles` özelliği, olay nesnesinin katıldığı olay akışının bölümleri hakkında bilgiler içerir.
- `eventPhase` özelliği, olay akışındaki geçerli aşamayı belirtir.
- `target` özelliği, olay hedefine bir başvuruyu saklar.
- `currentTarget` özelliği, geçerli olarak olay nesnesini işleyen görüntüleme listesi nesnesine bir başvuruyu saklar.

bubbles özelliği

Bir olayın olay nesnesi, olay akışının köpürme aşamasına katılıyorsa, başka bir deyişle, olay nesnesi, Sahne Alanı'na ulaşıncaya kadar üst öğeleri üzerinden hedef düğümden geri iletiliyorsa, o olayın köpürdüğü söylenebilir. `Event.bubbles` özelliği, olay nesnesinin köpürme aşamasına katılıp katılmadığını belirten bir Boolean değerini saklar. Köpüren tüm olaylar ayrıca yakalama ve hedef aşamalarına da katıldığından, köpüren tüm olaylar, olay akışı aşamalarının üçüne de katılır. Değer `true` olursa, olay nesnesi üç aşamanın tamamına katılır. Değer `false` olursa, olay nesnesi köpürme aşamasına katılmaz.

eventPhase özelliği

`eventPhase` özelliğini inceleyerek herhangi bir olay nesnesinin olay aşamasını belirleyebilirsiniz. `eventPhase` özelliği, olay akışının üç aşamasından birini temsil eden işaretsiz bir tam sayı değeri içerir. Flash Player API'si, aşağıdaki kod alıntısında gösterildiği gibi, üç işaretsiz tam sayı değerine karşılık gelen üç sabiti içeren ayrı bir EventPhase sınıfını tanımlar:

```
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;
    }
}
```

Bu sabitler, `eventPhase` özelliğinin üç geçerli değerine karşılık gelir. Kodunuzu daha okunaklı hale getirmek için bu sabitleri kullanabilirsiniz. Örneğin, yalnızca olay hedefi, hedef sahne alanı üzerinde olduğunda `myFunc()` adında bir işlevin çağrılmasını sağlamak istiyorsanız, bu koşulu test etmek için şu kodu kullanabilirsiniz:

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

target özelliği

`target` özelliği, olayın hedefi olan nesneye bir başvuru içerir. Bazı durumlarda bu olabildiğince nettir, örneğin, mikrofon etkin hale geldiğinde, olay nesnesinin hedefi Microphone nesnesidir. Ancak hedef, görüntüleme listesindeyse, görüntüleme listesi hiyerarşisinin dikkate alınması gerekir. Örneğin, kullanıcı, örtüşen görüntüleme listesi nesneleri içeren bir noktaya fare tıklatması girerse, Flash Player ve AIR uygulaması, olay hedefi olarak her zaman Sahne Alanı'nın en uzağında bulunan nesneyi seçer.

Karmaşık SWF dosyaları için, özellikle de içindeki düğmelerin rutin olarak daha küçük alt nesnelerle süslendiği SWF dosyaları için, `target` özelliği genellikle düğme yerine düğmenin alt nesnesini işaret ettiği için, bu özellik sık kullanılmaz. Bu durumlarda en yaygın uygulama, düğmeye olay dinleyicileri eklenmesi ve `target` özelliği düğmenin alt öğesini işaret edebilirken `currentTarget` özelliği düğmeyi işaret ettiğinden bu ikinci özelliğin kullanılmasıdır.

currentTarget özelliği

`currentTarget` özelliği, geçerli olarak olay nesnesini işleyen nesneye bir başvuruyu saklar. İncelediğiniz olay nesnesini geçerli olarak hangi düğümün işlediğinin bilinmesi garip görünse de, o olay nesnesinin olay akışında herhangi bir görüntüleme nesnesine dinleyici işlevi ekleyebileceğinizi ve dinleyici işlevinin herhangi bir konuma yerleştirilebileceğini unutmayın. Üstelik aynı dinleyici işlevi farklı görüntüleme nesnelerine eklenebilir. Projenin boyutu ve karmaşıklık düzeyi arttıkça, `currentTarget` özelliği çok daha fazla kullanışlı hale gelir.

Event sınıfının yöntemlerini anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Event sınıfı yöntemleri üç kategoriye ayrılır:

- Bir olay nesnesinin kopyalarını oluşturabilen veya olay nesnesini dizeye dönüştürebilen yardımcı program yöntemleri
- Olay akışından olay nesnelerini kaldıran olay akışı yöntemleri
- Varsayılan davranışı engelleyen veya engellenip engellenmediğini kontrol eden varsayılan davranış yöntemleri

Event sınıfı yardımcı program yöntemleri

Event sınıfında iki yardımcı program yöntemi vardır. `clone()` yöntemi bir olay nesnesinin kopyalarını oluşturmanızı sağlar. `toString()` yöntemi, bir olay nesnesinin değerleriyle beraber özelliklerinin dize olarak temsilini oluşturmanızı sağlar. Bu yöntemlerin ikisi de olay modeli sistemi tarafından dahili olarak kullanılır ancak genel kullanım için geliştiricilere sunulmuştur.

Event sınıfının alt sınıflarını oluşturan ileri düzey geliştiriciler için, olay alt sınıfının düzgün çalışmasını sağlamak için her iki yardımcı program yönteminin sürümlerini geçersiz kılmanız ve uygulamanız gerekir.

Olay akışını durdurma

Bir olay nesnesinin olay akışında yoluna devam etmesini önlemek için `Event.stopPropagation()` yöntemini veya `Event.stopImmediatePropagation()` yöntemini çağırabilirsiniz. Neredeyse aynı olan ve yalnızca geçerli düğümün diğer olay dinleyicilerinin çalıştırılmasına izin verilip verilmediğine göre değişiklik gösteren iki yöntem vardır:

- `Event.stopPropagation()` yöntemi, olay nesnesinin sonraki düğüme hareket etmesini önler ancak yalnızca geçerli düğümdeki diğer olay dinleyicilerinin çalıştırılmasına izin verildikten sonra bunu yapar.

- `Event.stopImmediatePropagation()` yöntemi de, olay nesnesinin sonraki düğüme hareket etmesini önler ancak geçerli düğümdeki diğer olay dinleyicilerinin çalıştırılmasına izin vermez.

Bu yöntemlerden herhangi birinin çağrılması, bir olayla ilişkilendirilmiş varsayılan davranışın gerçekleşip gerçekleşmemesi üzerinde herhangi bir etki yaratmaz. Varsayılan davranışı önlemek için Event sınıfının varsayılan davranış yöntemlerini kullanın.

Varsayılan olay davranışını iptal etme

Varsayılan davranışı iptal etmeyle ilgili iki yöntem, `preventDefault()` ve `isDefaultPrevented()` yöntemleridir. Bir olayla ilişkilendirilen varsayılan davranışı iptal etmek için `preventDefault()` yöntemini çağırın. Bir olay nesnesinde `preventDefault()` öğesinin önceden çağrılmış olup olmadığını kontrol etmek için, `isDefaultPrevented()` yöntemini çağırın, bu yöntem önceden çağrılmışsa `true` değeri, aksi takdirde `false` değeri döndürülür.

Yalnızca olayın varsayılan davranışı iptal edilebiliyorsa `preventDefault()` yöntemi çalışır. Bu olay türünün API belgelerine başvurarak veya olay nesnesinin `cancelable` özelliğini incelemek için ActionScript'i kullanarak bu şekilde olup olmadığını kontrol edebilirsiniz.

Varsayılan davranışın iptali, olay akışında olay nesnesinin ilerleyişi üzerinde bir etkiye sahip değildir. Bir olay nesnesini olay akışından kaldırmak için, Event sınıfının olay akışı yöntemlerini kullanın.

Event sınıfının alt sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Birçok olay için, Event sınıfında tanımlanan ortak özellik kümesi yeterlidir. Ancak diğer olaylar, Event sınıfında kullanılabilir özellikler tarafından yakalanamayan benzersiz özelliklere sahiptir. Bu olaylar için, ActionScript 3.0, Event sınıfının birçok alt sınıfını tanımlar.

Her alt sınıf, ilgili olay kategorisi için benzersiz olan ek özellikler ve olay türleri sağlar. Örneğin, fare girdisiyle ilgili olaylar, Event sınıfında tanımlı özellikler tarafından yakalanamayan birçok benzersiz özelliğe sahiptir. MouseEvent sınıfı, fare olayının konumu ve fare olayı sırasında belirli tuşlara basılmış olup olmadığı gibi bilgileri içeren on özellik ekleyerek Event sınıfını genişletir.

Event alt sınıfı ayrıca alt sınıfla ilişkilendirilmiş olay türlerini temsil eden sabitleri de içerir. Örneğin, MouseEvent sınıfı `click`, `doubleClick`, `mouseDown` ve `mouseUp` olay türleri de dahil olmak üzere birçok fare olayı türü için sabitleri tanımlar.

"Olay nesneleri" sayfa 132 altındaki Event sınıfı yardımcı program yöntemleri bölümünde açıklandığı gibi, Event alt sınıfı oluştururken, alt sınıfa özgü işlevsellik sağlamak için `clone()` ve `toString()` yöntemlerini geçersiz kılmanız gerekir.

Olay dinleyicileri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Olay işleyicileri olarak da adlandırılan olay dinleyicileri, Flash Player ve AIR uygulamasının belirli olaylara yanıt olarak çalıştırdığı işlevlerdir. Olay dinleyicisi ekleme, iki adımdan oluşan bir işlemdir. İlk olarak, Flash Player veya AIR uygulamasının olaya yanıt olarak çalıştırması için bir işlev veya sınıf yöntemi oluşturursunuz. Bu bazen dinleyici işlevi veya olay işleyici işlevi olarak adlandırılır. İkinci olarak, dinleyici işlevinizi olayın hedefiyle veya uygun olay akışında bulunan herhangi bir görüntüleme listesi nesnesiyle kaydetmek için `addEventListener()` yöntemini kullanırsınız.

Dinleyici işlevi oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dinleyici işlevlerinin oluşturulması, ActionScript 3.0 olay modelinin DOM olay modelinden değişiklik gösterdiği bir noktadır. DOM olay modelinde, olay dinleyicisi ile dinleyici işlevi arasında net bir ayrım vardır: olay dinleyicisi, EventListener arabirimini uygulayan bir sınıf örneğiyken, dinleyici işlevi, o sınıfın `handleEvent()` adındaki bir yöntemidir. DOM olay modelinde, gerçek dinleyici işlevini değil, dinleyici işlevini içeren sınıf örneğini kaydedersiniz.

ActionScript 3.0 olay modelinde, olay dinleyicisi ile dinleyici işlevi arasında belirgin bir fark yoktur. ActionScript 3.0, bir EventListener arabirimine sahip değildir ve dinleyici işlevleri, sınıfın dışında veya sınıfın parçası olarak tanımlanabilir. Üstelik, dinleyici işlevlerinin mutlaka `handleEvent()` olarak adlandırılması gerekmez—herhangi bir geçerli tanımlayıcı ile adlandırılabilir. ActionScript 3.0'da, gerçek dinleyici işlevinin adını kaydedersiniz.

Bir sınıfın dışında tanımlanan dinleyici işlevi

Şu kod, kırmızı bir kare şeklini görüntüleyen basit bir SWF dosyası oluşturur. Bir sınıfın parçası olmayan `clickHandler()` adında bir dinleyici işlevi, kırmızı kare üzerindeki fare tıklatma olaylarını dinler.

```
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);
}
```

Kullanıcı, kareyi tıklatarak sonuçta elde edilen SWF dosyasıyla etkileşim kurduğunda, Flash Player veya AIR şu iz çıktısını oluşturur:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

Olay nesnesinin `clickHandler()` öğesine bir argüman olarak iletildiğine dikkat edin. Bu, dinleyici işlevinizin olay nesnesini incelemesine olanak sağlar. Bu örnekte, olayın bir tıklatma olayı olduğundan emin olmak için olay nesnesinin `type` özelliğini kullanırsınız.

Bu örnek ayrıca `this` anahtar sözcüğünün değerini de kontrol eder. Bu durumda, `this` öğesi, genel nesneyi temsil eder; işlev herhangi bir özel sınıf veya nesne dışında tanımlandığından bu akıllıcadır.

Sınıf yöntemi olarak tanımlanan dinleyici işlevi

Aşağıdaki örnek, ClickExample sınıfını tanımlayan bir önceki örnekle aynıdır; tek farkı `clickHandler()` işlevinin, ChildSprite sınıfının bir yöntemi olarak tanımlanmasıdır:

```
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);
    }
}
```

Kullanıcı, kırmızı kareyi tıklatarak sonuçta elde edilen SWF dosyasıyla etkileşim kurduğunda, Flash Player veya AIR şu iz çıktısını oluşturur:

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

`this` anahtar sözcüğünün `child` adında ChildSprite örneğini ifade ettiğini unutmayın. Bu, ActionScript 2.0'daki davranışta gerçekleşen bir değişikliktir. ActionScript 2.0'daki bileşenleri kullandıysanız, `UIEventDispatcher.addEventListener()` öğesine bir sınıf yöntemi iletildiğinde, yöntem kapsamının, dinleyici yönteminin tanımlandığı sınıf ile değil, olayı yayınlayan bileşenle sınırlı olduğunu hatırlayabilirsiniz. Başka bir deyişle, ActionScript 2.0'da bu tekniği kullandıysanız, `this` anahtar sözcüğü, ChildSprite örneğini değil, olayı yayınlayan bileşeni ifade eder.

Dinleyici yöntemini içeren diğer sınıf yöntemlerine ve özelliklerine erişemedikleri anlamına geldiğinden, bazı programcılar için bu önemli bir sorundu. Geçici bir çözüm olarak, ActionScript 2.0 programcıları dinleyici yönteminin kapsamını değiştirmek için `mx.util.Delegate` sınıfını kullanabiliyordu. Ancak ActionScript 3.0, `addEventListener()` çağrıldığında bir sınır oluşturduğundan artık buna gerek kalmamıştır. Sonuç olarak, `this` anahtar sözcüğü `child` adındaki ChildSprite örneğini ifade eder ve programcı, ChildSprite sınıfının diğer yöntemlerine ve özelliklerine erişim kazanır.

Kullanılmaması gereken olay dinleyicisi

Atanmış dinleyici işlevini dinamik olarak işaret eden bir özellik ile genel bir nesne oluşturduğunuz üçüncü bir teknik daha vardır ancak bu teknik önerilmez. ActionScript 2.0'da yaygın olarak kullanıldığından bu teknik burada ele alınmıştır, ancak ActionScript 3.0'da bu tekniğin kullanılmaması gerekir. `this` anahtar sözcüğü dinleyici nesne yerine genel nesneyi ifade edeceğinden bu teknik önerilmez.

Aşağıdaki örnek, önceki ClickExample sınıfıyla aynıdır, tek farkı, dinleyici işlevinin `myListenerObj` adında genel bir nesnenin parçası olarak tanımlanmasıdır:

```
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);
}
```

İzleme sonuçları şöyle görünür:

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

`this` öğesinin `myListenerObj` öğesini ifade etmesini ve izleme çıktısının `[object Object]` olmasını beklersiniz ancak bu genel nesneyi ifade eder. `addEventListener()` öğesine argüman olarak dinamik bir özellik adı ilettiğinizde, Flash Player veya AIR uygulaması bir sınır yöntemi oluşturamaz. Bunun nedeni, `listener` parametresi olarak ilettiğiniz şeyin, dinleyici işlevinizin bellek adresinden başka bir şey olmaması ve Flash Player ve AIR uygulamasının, bu bellek adresini `myListenerObj` örneğiyle bağlamasının bir yolu olmamasıdır .

Olay dinleyicilerini yönetme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

IEventDispatcher arabiriminin yöntemlerini kullanarak dinleyici işlevlerinizi yönetebilirsiniz. IEventDispatcher arabirimi, DOM olay modelinin EventTarget arabiriminin ActionScript 3.0 sürümüdür. IEventDispatcher adı, ana amacının olay nesnelerini göndermek olduğunu ima etse de, bu sınıfın yöntemleri, olay nesnelerini kaydetmek, olay dinleyicilerini kontrol etmek ve olay dinleyicilerini kaldırmak için çok daha sık kullanılır. IEventDispatcher arabirimi, aşağıdaki kodda gösterildiği gibi beş yöntemi tanımlar:

```
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;
    }
}
```

Flash Player API'si, olay hedefleri veya olay akışının parçası olabilen tüm sınıflar için temel sınıf görevi gören EventDispatcher sınıfı ile IEventDispatcher arabirimini uygular. Örneğin, DisplayObject sınıfı, EventDispatcher sınıfından miras alır. Başka bir deyişle, görüntüleme listesindeki herhangi bir nesne, IEventDispatcher arabiriminin yöntemlerine erişime sahiptir.

Olay dinleyicileri ekleme

`addEventListener()` yöntemi, IEventDispatcher arabiriminin en etkin öğesidir. Dinleyici işlevlerinizi kaydetmek için bunu kullanırsınız. `type` ve `listener` iki zorunlu parametredir. Olayın türünü belirtmek için `type` parametresini kullanırsınız. Olay gerçekleştiğinde çalıştırılacak dinleyici işlevini belirtmek için `listener` parametresini kullanırsınız. `listener` parametresi bir işlev veya sınıf yöntemine başvuru olabilir.

`listener` parametresini belirtirken parantez kullanmayın. Örneğin, aşağıdaki `addEventListener()` yöntemine yapılan çağrıda, `clickHandler()` işlevi parantez olmadan belirtilmiştir:

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

`addEventListener()` yönteminin `useCapture` parametresi, dinleyicinizin etkin olacağı olay akışı aşamasını denetlemenize olanak sağlar. `useCapture` öğesi `true` değerine ayarlanırsa, olay akışının yakalama aşaması sırasında dinleyiciniz etkin olur. `useCapture` öğesi `false` değerine ayarlanırsa, olay akışının köpürme aşaması sırasında dinleyiciniz etkin olur. Olay akışının tüm aşamaları sırasında bir olayı dinlemek için, bir kez `useCapturetrue` değerine ayarlıyken ve daha sonra da bir kez `useCapturefalse` değerine ayarlıyken `addEventListener()` öğesini iki defa çağırmanız gerekir.

`addEventListener()` yönteminin `priority` parametresi, DOM Düzey 3 olay modelinin resmi bir parçası değildir. Bu, olay dinleyicilerinizi organize etmenizde size daha fazla esneklik sağlamak üzere ActionScript 3.0'a dahil edilmiştir. `addEventListener()` öğesini çağırdığınızda, `priority` parametresi olarak bir tam sayı değeri ileterek o olay dinleyicisi için önceliği ayarlayabilirsiniz. Varsayılan değer 0'dır ancak siz bunu negatif veya pozitif tam sayı değerlerine ayarlayabilirsiniz. Sayı ne kadar yükselirse, olay dinleyicisi o kadar önce çalıştırılır. Aynı önceliğe sahip olay dinleyicileri, eklendikleri sırayla çalıştırılır, bu nedenle dinleyici ne kadar önce eklenirse, o kadar önce çalıştırılır.

`useWeakReference` parametresi, dinleyici işlevine başvurunun zayıf mı yoksa normal mi olduğunu belirtmenize olanak sağlar. Bu parametrenin `true` değerine ayarlanması, dinleyici işlevlerinin ihtiyaç duyulmadığı halde bellekte kalmaya devam ettiği durumları önlemenize olanak sağlar. Flash Player ve AIR uygulaması, artık kullanılmayan nesneleri bellekten temizlemek için *çöp toplama* adı verilen bir tekniği kullanır. Bir nesneye herhangi bir başvuru yoksa, o nesne artık kullanılmıyor olarak değerlendirilir. Çöp toplayıcı, zayıf başvuruları yok sayar, başka bir deyişle, bir dinleyici işlevi kendisini işaret eden yalnızca bir zayıf başvuruya sahipse, çöp toplama için uygun demektir.

Olay dinleyicilerini kaldırma

Artık ihtiyacınız olmayan bir olay dinleyicisini kaldırmak için `removeEventListener()` yöntemini kullanabilirsiniz. Artık kullanılmayacak olan olay dinleyicilerinin tümünü kaldırmak yararlı olacaktır. Gerekli parametreler `eventName` ve `listener` parametrelerini içerir, bunlar `addEventListener()` yöntemi için gereken parametrelerle aynıdır. Bir kez `useCapturetrue` değerine ayarlıyken ve daha sonra tekrar `false` değerine ayarlıyken toplamda iki defa `addEventListener()` öğesini çağırarak tüm olay aşamaları sırasında olayları dinleyebileceğinizi unutmayın. Her iki olay dinleyicisini de kaldırmak için, bir kez `useCapturetrue` değerine ayarlıyken ve daha sonra tekrar `false` değerine ayarlıyken toplam iki defa `removeEventListener()` öğesini çağırmanız gerekir.

Olaylar gönderme

`dispatchEvent()` yöntemi, olay akışına özel bir olay nesnesi göndermek için ileri düzey programcılar tarafından kullanılabilir. Bu yöntem tarafından kabul edilen tek parametre, Event sınıfının veya Event sınıfının bir alt sınıfının örneği olması gereken bir olay nesnesine başvurudur. Gönderildikten sonra, olay nesnesinin `target` özelliği, `dispatchEvent()` öğesinin çağrıldığı nesneye ayarlanır.

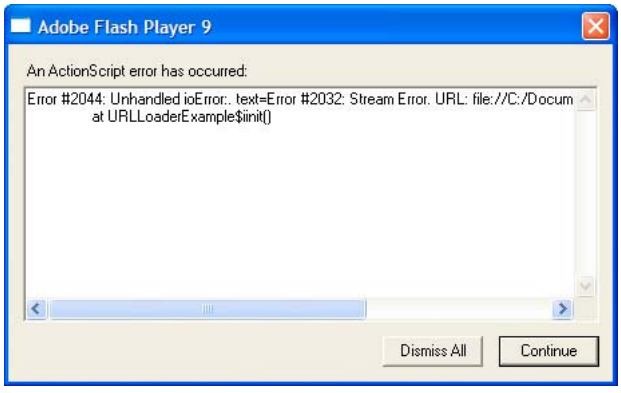
Mevcut olay dinleyicilerini kontrol etme

IEventDispatcher arabiriminin son iki yöntemi, olay dinleyicilerinin varlığı hakkında yararlı bilgiler sağlar. Belirli bir görüntüleme nesnesindeki belirli bir olay türü için bir olay dinleyicisi bulunursa, `hasEventListener()` yöntemi `true` değerini döndürür. Ayrıca belirli bir görüntüleme listesi nesnesi için bir dinleyici bulunursa da `willTrigger()` yöntemi `true` değerini döndürür, ancak `willTrigger()` öğesi yalnızca o görüntüleme nesnesinde değil, olay akışının tüm aşamaları için o görüntüleme listesi nesnesinin tüm üst öğelerinde de dinleyicileri kontrol eder.

Dinleyicisi olmayan Error olayları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da hata işleme için birincil mekanizma, olaylar değil istisnalardır ancak dosya yükleme gibi senkronize olmayan işlemler için istisna işleme çalışmaz. Böyle bir senkronize olmayan işlem sırasında bir hata oluşursa, Flash Player ve AIR uygulaması bir hata olayı nesnesi gönderir. Hata olayı için bir dinleyici oluşturmazsanız, Flash Player ve AIR uygulamasının hata ayıklayıcı sürümü, hata hakkındaki bilgileri içeren bir iletişim kutusu getirir. Örneğin, Flash Player uygulamasının hata ayıklayıcı sürümü, uygulama geçersiz bir URL'den dosya yüklemeye çalıştığında hatayı açıklayan şu iletişim kutusunu oluşturur:



Çoğu hata olayı, ErrorEvent sınıfını esas alır ve bu nedenle, Flash Player veya AIR uygulamasının görüntülediği hata mesajını saklamak için kullanılan `text` adında bir özelliğe sahiptir. StatusEvent ve NetStatusEvent sınıfları iki istisnadır. Bu sınıfların her ikisi de bir `level` özelliğine (`StatusEvent.level` ve `NetStatusEvent.info.level`) sahiptir. `level` özelliğinin değeri "`error`" olduğunda, bu olay türleri hata olayları olarak değerlendirilir.

Bir hata olayı, SWF dosyasının çalışmasını durdurmaz. Yalnızca tarayıcı eklentilerinin ve bağımsız oynatıcıların hata ayıklayıcı sürümünde bir iletişim kutusu olarak, geliştirme oynatıcısındaki çıktı panelinde bir mesaj olarak ve Adobe Flash Builder'ın günlük dosyasında bir giriş olarak bildirilir. Flash Player veya AIR uygulamalarının yayınlama sürümlerinde ise bu bildirilmez.

Olay işleme örneği: Çalar Saat

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Alarm Clock örneği, kullanıcının alarm çalacağı bir saat belirtmesine ve o anda bir mesajın görüntülenmesine olanak sağlayan bir saatten oluşur. Alarm Clock örneği, “Tarih ve saatlerle çalışma” sayfa 1 bölümündeki SimpleClock uygulamasına dayanır. Alarm Clock, arasında şunların yer aldığı, ActionScript 3.0'daki olaylarla çalışmanın bir çok yönünü gösterir:

- Bir olayı dinleme ve yanıtlama
- Dinleyicilere bir olayı bildirme
- Özel bir olay türü oluşturma

Bu örneğin Flash Professional uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Bu örneğin Flex uygulama dosyalarını edinmek için bkz. http://www.adobe.com/go/as3examples_tr. Alarm Clock uygulama dosyaları, Samples/AlarmClock klasöründe bulunabilir. Uygulama şu dosyaları içerir:

File	Açıklama
AlarmClockApp.mxml veya AlarmClockApp.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/clock/AlarmClock.as	SimpleClock sınıfını genişleterek çalar saat işlevi ekleyen bir sınıf.
com/example/programmingas3/clock/AlarmEvent.as	AlarmClock sınıfının `alarm` olayı için olay nesnesi görevi gören özel bir olay sınıfı (flash.events.Event öğesinin alt sınıfı).
com/example/programmingas3/clock/AnalogClockFace.as	Saati esas alarak yuvarlak bir saat yüzü, akrep, yelkovan ve saniye göstergesi çizer. (SimpleClock örneğinde açıklanmıştır.)
com/example/programmingas3/clock/SimpleClock.as	Basit zaman tutma işlevine sahip bir saat arabirimi bileşeni. (SimpleClock örneğinde açıklanmıştır.)

Alarm Clock öğesine genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnekte, zamanı izleme ve saat yüzünü görüntüleme dahil olmak üzere, saatin birincil işlevi, "Tarih ve saat örneği: Basit analog saat" sayfa 6 bölümünde açıklanan SimpleClock uygulama kodunu yeniden kullanır. AlarmClock sınıfı, alarm saatini ayarlama ve alarm "çaldığında" bildirim sağlama gibi çalar saat için gerekli işlevleri ekleyerek bu örnekteki SimpleClock sınıfını genişletir.

Bir şey gerçekleştiğinde bildirim sağlamak, olayların asıl işidir. AlarmClock sınıfı, istenen eylemleri gerçekleştirmek için diğer nesnelerin dinleyebildiği Alarm olayını kullanıma sunar. Ayrıca AlarmClock sınıfı, alarmını ne zaman tetikleyeceğini belirlemek için Timer sınıfının bir örneğini kullanır. AlarmClock sınıfı gibi, Timer sınıfı da belirli bir süre geçtiğinde diğer nesnelere (bu durumda bir AlarmClock örneği) bildirim göndermek için bir olay sağlar. Çoğu ActionScript uygulamasında olduğu gibi, olaylar, Alarm Clock örnek uygulaması işlevinin önemli bir parçasını oluşturur.

Alarmı tetikleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Önceden de belirtildiği gibi, AlarmClock sınıfının gerçekten sağladığı tek işlev, alarmın ayarlanması ve tetiklenmesiyle ilgilidir. Yerleşik Timer sınıfı (flash.utils.Timer), geliştiricinin belirtilen bir süre geçtikten sonra çalıştırılacak kod tanımlaması için bir yol sağlar. AlarmClock sınıfı, alarmın ne zaman verileceğini belirlemek için bir Timer örneği kullanır.

```
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);
}
```

AlarmClock sınıfında tanımlanan Timer örneği, `alarmTimer` olarak adlandırılır. AlarmClock örneği için gerekli kurulum işlemlerini gerçekleştiren `initClock()` yöntemi, `alarmTimer` değişkeniyle iki şey yapar. İlk olarak, Timer örneğine 0 milisaniye beklemesini ve yalnızca bir defa zamanlayıcı olayını tetiklemesini bildiren parametrelerle değişken başlatılır. `alarmTimer` başlatıldıktan sonra kod, değişkenin `timer` olayını dinlemek istediğini belirtmek için değişkenin `addEventListener()` yöntemini çağırır. Timer örneği, belirtilen bir süre geçtikten sonra `timer` olayını göndererek çalışır. AlarmClock sınıfının, kendi alarmını vermesi için `timer` olayının ne zaman gönderildiğini bilmesi gerekir. AlarmClock kodu, `addEventListener()` öğesini çağırarak, `alarmTimer` ile bir dinleyici olarak kendisini kaydeder. Bu iki parametre, AlarmClock öğesinin `timer` olayını (`TimerEvent.TIMER` sabitiyle belirtilen) dinlemek istediğini ve olay gerçekleştiğinde, olaya yanıt olarak AlarmClock sınıfının `onAlarm()` yönteminin çağrılması gerektiğini belirtir.

Gerçekten alarmı ayarlamak için, şu şekilde AlarmClock sınıfının `setAlarm()` yöntemi çağrılır:

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

Bu yöntem, alarm mesajını saklama ve alarmın verileceği gerçek anı temsil eden bir Date nesnesi (`alarmTime`) oluşturma dahil olmak üzere, birçok şeyi gerçekleştirir. Şu an ele alınan konu içinde en önemlisi, yöntemin son birkaç satırında `alarmTimer` değişkeninin zamanlayıcısının ayarlanması ve etkinleştirilmesidir. İlk olarak, `reset()` yöntemi çağrılarak zamanlayıcı durdurulur ve önceden çalışıyor olma ihtimaline karşı sıfırlanır. Daha sonra, alarm verilmeden önce kaç milisaniye geçmesi gerektiğini belirlemek için, `alarmTime` değişkeninin değerinden geçerli zaman (`now` değişkeni tarafından temsil edilen) çıkarılır. Timer sınıfı `timer` olayını mutlak bir zamanda tetiklemez, bu nedenle de bu, `alarmTimer` öğesinin `delay` özelliğine atanan göreceli zaman farkıdır. Son olarak, zamanlayıcıyı gerçekten başlatmak için `start()` yöntemi çağrılır.

Belirtilen süre geçtikten sonra, `alarmTimer` öğesi `timer` olayını gönderir. AlarmClock sınıfı, `onAlarm()` yöntemini o olay için bir dinleyici olarak kaydettiğinden, `timer` olayı gerçekleştiğinde `onAlarm()` çağrılır.

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

Olay dinleyicisi olarak kaydedilen bir yöntemin uygun imza (başka bir deyişle, yöntemin parametreler kümesi ve döndürme türü) ile tanımlanması gerekir. Bir yöntemin, Timer sınıfının `timer` olayına yönelik bir dinleyici olması için, Event sınıfının bir alt sınıfı olan TimerEvent (flash.events.TimerEvent) veri türüne sahip bir parametre tanımlaması gerekir. Timer örneği, olay dinleyicilerini çağırdığında, olay nesnesi olarak bir TimerEvent örneği iletir.

Diğerlerine alarmı bildirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Timer sınıfı gibi, AlarmClock sınıfı da, alarm verildiğinde diğer kodun bildirim almasına olanak veren bir olay sağlar. Bir sınıfın ActionScript'te yerleşik olay işleme çerçevesini kullanması için, o sınıfın flash.events.IEventDispatcher arabirimini uygulaması gerekir. Daha yaygın olarak bu, flash.events.EventDispatcher sınıfı genişletilerek (veya EventDispatcher'ın alt sınıfları genişletilerek) yapılır, böylece IEventDispatcher öğesinin standart bir uygulaması sağlanır. Daha önce açıklandığı gibi, AlarmClock sınıfı SimpleClock sınıfını genişletir, ve bu da EventDispatcher sınıfını (bir miras zinciri üzerinden) genişletir. Tüm bunlar, AlarmClock sınıfının kendi olaylarını sağlamak için yerleşik işlevlere önceden sahip olduğu anlamına gelir.

Diğer kod, AlarmClock öğesinin EventDispatcher öğesinden miras aldığı `addEventListener()` yöntemini çağırarak AlarmClock sınıfının `alarm` olayının kendisine bildirilmesi için kaydolabilir. Bir AlarmClock örneği, `alarm` olayının verildiğini diğer koda bildirmeye hazır olduğunda, yine EventDispatcher öğesinden miras alınan `dispatchEvent()` yöntemini çağırarak bunu yapar.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

Bu kod satırları, AlarmClock sınıfının `onAlarm()` yönteminden (önceden tamamı gösterilen) alınır. AlarmClock örneğinin `dispatchEvent()` yöntemi çağrılır, daha sonra bu yöntem, tüm kayıtlı dinleyicilere AlarmClock örneğinin `alarm` olayının tetiklendiğini bildirir. `dispatchEvent()` öğesine iletilen parametre, dinleyici yöntemlerine iletilecek olay nesnesidir. Bu durumda, bu örnek için özel olarak oluşturulmuş bir Event alt sınıfı olan AlarmEvent sınıfının bir örneğidir.

Özel bir alarm olayı sağlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tüm olay dinleyicileri, tetiklenmekte olan belirli bir olay hakkında bilgilerin yer aldığı bir olay nesnesi parametresi alır. Çoğu durumda olay nesnesi, Event sınıfının bir örneğidir. Ancak bazı durumlarda da, olay dinleyicilerine ek bilgi sağlanması yararlı olacaktır. Bunu gerçekleştirmenin yaygın bir yolu, yeni bir sınıfın, Event sınıfının bir alt sınıfının tanımlanması ve olay nesnesi olarak bu sınıfın bir örneğinin kullanılmasıdır. Bu örnekte, AlarmClock sınıfının `alarm` olayı gönderildiğinde bir AlarmEvent örneği olay nesnesi olarak kullanılır. Burada gösterilen AlarmEvent sınıfı, `alarm` olayı hakkında, özellikle de alarm mesajı hakkında ek bilgiler sağlar:

```
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;
    }
    ...
}
```

Özel bir olay nesnesi sınıfı oluşturmanın en iyi yolu, önceki örnekte gösterildiği gibi, Event sınıfını genişleten bir sınıfın tanımlanmasıdır. AlarmEvent sınıfı, miras alınan işlevleri tamamlamak için, olayla ilişkilendirilmiş alarm mesajının metnini içeren bir `message` özelliğini tanımlar; `message` değeri, AlarmEvent yapıcısında bir parametre olarak iletilir. AlarmEvent sınıfı ayrıca AlarmClock sınıfının `addEventListener()` yöntemi çağrıldığında belirli bir olayı (`alarm`) ifade etmek için kullanılabilen `ALARM` sabitini de tanımlar.

Her Event alt sınıfının, özel işlevler eklemenin yanı sıra ActionScript olay işleme çerçevesinin parçası olarak miras alınan `clone()` yöntemini geçersiz kılması da gerekir. Ayrıca Event alt sınıfları, `toString()` yöntemi çağrıldığında döndürülen değere özel olayın özelliklerini dahil etmek için, miras alınan `toString()` yöntemini isteğe bağlı olarak geçersiz kılabilir.

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

Geçersiz kılınan `clone()` yönteminin, tüm özel özellikleri geçerli örnekle eşleşecek şekilde ayarlanmış olarak özel Event alt sınıfının yeni bir örneğini döndürmesi gerekir. Geçersiz kılınmış `toString()` yönteminde, özel türün adını ve tüm özelliklerinin adlarını ve değerlerini içeren bir dize sağlamak için `formatToString()` yardımcı program yöntemi (Event öğesinden miras alınan) kullanılır.

Bölüm 9: Uygulama etki alanlarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ApplicationDomain sınıfının amacı, ActionScript 3.0 tanımlarının bir tablosunu saklamaktır. Bir SWF dosyasındaki tüm kodlar, bir uygulama etki alanında var olacak şekilde tanımlanır. Aynı güvenlik etki alanında bulunan sınıfları bölümlemek için uygulama etki alanlarını kullanırsınız. Böylece, aynı sınıfın birden çok tanımının var olmasına ve alt öğelerin üst tanımları yeniden kullanmasına olanak sağlanır.

Loader sınıfı API'sini kullanarak ActionScript 3.0'da yazılmış harici bir SWF dosyasını yüklerken uygulama etki alanlarını kullanabilirsiniz. (ActionScript 1.0 veya ActionScript 2.0'da yazılmış bir görüntüyü ya da SWF dosyasını yüklerken uygulama etki alanlarını kullanamayacağınızı unutmayın.) Yüklenen sınıfta bulunan tüm ActionScript 3.0 tanımları, uygulama etki alanında saklanır. SWF dosyasını yüklerken LoaderContext nesnesinin `applicationDomain` parametresini `ApplicationDomain.currentDomain` öğesine ayarlayarak, dosyanın, Loader nesnesinin uygulama etki alanıyla aynı uygulama etki alanında bulunacağını belirtebilirsiniz Yüklenen SWF dosyasını aynı uygulama etki alanına koyarak, doğrudan sınıflarına erişebilirsiniz. İlişkilendirilmiş sınıf adları üzerinden erişebileceğiniz gömülü medya içeren bir SWF dosyası yüklüyorsanız veya yüklü SWF dosyasının yöntemlerine erişmek istiyorsanız, bu faydalı olabilir.

Şu örnekte, welcome() adlı genel bir yöntem tanımlayan ayrı bir Greeter.swf dosyasına erişimi olduğu varsayılmaktadır:

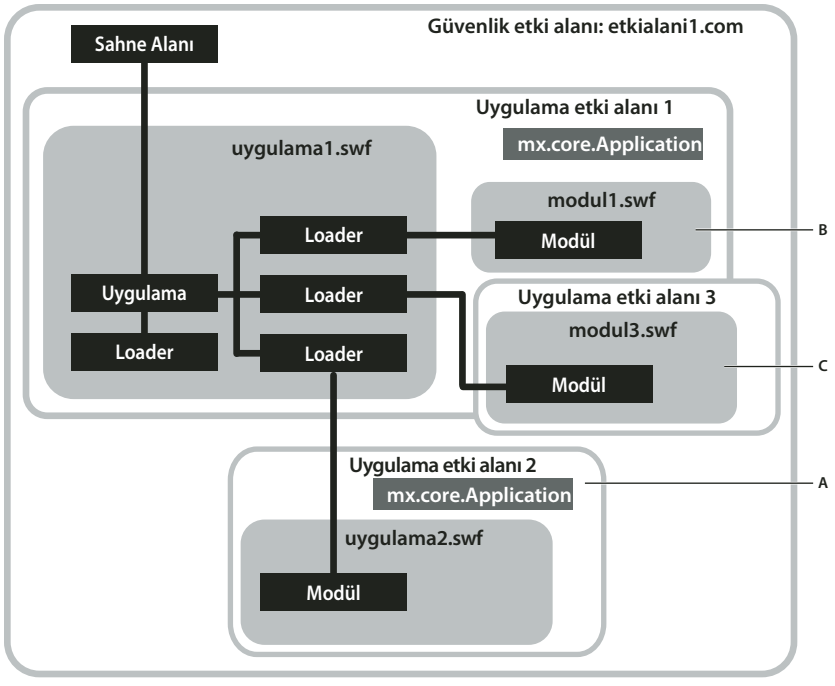
```
package
{
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.*;
    import flash.net.URLRequest;
    import flash.system.ApplicationDomain;
    import flash.system.LoaderContext;

    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
        }
    }
}
```

Ayrıca Flash Platformu için Adobe ActionScript 3.0 Başvurusu ApplicationDomain sınıfı örneğine bakın.

Uygulama etki alanlarıyla çalıştığınızda dikkate alınması gereken diğer şeyler şunlardır:

- Bir SWF dosyasındaki tüm kodlar, bir uygulama etki alanında var olacak şekilde tanımlanır. *Geçerli etki alanı*, ana uygulamanızın çalıştığı yerdir. *Sistem etki alanı*, geçerli etki alanı da dahil olmak üzere tüm uygulama etki alanlarını içerir, başka bir deyişle, tüm Flash Player sınıflarını içerir.
- Sistem etki alanı dışındaki tüm uygulama etki alanları, ilişkilendirilmiş bir üst etki alanına sahiptir. Ana uygulamanızın uygulama etki alanının üst etki alanı, sistem etki alanıdır. Yüklenmiş sınıflar yalnızca üst öğeleri tarafından önceden tanımlanmamışsa tanımlanır. Yüklenmiş bir sınıf tanımını, yeni bir tanımla geçersiz kılamazsınız.

Aşağıdaki diyagramda, domain1.com adında tek bir etki alanındaki çeşitli SWF dosyalarından içerik yükleyen bir uygulama gösterilmektedir. Yüklediğiniz içeriğe bağlı olarak, farklı uygulama etki alanları kullanılabilir. Aşağıdaki metin, uygulamadaki her SWF dosyası için uygun uygulama etki alanını ayarlamak için kullanılan mantığı açıklar.



***A.** Kullanım A: **B.** Kullanım B **C.** Kullanım C*

Ana uygulama dosyası application1.swf'dir. Bu, diğer SWF dosyalarından içerik yükleyen Loader nesnelerini içerir. Bu senaryoda, geçerli etki alanı Uygulama etki alanı 1'dir. Kullanım A, kullanım B ve kullanım C, bir uygulamadaki her SWF dosyası için uygun uygulama etki alanının ayarlanmasına yönelik farklı teknikleri gösterir.

Kullanım A: Sistem etki alanının bir alt öğesini oluşturarak alt SWF dosyasını bölümleyin. Diyagramda, Uygulama etki alanı 2, sistem etki alanının bir alt öğesi olarak oluşturulur. application2.swf dosyası Uygulama etki alanı 2'de yüklenir ve böylece sınıf tanımları, application1.swf dosyasında tanımlanan sınıflardan bölümlenir.

Bu tekniğin kullanıldığı koşullardan biri, eski bir uygulamanın, aynı uygulamanın yeni bir sürümünü çakışma olmadan dinamik olarak yüklemesini sağlamaktır. Aynı sınıf adları kullanılsa da, bunlar farklı uygulama etki alanlarına bölümlendiğinden, çakışma oluşmaz.

Aşağıdaki kod, sistem etki alanının alt öğesi olan bir uygulama etki alanı oluşturur ve bu uygulama etki alanını kullanarak SWF dosyasının yüklemesini başlatır:

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

Kullanım B: Geçerli sınıf tanımlarına yeni sınıf tanımları ekleyin. module1.swf dosyasının uygulama etki alanı, geçerli etki alanına (Uygulama etki alanı 1) ayarlanır. Böylece, uygulamanın geçerli sınıf tanımları kümesine yeni sınıf tanımları eklemeniz sağlanır. Bu, ana uygulamanın çalışma zamanı paylaşılan kütüphanesi için kullanılabilir. Yüklenen SWF, uzak paylaşılan kütüphane (RSL) olarak değerlendirilir. Uygulama başlatılmadan önce önyükleyici tarafından RSL'lerin yüklenmesini sağlamak için bu tekniği kullanın.

Aşağıdaki kod, bir SWF'yi yükler ve bunun uygulama etki alanını geçerli etki alanına ayarlar:

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

Kullanım C: Geçerli etki alanının yeni bir alt etki alanını oluşturarak üst sınıfın sınıf tanımlarını kullanın. module3.swf dosyasının uygulama etki alanı, geçerli etki alanının bir alt öğesidir ve alt öğe, tüm sınıfların üst öğe sürümlerini kullanır. Bu tekniğin kullanım koşullarından biri, ana uygulama türlerini kullanan ve ana uygulamanın alt öğesi olarak yüklenmiş çok ekranlı zengin Internet uygulamasının (RIA) bir modülü olabilir. Tüm sınıfların her zaman geriye doğru uyumlu olacak şekilde güncellenmesini ve yükleyen uygulamanın her zaman yüklenen şeylerden daha yeni olmasını sağlayabilirseniz, alt öğeler üst öğe sürümlerini kullanır. Yeni bir uygulama etki alanınızın olması, alt SWF dosyasına başvuru olmamasını sağlamanız durumunda, çöp toplama için tüm sınıf tanımlarınızı boşaltmanıza olanak sağlar.

Bu teknik, yüklenen modüllerin, yükleyenin tek nesnelerini ve statik sınıf üyelerini paylaşmasına olanak sağlar.

Aşağıdaki kod, geçerli etki alanının yeni bir alt etki alanını oluşturur ve bu uygulama etki alanını kullanarak SWF dosyasının yüklemesini başlatır:

```
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);
```

Bölüm 10: Görüntü programlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

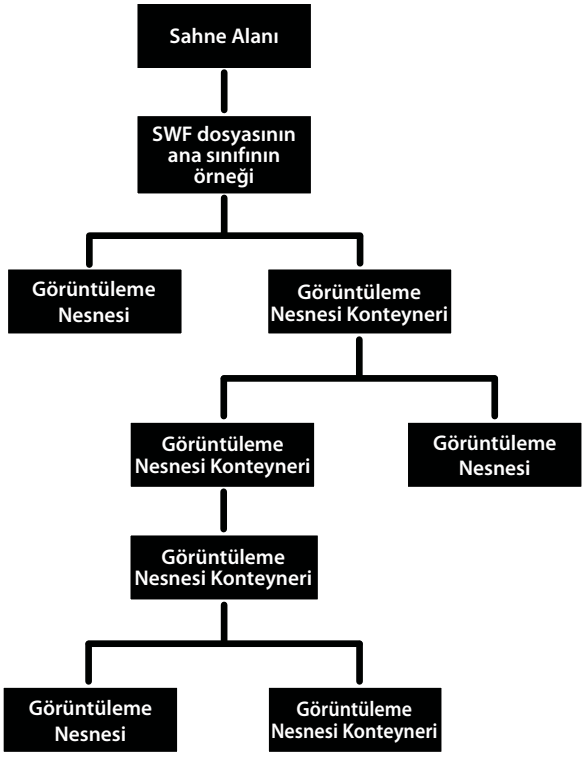
Görsel öğeler Adobe® ActionScript® 3.0 'da görüntü sahnesindeki görüntüleme nesneleriyle çalışarak programlanır. Örneğin, ActionScript görüntü programlama API'sini kullanarak görüntüleme nesnelerini ekleyebilir, taşıyabilir, kaldırabilir ve sıralayabilir, filtreler ve maskeler uygulayabilir, vektör ve bitmap grafikleri çizebilir ve üç boyutlu dönüştürmeler gerçekleştirebilirsiniz. Görüntüleme programlaması için kullanılan ana sınıflar flash.display paketinin parçalarıdır.

Not: *Adobe® AIR™ HTML oluşturma ve görüntüleme için HTMLLoader nesnesini sağlar. HTMLLoader HTML DOM öğesinin görsel öğelerini bir tek görüntüleme nesnesi olarak oluşturur. DOM'un bağımsız öğelerine ActionScript görüntüleme listesi hiyerarşisi üzerinden doğrudan erişemezsiniz. Onun yerine, bu DOM öğelerine HTMLLoader tarafından sağlanan ayrı DOM API'leri kullanarak erişebilirsiniz.*

Görüntü programlama temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0 ile yerleşik olan her uygulama, aşağıda gösterildiği gibi, *görüntüleme listesi* olarak bilinen görüntülenen nesnelerin bir hiyerarşisine sahiptir. Görüntüleme listesi, uygulamadaki tüm görünür öğeleri içerir.



Resimde gösterildiği gibi, görüntüleme öğeleri şu gruplardan bir veya birkaçına ayrılır:

- Sahne Alanı

  Sahne Alanı, görüntüleme nesnelerinin temel kabıdır. Her uygulama, ekrandaki tüm görüntüleme nesnelerini içeren tek bir Stage nesnesine sahiptir. Sahne Alanı, üst düzey bir kap olup görüntüleme listesi hiyerarşisinin en üstünde yer alır:

  Her SWF dosyası, *SWF dosyasının ana sınıfı* olarak bilinen ilişkilendirilmiş bir ActionScript sınıfına sahiptir. SWF dosyası Flash Player veya Adobe AIR'de açıldığında, Flash Player veya AIR bu sınıfa ilişkin yapıcı işlevini çağırır ve oluşturulan örnek (her zaman bir görüntüleme nesnesi türüdür) Stage nesnesinin alt öğesi olarak eklenir. Bir SWF dosyasının ana sınıfı her zaman Sprite sınıfını genişletir (Daha fazla bilgi için, bkz. "Görüntüleme listesi yaklaşımının avantajları" sayfa 158).

  Herhangi bir DisplayObject örneğinin `stage` özelliği üzerinden Sahne Alanı'na erişebilirsiniz. Daha fazla bilgi için, bkz. "Sahne Alanı özelliklerini ayarlama" sayfa 167.

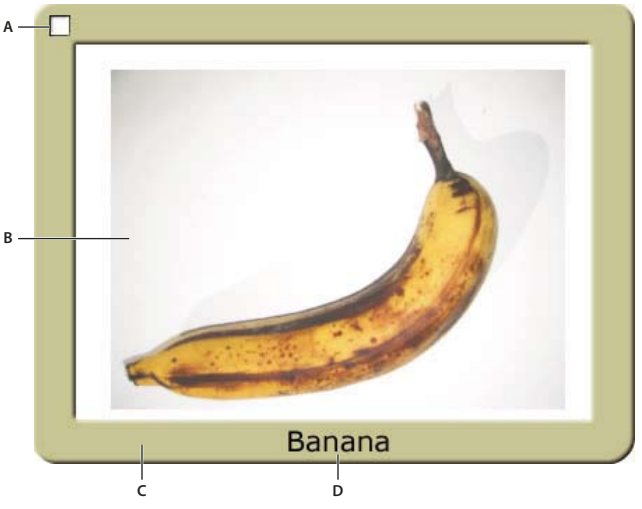
- Görüntüleme nesneleri

  ActionScript 3.0'da, bir uygulamanın ekranında görüntülenen tüm öğeler, *görüntüleme nesneleri* türüdür. flash.display paketi birkaç başka sınıf tarafından genişletilen temel sınıf olan DisplayObject sınıfını içerir. Bu farklı sınıflar, vektör şekilleri, film klipleri ve metin alanları gibi farklı türlerdeki görüntüleme nesnelerini temsil eder. Bu sınıflara bir genel bakış için, bkz. "Görüntüleme listesi yaklaşımının avantajları" sayfa 158.

- Görüntüleme nesnesi kapları

  Görüntüleme nesnesi kapları, kendi görsel temsillerine sahip olmanın yanı sıra, görüntüleme nesnesi niteliğindeki alt nesneleri de içerebilen özel bir görüntüleme nesnesi türüdür.

  DisplayObjectContainer sınıfı, DisplayObject'in alt sınıfıdır. DisplayObjectContainer nesnesi, *alt öğelistesinde* birden çok görüntüleme nesnesi içerebilir. Örneğin, aşağıdaki resimde, çeşitli görüntüleme nesnelerini içeren ve Sprite olarak bilinen bir DisplayObjectContainer nesnesi türü gösterilmektedir:



***A.** SimpleButton nesnesi. Bu görüntüleme nesnesi türü, farklı "basılı olmama", "basılı" ve "üzerinde" durumlarına sahiptir. **B.** Bitmap nesnesi. Bu durumda, Bitmap nesnesi bir Loader nesnesi yoluyla harici bir JPEG'den yüklenmiştir. **C.** Shape nesnesi. "Resim karesi", ActionScript'te çizilen yuvarlak köşeli bir dikdörtgen içerir. Bu Shape nesnesine Gölge filtresi uygulanmıştır. **D.** TextField nesnesi.*

  Görüntüleme nesneleri ele alındığında, DisplayObjectContainer nesneleri ayrıca *görüntüleme nesnesi kabı* veya yalnızca *kap* olarak bilinir. Daha önce de belirtildiği gibi, Sahne Alanı bir görüntüleme nesnesi kabıdır.

Görünebilen tüm görüntüleme nesneleri DisplayObject sınıfından miras alsa da, her birinin türü DisplayObject sınıfının belirli bir alt sınıfındandır. Örneğin, Shape sınıfı veya Video sınıfı için bir yapıcı işlevi vardır ancak DisplayObject sınıfı için herhangi bir yapıcı işlevi yoktur.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi, ActionScript grafiklerini programlarken karşılaşacağınız önemli terimleri içerir:

Alfa Bir renkteki saydamlık miktarını (daha doğrusu, opaklık miktarını) temsil eden renk değeri. Örneğin, %60 alfa kanalı değerine sahip bir renk, tüm gücünün yalnızca %60'ını gösterir ve %40 saydamdır.

Bitmap grafik Bilgisayarda, renkli piksellerin bir ızgarası (satır ve sütunları) olarak tanımlanan bir grafik. Yaygın bitmap grafikleri dijital fotoğrafları ve benzeri görüntüleri içerir.

Karışım modu Örtüşen iki görüntünün içeriklerinin nasıl etkileşim kurması gerektiğine yönelik bir belirtim. Genellikle, diğerinin üzerindeki bir opak görüntü, altındaki görüntünün görünmesini tamamen engeller; ancak farklı karışım modları, en son elde edilen görüntü iki görüntünün birleşimi olacak şekilde görüntü renklerinin farklı şekillerde karışım oluşturmasını sağlar.

Görüntüleme listesi Flash Player ve AIR uygulaması tarafından görünür ekran içeriği olarak oluşturulacak görüntüleme nesnelerinin hiyerarşisi. Sahne Alanı, görüntüleme listesinin köküdür ve Sahne Alanı'na veya alt öğelerinden birine eklenen tüm görüntüleme nesneleri, görüntüleme listesini oluşturur (nesne gerçekten oluşturulmasa da, örneğin, Sahne Alanı sınırlarının dışında olsa da).

Görüntüleme nesnesi Flash Player veya AIR uygulamasında bir tür görsel içeriği temsil eden bir nesne. Görüntüleme listesine yalnızca görüntüleme nesneleri dahil edilebilir ve tüm görüntüleme nesnesi sınıfları, DisplayObject sınıfının alt sınıflarıdır.

Görüntüleme nesnesi kabı Kendi görsel temsillerinin yanı sıra (genellikle) alt görüntüleme nesnelerini de içerebilen özel bir görüntüleme nesnesi türüdür.

SWF dosyasının ana sınıfı Bir SWF dosyasında en dıştaki görüntüleme nesnesinin davranışını tanımlayan sınıf. Bu, kavramsal olarak, SWF dosyasının kendi sınıfıdır. Örneğin Flash geliştirme içinde oluşturulan SWF'de ana sınıf belge sınıfıdır. Diğer tüm zaman çizelgelerini içeren bir "ana zaman çizelgesine" sahiptir; SWF dosyasının ana sınıfı, ana zaman çizelgesinin bir örnek olduğu sınıftır.

Maskeleme Görüntünün belirli bir bölümünü görüntülenmeyecek şekilde gizleme (veya tam tersi şekilde, görüntünün yalnızca belirli bölümlerinin görüntülenmesine izin verme) tekniği. Maske görüntüsünün bazı bölümleri saydam hale gelir, böylece alttaki içerik görünür. Bu terim, ressamların belirli bölgelere boya uygulanmasını önlemek için kullandıkları maskeleme bandıyla ilişki kurularak oluşturulmuştur.

Sahne alanı SWF'deki tüm görsel içeriğin tabanı veya arka planı olan görsel kap.

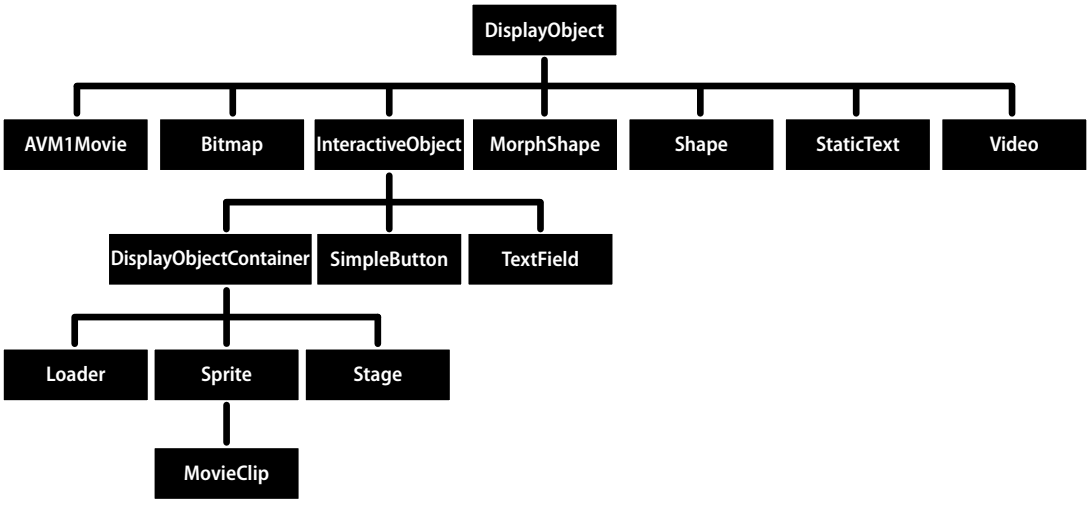
Dönüştürme Nesneyi döndürme, ölçeğini değiştirme, şeklini eğme veya deforme etme ya da rengini değiştirme şeklinde bir grafiğin görsel özellikleri üzerinde yapılan düzenleme.

Vektör grafiği Bilgisayarda, belirli özelliklerle (örn. kalınlık, uzunluk, boyut, açı ve konum) çizilmiş çizgiler ve şekiller olarak tanımlanan bir grafik.

Çekirdek görüntüleme sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da flash.display paketi, Flash Player veya AIR uygulamasında görüntülenebilen görsel nesnelerin sınıflarını içerir. Aşağıdaki resimde, bu çekirdek görüntüleme nesnesi sınıflarının alt sınıf ilişkileri gösterilmektedir.



Resimde, görüntüleme nesnesi sınıflarının sınıf mirası gösterilmektedir. Özellikle StaticText, TextField ve Video olmak üzere bu sınıflardan bazılarının flash.display paketinde bulunmadığı halde DisplayObject sınıfından miras aldığını unutmayın.

DisplayObject sınıfını genişleten tüm sınıflar, yöntem ve özelliklerini miras alır. Daha fazla bilgi için, bkz. "DisplayObject sınıfının özellikleri ve yöntemleri" sayfa 160.

flash.display paketinde bulunan şu sınıfların nesnelerini başlatabilirsiniz:

- Bitmap—Harici dosyalardan yüklenmiş veya ActionScript yoluyla oluşturulmuş bitmap nesnelerini tanımlamak için Bitmap sınıfını kullanırsınız. Loader sınıfı yoluyla, harici dosyalardan bitmap'leri yükleyebilirsiniz. GIF, JPG veya PNG dosyalarını yükleyebilirsiniz. Ayrıca özel veriler içeren bir BitmapData nesnesi oluşturabilir ve sonra bu verileri kullanan bir Bitmap nesnesi oluşturabilirsiniz. İster yüklenmiş isterse ActionScript'te oluşturulmuş olsun, bitmap'leri değiştirmek için BitmapData sınıfının yöntemlerini kullanabilirsiniz. Daha fazla bilgi için, bkz. "Görüntüleme nesnelerini yükleme" sayfa 202 ve "Bitmap'lerle çalışma" sayfa 245.
- Loader—Harici varlıkları (SWF dosyası veya grafik) yüklemek için Loader sınıfını kullanırsınız. Daha fazla bilgi için, bkz. "Görüntüleme içeriğini dinamik olarak yükleme" sayfa 201.
- Shape—Dikdörtgen, çizgi, daire, vb. gibi vektör grafikleri oluşturmak için Shape sınıfını kullanırsınız. Daha fazla bilgi için, bkz. "Çizim API'sini kullanma" sayfa 225.
- SimpleButton—SimpleButton nesnesi, Flash geliştirme aracında oluşturulmuş bir düğme sembolünün ActionScript temsilidir. SimpleButton örneği, dört düğme durumuna sahiptir: basılı olmama, basılı, üzerinde ve vuruş testi (fare ve klavye olaylarına yanıt veren alan).
- Sprite—Sprite nesnesi, kendi grafiklerini içerebildiği gibi alt görüntüleme nesnelerini de içerebilir. (Sprite sınıfı, DisplayObjectContainer sınıfını genişletir.) Daha fazla bilgi için bkz. "Görüntüleme nesnesi kaplarıyla çalışma" sayfa 161 ve "Çizim API'sini kullanma" sayfa 225.

- MovieClip—MovieClip nesnesi, Flash geliştirme aracında oluşturulmuş bir film klibi sembolünün ActionScript biçimidir. MovieClip pratikte Sprite nesnesine benzer, tek farkı bir zaman çizelgesine de sahip olmasıdır. Daha fazla bilgi için, bkz. "Film klipleriyle çalışma" sayfa 324.

flash.display paketinde bulunmayan şu sınıflar, DisplayObject sınıfının alt sınıflarıdır:

- flash.text paketine dahil edilen TextField sınıfı, metin görüntüsü ve girdisi için bir görüntüleme nesnesidir. Daha fazla bilgi için, bkz. "Metinle Çalışmanın temelleri" sayfa 373.
- Flash.text.engine paketine dahil edilen TextLine sınıfı, Flash Metin Motoru ve Metin Mizanpajı Çerçevesi tarafından oluşturulan metin satırlarını görüntülemek için kullanılan ekran nesnesidir. Daha fazla bilgi için bkz. "Flash Metin Motoru'nu kullanma" sayfa 400 ve "Metin Mizanpajı Çerçevesini Kullanma" sayfa 430.
- flash.media paketine dahil edilen Video sınıfı, video dosyalarının görüntülenmesi için kullanılan görüntüleme nesnesidir. Daha fazla bilgi için, bkz. "Videoyla çalışma" sayfa 478.

flash.display paketindeki şu sınıflar, DisplayObject sınıfını genişletir ancak bunların örneklerini oluşturamazsınız. Daha çok bunlar diğer görüntüleme nesnelerinin üst sınıfları görevini görerek ortak işlevselliği tek bir sınıfta birleştirir.

- AVM1Movie—AVM1Movie sınıfı, ActionScript 1.0 ve 2.0'da geliştirilen yüklenmiş SWF dosyalarını temsil etmek için kullanılır.
- DisplayObjectContainer—Loader, Stage, Sprite ve MovieClip sınıflarının her biri, DisplayObjectContainer sınıfını genişletir. Daha fazla bilgi için bkz. "Görüntüleme nesnesi kaplarıyla çalışma" sayfa 161.
- InteractiveObject—InteractiveObject, fare ve klavye ile etkileşim kurmak için kullanılan tüm nesnelerin temel sınıfıdır. SimpleButton, TextField, Loader, Sprite, Stage ve MovieClip nesnelerinin tümü, InteractiveObject sınıfının alt sınıflarıdır. Fare ve klavye etkileşimi oluşturma hakkında daha fazla bilgi için bkz. "Kullanıcı etkileşiminin temelleri" sayfa 562.
- MorphShape—Flash geliştirme aracında bir şekil arası oluşturduğunuzda bu nesneler oluşturulur. ActionScript kullanarak bunları başlatamazsınız ancak görüntüleme listesinden bunlara erişilebilir.
- Stage—Stage sınıfı DisplayObjectContainer sınıfını genişletir. Bir uygulama için yalnızca bir Stage örneği vardır ve bu, görüntüleme listesi hiyerarşisinin en üstünde bulunur. Sahne Alanı'na erişmek için, herhangi bir DisplayObject örneğinin `stage` özelliğini kullanın. Daha fazla bilgi için, bkz. "Sahne Alanı özelliklerini ayarlama" sayfa 167.

Ayrıca flash.text paketindeki StaticText sınıfı, DisplayObject sınıfını genişletir, ancak kodda bunun bir örneğini oluşturamazsınız. Statik metin alanları yalnızca Flash uygulamasında oluşturulur.

Aşağıdaki sınıflar görüntüleme nesnesi veya görüntüleme nesnesi kabı değildir ve görüntüleme listesinde gösterilmezler, ancak sahne alanı üzerinde grafikleri görüntülerler. Bu sınıflar görüntü kapısı adı verilen bir dikdörtgen içine çizilir ve sahne alanına göre konumlandırılır.

- StageVideo—StageVideo sınıfı mümkün olduğunda donanım ivmesini kullanarak video içeriğini görüntüler. Bu sınıf, Flash Player 10.2 sürümünden itibaren kullanılabilir. Daha fazla bilgi için bkz. "Donanım hızlandırmalı gösterim için StageVideo sınıfını kullanma" sayfa 517.
- StageWebView—StageWebView sınıfı HTML içeriğini görüntüler. Bu sınıf AIR 2.5'ten itibaren kullanılabilir. Daha fazla bilgi için bkz. "StageWebView nesneleri" sayfa 1036.

Aşağıdaki fl.display sınıfları, flash.display.Loader ve LoaderInfo sınıflarıyla paralel işlevsellik sağlar. Flash Professional ortamında (CS5.5 veya üstü) geliştirme yapıyorsanız flash.display eşdeğerleri yerine bu sınıfları kullanın. Flash Professional ortamında bu sınıflar RSL önyüklemesine sahip TLF'yi içeren sorunların çözülmesine yardımcı olur. Daha fazla bilgi için bkz. "ProLoader ve ProLoaderInfo sınıflarını kullanma" sayfa 205.

- fl.display.ProLoader—flash.display.Loader sınıfına benzerdir
- fl.display.ProLoaderInfo—flash.display.LoaderInfo sınıfına benzerdir

Görüntüleme listesi yaklaşımının avantajları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da, görüntüleme nesnelerinin farklı türleri için ayrı sınıflar vardır. ActionScript 1.0 ve 2.0'da, aynı nesne türlerinin çoğu MovieClip adındaki tek bir sınıfa dahil edilmiştir.

Sınıfların ve görüntüleme listelerinin hiyerarşik yapısının bu şekilde bireyselleştirilmesi, şu avantajları sağlar:

- Daha verimli oluşturma ve düşük bellek tüketimi
- Gelişmiş derinlik yönetimi
- Görüntüleme listesinin tam çapraz geçişi
- Liste dışı görüntüleme nesneleri
- Görüntüleme nesnelerinin daha kolay şekilde alt sınıflara ayrılması

Daha verimli oluşturma ve daha küçük dosya boyutları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 1.0 ve 2.0'da, yalnızca bir MovieClip nesnesinde şekil çizebilirsiniz. ActionScript 3.0'da, şekil çizebileceğiniz daha basit görüntü nesnesi sınıfları vardır. Bu ActionScript 3.0 görüntüleme nesnesi sınıfları, MovieClip nesnesinin içerdiği tüm yöntem ve özellik kümesini içermediğinden, bellek ve işlemci kaynaklarını daha az tüketir.

Örneğin, her MovieClip nesnesi, film klibinin zaman çizelgesine yönelik özellikleri içerirken Shape nesnesi bu özellikleri içermez. Zaman çizelgesinin yönetilmesine yönelik özellikler çok fazla bellek ve işlemci kaynağı tüketebilir. ActionScript 3.0'da, Shape nesnesinin kullanılması daha iyi performans elde edilmesini sağlar. Shape nesnesi, daha karmaşık olan MovieClip nesnesinden daha az yüke sahiptir. Flash Player ve AIR uygulamalarının kullanılmayan MovieClip özelliklerini yönetmesi gerekmez, bu da hızı artırıp nesnenin kullandığı bellek alanını azaltır.

Gelişmiş derinlik yönetimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 1.0 ve 2.0'da, derinlik doğrusal bir derinlik yönetimi şeması ve `getNextHighestDepth()` gibi yöntemlerle yönetiliyordu.

ActionScript 3.0, görüntüleme nesnelerinin derinliğinin yönetilmesi için daha kullanışlı yöntem ve özelliklere sahip olan DisplayObjectContainer sınıfını içerir.

ActionScript 3.0'da, bir görüntüleme nesnesini, DisplayObjectContainer örneğinin alt öğe listesinde yeni bir konuma taşıdığınızda, görüntüleme nesnesi kabındaki diğer alt öğeler otomatik olarak yeniden konumlandırılır ve bunlara görüntüleme nesnesi kabında uygun alt dizin konumları atanır.

Ayrıca, ActionScript 3.0'da herhangi bir görüntüleme nesnesi kabının alt nesnelerinin tümünün keşfedilmesi de her zaman mümkündür. Her DisplayObjectContainer örneği, görüntüleme nesnesi kabındaki alt öğelerin sayısını listeleyen bir `numChildren` özelliğine sahiptir. Ve görüntüleme nesnesi kabının alt öğe listesi her zaman dizinlenmiş bir liste olduğundan, 0 dizin konumundan son dizin konumuna (`numChildren - 1`) kadar listedeki her nesneyi inceleyebilirsiniz. ActionScript 1.0 ve 2.0'da, MovieClip nesnesinin yöntemleri ve özellikleriyle bu mümkün değildi.

ActionScript 3.0'da, görüntüleme listesinde sırayla geçiş yapabilirsiniz; görüntüleme nesnesi kabının alt öğe listesinin dizin sayılarında herhangi bir boşluk yoktur. Görüntüleme listesinde geçiş yapılması ve nesnelerin derinliğinin yönetilmesi, ActionScript 1.0 ve 2.0'a göre daha kolaydır. ActionScript 1.0 ve 2.0'da, bir film klibi derinlik sırasında kesintili boşlukların bulunduğu nesneleri içerebiliyordu, bu da nesne listesinde geçiş yapılmasını güçleştirebiliyordu. ActionScript 3.0'da, görüntüleme nesnesi kabının her alt öğe listesi dahili olarak bir dizi şeklinde önbelleğe alınır ve bu da çok hızlı arama (dizine göre) yapılmasını sağlar. Görüntüleme nesnesi kabının tüm alt öğelerinin döngüye alınması da çok hızlıdır.

ActionScript 3.0'da, DisplayObjectContainer sınıfının `getChildByName()` yöntemini kullanarak görüntüleme nesnesi kabındaki alt öğelere erişebilirsiniz.

Görüntüleme listesinin tam çapraz geçişi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 1.0 ve 2.0'da, Flash geliştirme aracında çizilmiş vektör şekilleri gibi bazı nesnelere erişemezdiniz. ActionScript 3.0'da, ActionScript kullanılarak oluşturulmuş olan nesneler ve Flash geliştirme aracında oluşturulmuş tüm görüntüleme nesneleri dahil olmak üzere, görüntüleme nesnesindeki tüm nesnelere erişebilirsiniz. Ayrıntılar için bkz. “Görüntüleme listesinde geçiş yapma” sayfa 165.

Liste dışı görüntüleme nesneleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da, görünebilir görüntüleme listesinde olmayan görüntüleme nesneleri oluşturabilirsiniz. Bunlar *liste dışı* görüntüleme nesneleri olarak bilinir. Yalnızca önceden görüntüleme listesine eklenmiş bir DisplayObjectContainer örneğinin `addChild()` veya `addChildAt()` yöntemini çağırdığınızda görüntüleme nesnesi görünebilir görüntüleme listesine eklenir.

Birden çok görüntüleme nesnesinin bulunduğu birden çok görüntüleme nesnesi kabını içeren görüntüleme nesneleri gibi karmaşık görüntüleme nesnelerini birleştirmek için, liste dışı görüntüleme nesnelerini kullanabilirsiniz. Görüntüleme nesnelerini liste dışında tutarak, bu görüntüleme nesnelerini oluşturmaya yönelik işleme zamanını kullanmadan karmaşık nesneleri birleştirebilirsiniz. Daha sonra gerektiğinde liste dışı görüntüleme nesnesini görüntüleme nesnesine ekleyebilirsiniz. Ayrıca, bir görüntüleme nesnesi kabının alt öğesini görüntüleme listesinin içine veya dışına ya da görüntüleme listesindeki istediğiniz herhangi bir konuma taşıyabilirsiniz.

Görüntüleme nesnelerinin daha kolay şekilde alt sınıflara ayrılması

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 1.0 ve 2.0'da, temel şekiller oluşturmak veya bitmap'leri görüntülemek için sık sık bir SWF dosyasına yeni MovieClip nesneleri eklemeniz gerekir. ActionScript 3.0'da, DisplayObject sınıfı, Shape ve Bitmap de dahil olmak üzere birçok yerleşik alt sınıfı içerir. ActionScript 3.0'daki sınıflar belirli nesne türleri için daha özelleştirilmiş olduğundan, yerleşik sınıfların temel alt sınıflarının oluşturulması daha kolaydır.

Örneğin, ActionScript 2.0'da bir daire çizmek için, özel sınıfın bir nesnesi başlatıldığında MovieClip sınıfını genişleten bir CustomCircle sınıfı oluşturabilirdiniz. Ancak bu sınıf, MovieClip sınıfındaki, söz konusu sınıf için geçerli olmayan birçok özellik ve yöntemleri de (örn. `totalFrames`) içerirdi. ActionScript 3.0'da ise, Shape nesnesini genişleten ve bu nedenle MovieClip sınıfında bulunan ilgisiz özellik ve yöntemleri içeremeyen bir CustomCircle sınıfını oluşturabilirsiniz. Aşağıdaki kod, bir CustomCircle sınıfı örneğini gösterir:

```
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);
    }
}
```

Görüntüleme nesneleriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sahne Alanı, görüntüleme nesneleri, görüntüleme nesnesi kapları ve görüntüleme listesi ile ilgili temel kavramları anladığınıza göre, bu bölümde, ActionScript 3.0'da görüntüleme nesneleriyle çalışma hakkında daha ayrıntılı bilgiler sağlanmıştır.

DisplayObject sınıfının özellikleri ve yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tüm görüntüleme nesneleri, DisplayObject sınıfının alt sınıflarıdır ve bu nedenle DisplayObject sınıfının özelliklerini ve yöntemlerini miras alır. Miras alınan özellikler, tüm görüntüleme nesneleri için geçerli olan temel özelliklerdir. Örneğin, her görüntüleme nesnesi, görüntüleme nesnesi kabında nesnenin konumunu belirten bir `x` özelliğine ve bir `y` özelliğine sahiptir.

DisplayObject sınıf yapıcısını kullanarak bir DisplayObject örneği oluşturamazsınız. `new` operatörüyle bir nesneyi başlatmak için, Sprite gibi başka türde bir nesne (DisplayObject sınıfının alt sınıf olan bir nesne) oluşturmanız gerekir. Ayrıca, özel bir görüntüleme nesnesi sınıfı oluşturmak isterseniz, görüntüleme nesnesi alt sınıflarından birinin, kullanılabilir bir yapıcı işlevine sahip bir alt sınıfını (örn. Shape sınıfı veya Sprite sınıfı) oluşturmanız gerekir. Ayrıntılı bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki DisplayObject sınıfı tanımına bakın.

Görüntüleme listesine görüntüleme nesneleri ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesi örneğini, görüntüleme listesindeki bir görüntüleme nesnesi kabına eklemezseniz, bir görüntüleme nesnesini başlattığınızda söz konusu nesne ekranda (Sahne Alanı'nda) görüntülenmez. Örneğin, aşağıdaki kodda, kodun son satırını çıkarırsanız, `myText` TextField nesnesi görünebilir durumda olmaz. Kodun son satırında, `this` anahtar sözcüğünün önceden görüntüleme listesine eklenmiş bir görüntüleme nesnesi kabını ifade etmesi gerekir.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

Sahne Alanı'na herhangi bir görsel öğe eklediğinizde, söz konusu öğe, Stage nesnesinin bir *alt öğesi* olur. Bir uygulamada yüklenen birinci SWF dosyası (örneğin, bir HTML sayfasına gömdüğünüz dosya) otomatik olarak Sahne Alanı'nın alt öğesi olarak eklenir. Bu, Sprite sınıfını genişleten herhangi türde bir nesne olabilir.

ActionScript *kullanmadan* (örneğin, bir MXML Flex dosyasına MXML etiketi ekleyerek veya Flash Professional'da Sahne Alanı'na bir öğe yerleştirerek) oluşturduğunuz tüm görüntüleme nesneleri, görüntüleme listesine eklenir. Bu görüntüleme nesnelerini ActionScript yoluyla eklemeseniz de, bu nesnelere ActionScript üzerinden erişebilirsiniz. Örneğin, aşağıdaki kod, geliştirme aracında (ActionScript üzerinden değil) eklenmiş `button1` adında bir nesnenin genişliğini ayarlar:

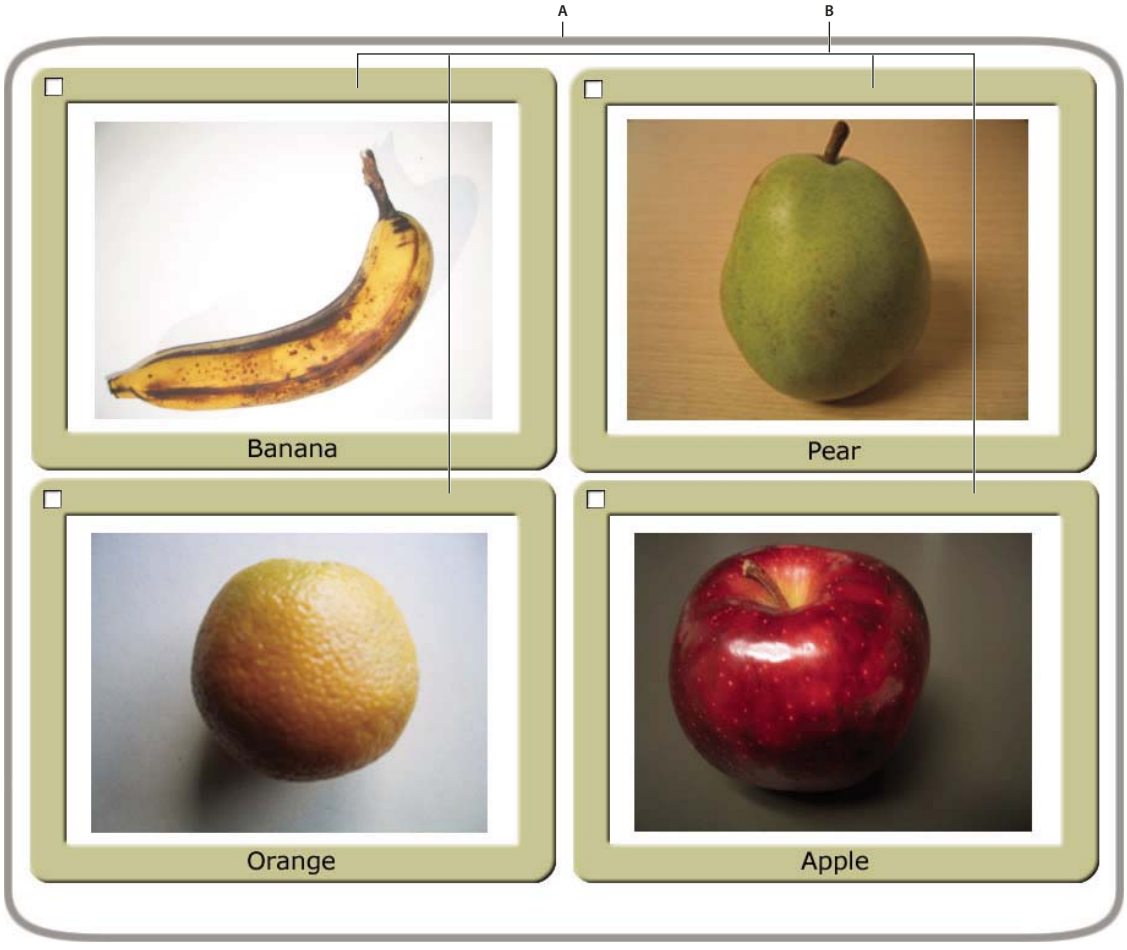
```
button1.width = 200;
```

Görüntüleme nesnesi kaplarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir DisplayObjectContainer nesnesi görüntüleme listesinden silinirse veya bir şekilde taşınır ya da dönüştürülürse, DisplayObjectContainer öğesindeki görüntüleme nesnelerinin her biri de silinir, taşınır veya dönüştürülür.

Görüntüleme nesnesi kabı da bir tür görüntüleme nesnesidir—başka bir görüntüleme nesnesi kabına eklenebilir. Örneğin, aşağıdaki görüntü, bir anahat şeklinin ve başka dört tane görüntüleme nesnesi kabının (PictureFrame türünde) yer aldığı bir görüntüleme nesnesi kabını (`pictureScreen`) göstermektedir:



A. *pictureScreen görüntüleme nesnesi kabının kenarlığını tanımlayan bir şekil* ***B.*** *pictureScreen nesnesinin alt öğeleri olan dört görüntüleme nesnesi kabı*

Bir görüntüleme nesnesinin görüntüleme listesinde görüntülenmesi için, söz konusu görüntüleme nesnesini, görüntüleme listesindeki görüntüleme nesnesi kabına eklemeniz gerekir. Kap nesnesinin `addChild()` yöntemini veya `addChildAt()` yöntemini kullanarak bunu yaparsınız. Örneğin, aşağıdaki kodun son satırı olmadan `myTextField` nesnesi görüntülenmez:

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

Bu kod örneğinde, `this.root` öğesi, kodu içeren MovieClip görüntüleme nesnesi kabını işaret eder. Gerçek kodunuzda farklı bir kap belirtebilirsiniz.

Alt öğeyi, görüntüleme nesnesi kabının alt öğe listesinde belirli bir konuma eklemek için `addChildAt()` yöntemini kullanın. Alt öğe listesindeki sıfır tabanlı dizin konumları, görüntüleme nesnelerinin katmanlamasına (önden arkaya sıralama) göredir. Örneğin, şu üç görüntüleme nesnesini göz önünde bulundurun. Her nesne, Ball adında özel bir sınıftan oluşturulmuştur.

Bu görüntüleme nesnelerinin katmanlaması, `addChildAt()` yöntemi kullanılarak ayarlanabilir. Örneğin, şu kodu göz önünde bulundurun:

```
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);
```

Bu kod çalıştırıldıktan sonra, görüntüleme nesneleri `container` DisplayObjectContainer nesnesine şu şekilde konumlandırılır. Nesnelerin katmanlamasına dikkat edin.

Bir nesneyi görüntüleme listesinin en üstüne yeniden konumlandırmak için söz konusu nesneyi listeye yeniden eklemeniz yeterli olacaktır. Örneğin, bir önceki koddan sonra, `ball_A` öğesini yığının en üst kısmına taşımak için bu kod satırını kullanın:

```
container.addChild(ball_A);
```

Bu kod, `ball_A` öğesini `container` öğesinin görüntüleme listesindeki konumundan etkili şekilde kaldırır ve listenin en üst kısmına yeniden ekler, böylece yığının en üst kısmına taşınmış olur.

Görüntüleme nesnelerinin katman sırasını doğrulamak için `getChildAt()` yöntemini kullanabilirsiniz. `getChildAt()` yöntemi, kendisine ilettiğiniz dizin sayısını esas alarak bir kabın alt nesnelerini döndürür. Örneğin, aşağıdaki kod, `container` DisplayObjectContainer nesnesinin alt öğe listesindeki farklı konumlarda bulunan görüntüleme nesnelerinin adlarını gösterir:

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

Üst kabın alt öğe listesinden bir görüntüleme nesnesini kaldırırsanız, listedeki yüksek konumda bulunan öğelerin her biri alt dizinde bir konum aşağı taşınır. Örneğin, önceki kodla devam ederek, aşağıdaki kod, alt öğe listesinde daha düşük düzeyde bulunan bir görüntüleme nesnesi kaldırıldığında `container` DisplayObjectContainer öğesinde konum 2'de bulunan görüntüleme nesnesinin nasıl konum 1'e hareket ettiğini gösterir:

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

`removeChild()` ve `removeChildAt()` yöntemleri, görüntüleme nesnesi örneğinin tamamını silmez. Yalnızca kabın alt öğe listesinden kaldırır. Halen başka bir değişken tarafından örneğe başvurulabilir. (Bir nesneyi tamamen kaldırmak için `delete` operatörünü kullanın.)

Görüntüleme nesnesi yalnızca bir üst kaba sahip olduğundan, görüntüleme nesnesi örneğini yalnızca bir görüntüleme nesnesi kabına ekleyebilirsiniz. Örneğin, aşağıdaki kod, `tf1` görüntüleme nesnesinin yalnızca bir kapta (bu durumda, DisplayObjectContainer sınıfını genişleten bir Sprite öğesi) bulunabildiğini gösterir:

```
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
```

Bir görüntüleme nesnesi kabında bulunan görüntüleme nesnesini başka bir görüntüleme nesnesi kabına eklerseniz, söz konusu görüntüleme nesnesi birinci görüntüleme nesnesi kabının alt öğe listesinden kaldırılır.

DisplayObjectContainer sınıfı, yukarıda açıklanan yöntemlere ek olarak, aralarında aşağıdakilerin de yer aldığı alt görüntüleme nesneleriyle çalışılmasına yönelik birçok yöntemi tanımlar:

- `contains()`: Görüntüleme nesnesinin bir DisplayObjectContainer öğesinin alt öğesi olup olmadığını belirler.
- `getChildByName()`: Ada göre bir görüntüleme nesnesini alır.
- `getChildIndex()`: Görüntüleme nesnesinin dizin konumunu döndürür.
- `setChildIndex()`: Alt görüntüleme nesnesinin konumunu değiştirir.
- `removeChildren()`: Birden çok alt görüntü nesnesi kaldırır.
- `swapChildren()`: İki görüntüleme nesnesinin önden arkaya sırasını değiştirir.

- `swapChildrenAt()`: Dizin değerleriyle belirtildiği şekilde, iki görüntüleme nesnesinin önden arkaya sırasını değiştirir.

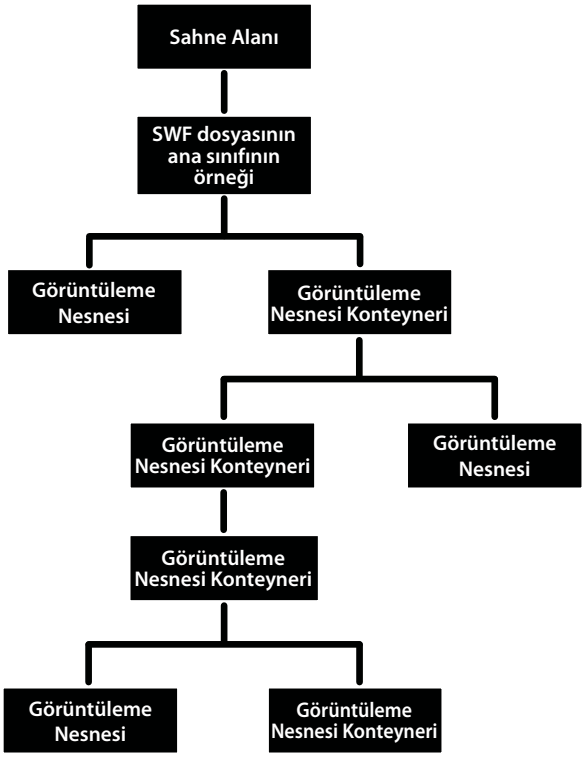
Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu içindeki ilgili girişlere bakın.

Görüntüleme listesi dışında olan (Sahne Alanı'nın alt öğesi olan bir görüntüleme nesnesi kabında bulunmayan) bir görüntüleme nesnesinin *liste dışı* görüntüleme nesnesi olarak bilindiğini unutmayın.

Görüntüleme listesinde geçiş yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Gördüğünüz gibi, görüntüleme listesi bir ağaç yapısındadır. Ağacın en üst kısmında Sahne Alanı bulunur ve birden çok görüntüleme nesnesi içerebilir. Kendileri de görüntüleme nesnesi kabı olan bu görüntüleme nesneleri, başka görüntüleme nesnelerini veya görüntüleme nesnesi kaplarını içerebilir.



DisplayObjectContainer sınıfı, görüntüleme nesnesi kaplarının alt öğe listeleri yoluyla görüntüleme listesinde geçiş yapılmasına yönelik özellik ve yöntemleri içerir. Örneğin, `container` nesnesine (bu Sprite olur ve Sprite sınıfı DisplayObjectContainer sınıfını genişletir) `title` ve `pict` olmak üzere iki görüntüleme nesnesi ekleyen şu kodu göz önünde bulundurun:

```
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);
```

`getChildAt()` yöntemi, belirli bir dizin konumunda görüntüleme listesinin alt öğesini döndürür:

```
trace(container.getChildAt(0) is TextField); // true
```

Ayrıca ada göre de alt nesnelere erişebilirsiniz. Her görüntüleme nesnesi bir ad özelliğine sahiptir ve ad özelliği atamazsanız, Flash Player veya AIR uygulaması `"instance1"` gibi varsayılan bir değer atar. Örneğin, aşağıdaki kod, `"banana loader"` adına sahip bir alt görüntüleme nesnesine erişmek için `getChildByName()` yönteminin nasıl kullanıldığını gösterir:

```
trace(container.getChildByName("banana loader") is Loader); // true
```

`getChildByName()` yönteminin kullanılması, `getChildAt()` yöntemine göre daha düşük performans elde edilmesine neden olabilir.

Görüntüleme nesnesi kabı, görüntüleme listesinde alt nesneler olarak başka görüntüleme nesnesi kaplarını içerebildiğinden, ağaç olarak uygulamanın tam görüntüleme listesinde geçiş yapabilirsiniz. Örneğin, daha önce gösterilen kod alıntısında, `pict` Loader nesnesi için yükleme işlemi tamamlandıktan sonra, `pict` nesnesine yüklenmiş tek bir alt görüntüleme nesnesi olur ve bu da bitmap'tir. Bu bitmap görüntüleme nesnesine erişmek için, `pict.getChildAt(0)` yazabilirsiniz. Ayrıca şunu da yazabilirsiniz: `container.getChildAt(0).getChildAt(0)` (`container.getChildAt(0) == pict` olduğundan).

Aşağıdaki işlev, görüntüleme nesnesi kabından görüntüleme listesinin girintili bir `trace()` çıktısını sağlar:

```
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

Flex kullanıyorsanız, Flex öğesinin birçok bileşen görüntüleme nesnesi sınıfını tanımladığını ve bu sınıfların DisplayObjectContainer sınıfının görüntüleme listesi erişim yöntemlerini geçersiz kıldığını bilmeniz gerekir. Örneğin, mx.core paketinin Container sınıfı, `addChild()` yöntemini ve DisplayObjectContainer sınıfının diğer yöntemlerini (Container sınıfının genişlettiği) geçersiz kılar. `addChild()` yöntemi olması durumunda, tüm görüntüleme nesnesi türlerini Flex'te bir Container örneğine ekleyemeyeceğiniz şekilde sınıf yöntemi geçersiz kılar. Bu durumda, geçersiz kılınan yöntem, eklediğiniz alt nesnenin bir mx.core.UIComponent nesnesi türü olmasını gerektirir.

Sahne Alanı özelliklerini ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Stage sınıfı, DisplayObject sınıfının çoğu özellik ve yöntemini geçersiz kılar. Bu geçersiz kılınmış özellik veya yöntemlerden birini çağırırsanız, Flash Player ve AIR bir istisna atar. Örneğin, uygulama için ana kap olarak konumu sabitlenmiş olduğundan, Sahne Alanı `x` veya `y` özelliğine sahip değildir. `x` ve `y` özellikleri, bir görüntüleme nesnesinin, kabına göre konumunu ifade eder ve Sahne Alanı başka bir görüntüleme nesnesi kabında bulunmadığından bu özellikler geçerli değildir.

Not: *Stage sınıfının bazı özellik ve yöntemleri, yalnızca yüklenen birinci SWF dosyası olarak aynı güvenlik sanal alanında bulunan görüntüleme nesneleri için kullanılabilir durumdadır. Ayrıntılar için, bkz. "Sahne Alanı güvenliği" sayfa 1075.*

Oynatma kare hızını denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir uygulamaya yüklenen tüm SWF dosyalarının kare hızını ayarlamak için, Stage sınıfının `framerate` özelliği kullanılır. Daha fazla bilgi için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Sahne Alanı ölçeklemeyi denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Ekranda Flash Player'ı veya AIR'yi temsil eden bölüm yeniden boyutlandırıldığında, çalışma zamanı bunu düzeltmek için Sahne Alanı'nı otomatik olarak ayarlar. Stage sınıfının `scaleMode` özelliği, Sahne Alanı içeriklerinin nasıl ayarlanacağını belirler. Bu özellik, flash.display.StageScaleMode sınıfında sabitler olarak tanımlanan dört farklı değere ayarlanabilir:

- `StageScaleMode.EXACT_FIT` öğesi SWF'yi orijinal içeriğin en boy oranını dikkate almaksızın yeni sahne alanı boyutlarını kaplayacak şekilde ölçeklendirir. Ölçek faktörleri en ve boy için aynı olmayabilir, bu yüzden sahne alanının en boy oranı değiştirilirse içerik sıkıştırılmış veya uzatılmış görünebilir.
- `StageScaleMode.SHOW_ALL` öğesi SWF'yi içeriğin en boy oranını değiştirmeden yeni sahne alanı boyutları içine sığdırır. Bu ölçek modu tüm içeriği görüntüler ancak kenarlarda, standart bir televizyonda geniş ekran film izlerken görülen siyah çubuklara benzer şekilde bir "siyah bant" oluşmasına neden olabilir.
- `StageScaleMode.NO_BORDER` öğesi SWF'yi içeriğin en boy oranını değiştirmeden tamamen yeni sahne alanı boyutları içine sığdırır. Bu ölçek modu sahne görüntüleme alanının tamamını kullanır ancak kırpılmaya neden olabilir.
- `StageScaleMode.NO_SCALE` SWF'yi ölçeklendirmez. Yeni sahne alanı boyutları daha küçükse içerik kırpılır; daha genişse boş alan eklenir.

  Yalnızca `StageScaleMode.NO_SCALE` ölçekleme modunda, yeniden boyutlandırılan sahne alanının gerçek piksel boyutlarını belirlemek için sadece Stage sınıfının `Width` ve `Height` özellikleri kullanılabilir. (Diğer ölçekleme modlarında, `stageWidth` ve `stageHeight` özellikleri her zaman SWF'nin orijinal genişliğini ve yüksekliğini yansıtır.) Ayrıca, `scaleMode` öğesi `StageScaleMode.NO_SCALE` olarak ayarlandığında ve SWF dosyası yeniden boyutlandırıldığında, Stage sınıfının `resize` olayı gönderilerek uygun şekilde ayarlamalar yapmanız sağlanır.

Sonuç olarak, `scaleMode` öğesinin `StageScaleMode.NO_SCALE` olarak ayarlanması, istediğiniz zaman pencere yeniden boyutlandırmasına ekran içeriklerinin nasıl ayarlanacağını daha fazla denetlemenizi sağlar. Örneğin, video ve denetim çubuğu içeren bir SWF dosyasında, Sahne Alanı yeniden boyutlandırıldığında denetim çubuğunun aynı boyutta kalmasını sağlamak ve yalnızca Sahne Alanı boyutundaki değişikliği barındıracak şekilde video penceresinin boyutunu değiştirmek isteyebilirsiniz. Aşağıdaki örnekte bu gösterilmektedir:

```
// 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;
}
```

Adobe AIR için sahne ölçek modu ayarlama

Sahne alanının `scaleMode` özelliği, sahne ölçeklerini belirler ve bir pencere yeniden boyutlandırıldığında alt öğe görüntü ekranlarını kırpar. AIR uygulamasında sadece `noScale` modu kullanılır. Bu modda sahne ölçeklendirilmez. Bunun yerine, sahnenin boyutu pencerenin sınırları içinde direkt olarak değişir. Pencere daha küçük bir boyuta yeniden boyutlandırılırsa nesneler kırpılabilir.

Sahne ölçek modülleri, sahnenin boyutunu veya boyut oranını her zaman kontrol edemeyeceğiniz Web tarayıcı gibi ortamlarda kullanılmak üzere tasarlanmıştır. Sahne alanı, uygulamanıza en uygun boyut veya boyut oranıyla eşleşmediği zaman bu boyut, en az ödünle seçim yapmanızı sağlar. AIR'de sahne alanının kontrolü daima sizdedir, bu yüzden çoğu durumda içeriğinizi yeniden düzenlediğinizde veya pencere boyutlarınızı ayarladığınızda sahne alanını ölçeklendirmeye kıyasla daha iyi sonuçlar elde edersiniz.

Tarayıcıda ve AIR başlangıç penceresinde, pencere boyutu ile başlangıç ölçek çarpanı arasındaki ilişki, yüklenen SWF dosyasından okunabilir. Ancak NativeWindow nesnesi oluşturduğunuz zaman AIR uygulaması, pencere boyutu ile 72:1 ölçek faktörü arasında nedensiz bir ilişki seçer. Bu nedenle, pencereniz 72x72 pikselse, pencereye eklenen 10x10 piksellik bir dikdörtgen, 10x10 piksellik doğru boyutu çizer. Diğer taraftan, pencere 144x144 pikselse o zaman 10x10 piksellik bir dikdörtgen, 20x20 piksele ölçeklendirilir. Bir pencere sahnesi için `noScale` yerine `scaleMode` kullanmakta ısrarcıysanız, penceredeki herhangi bir görüntü nesnesinin ölçek faktörlerini sahnenin geçerli genişliği ve yüksekliğinin 72 piksel oranında ayarlayarak durumu düzeltebilirsiniz. Örneğin aşağıdaki kod `client` isimli görüntü nesnesi için gerekli ölçek faktörünü hesaplar:

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

Not: *Flex ve HTML pencereleri sahneyi `scaleMode` öğesinden `noScale` öğesine otomatik olarak ayarlar. `scaleMode` öğesinin değiştirilmesi, bu tür pencerelerde kullanılan otomatik mizanpaj mekanizmalarını bozar.*

Tam ekran modunda çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tam ekran modu, herhangi bir kap kenarlığı veya menüler olmadan, bir filmin sahne alanını izleyicinin tüm monitörünü dolduracak şekilde ayarlamanıza olanak sağlar. SWF dosyası için tam ekran modunu etkinleştirme ve devre dışı bırakma arasında geçiş yapmak üzere, Stage sınıfının `displayState` özelliği kullanılır. `displayState` özelliği, flash.display.StageDisplayState sınıfındaki sabitler tarafından tanımlanan değerlerden birine ayarlanabilir. Tam ekran modunu açmak için, `displayState` özelliğini `StageDisplayState.FULL_SCREEN` olarak ayarlayın:

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

Tam ekran etkileşimli modunu açmak için (Flash Player 11.3'te yeni), `displayState` özelliğini `StageDisplayState.FULL_SCREEN_INTERACTIVE` olarak ayarlayın:

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Flash Player'da, tam ekran modu yalnızca fare tıklatmasına (sağ tıklatma dahil) veya tuş basışına yanıt olarak ActionScript üzerinden başlatılabilir. Uygulama güvenlik sanal alanında çalışan AIR içeriği için bir kullanıcı hareketine yanıt olarak tam ekran moduna girilmesi gerekmez.

Tam ekran modundan çıkmak için, `displayState` özelliğini `StageDisplayState.NORMAL` olarak ayarlayın.

```
stage.displayState = StageDisplayState.NORMAL;
```

Ayrıca kullanıcı, odağı farklı bir pencereye geçirerek veya çok sayıdaki tuş birleşimlerinden birini kullanarak (Esc tuşu (tüm platformlar), Control-W (Windows), Command-W (Mac) ya da Alt-F4 (Windows)) tam ekran modundan çıkmayı seçebilir.

Flash Player'da tam ekran modunu etkinleştirme

Bir HTML sayfasında gömülü SWF dosyası için tam ekran modunu etkinleştirmek üzere, Flash Player'ı gömmeye yönelik HTML kodunun aşağıdaki gibi, `allowFullScreen` adı ve `true` değeri ile bir `param` etiketini ve `embed` niteliğini içermesi gerekir:

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

Flash geliştirme aracında, Dosya -> Yayınlama Ayarları seçeneklerini belirleyin ve Yayınlama Ayarları iletişim kutusunun HTML sekmesinde Yalnızca Flash - Tam Ekran Moduna İzin Ver şablonunu seçin.

Flex'te, HTML şablonunun tam ekranı destekleyen `<object>` ve `<embed>` etiketlerini içerdiğinden emin olun.

SWF gömme etiketleri oluşturmak için bir web sayfasında JavaScript kullanıyorsanız, `allowFullScreen param` etiketini ve niteliğini eklemek için JavaScript'i değiştirmeniz gerekir. Örneğin, HTML sayfanız `AC_FL_RunContent()` işlevini (Flash Professional ve Flash Builder tarafından oluşturulan HTML sayfaları tarafından kullanılan) kullanıyorsa, aşağıdaki gibi bu işleve `allowFullScreen` parametresini eklemeniz gerekir:

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

Bu, bağımsız Flash Player'da çalışan SWF dosyaları için geçerli değildir.

Not: *Pencere Modu'nu (HTML'de wmode) Opak Penceresiz (opak) veya Saydam Penceresiz (saydam) olarak ayarlarsanız, tam ekran penceresi her zaman opak olur*

Tam ekran modunu tarayıcıda Flash Player ile kullanmaya yönelik güvenlikle ilgili kısıtlamalar da bulunmaktadır. Bu kısıtlamalar "Güvenlik" sayfa 1053 konusunda açıklanmıştır.

Flash Player 11.3 ve üst sürümlerinde tam ekran etkileşimli modunu etkinleştirme

Flash Player 11.3 ve üst sürümleri, tüm klavye tuşları (tam ekran etkileşimli moddan çıkmaya yarayan **Esc** tuşu hariç) için tam destek sağlayan tam ekran etkileşimli modu destekler. Tam ekran etkileşimli mod, oyun oynamak için kullanışlıdır (örneğin, çok oyunculu bir oyunda sohbeti etkinleştirmek veya birinci şahıs nişancı oyunlarında WASD klavye kontrollerini etkinleştirmek için)

Bir HTML sayfasında gömülü SWF dosyası için tam ekran etkileşimli modunu etkinleştirmek üzere, Flash Player'ı gömmeye yönelik HTML kodunun aşağıdaki gibi, `param` etiketi ve `embed` niteliği ile birlikte `allowFullScreenInteractive` adı ve `true` değerini içermesi gerekir:

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

Flash yazma aracında, Dosya -> Yayınlama Ayarları seçeneklerini belirleyin ve Yayınlama Ayarları iletişim kutusunun HTML sekmesinde Yalnızca Flash - Tam Ekran Etkileşimli Moduna İzin Ver şablonunu seçin.

Flash Builder ve Flex'te HTML şablonlarının, tam ekran etkileşimli modunu destekleyen `<object>` ve `<embed>` etiketlerini içerdiğinden emin olun.

SWF gömme etiketleri oluşturmak için bir web sayfasında JavaScript kullanıyorsanız, `allowFullScreenInteractive param` etiketini ve niteliğini eklemek üzere JavaScript'i değiştirmeniz gerekir. Örneğin, HTML sayfanız `AC_FL_RunContent()` işlevini (Flash Professional ve Flash Builder tarafından oluşturulan HTML sayfalarında kullanılan) kullanıyorsa, aşağıdaki gibi bu işlev çağrısına `allowFullScreenInteractive` parametresini eklemeniz gerekir:

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

Bu, bağımsız Flash Player'da çalışan SWF dosyaları için geçerli değildir.

Tam ekran sahne boyutu ve ölçekleme

`Stage.fullScreenHeight` ve `Stage.fullScreenWidth` özellikleri, hemen duruma girilirse, tam ekran boyutuna girilirken kullanılan monitör yüksekliğini ve genişliğini döndürür. Siz bu değerleri aldıktan sonra ancak tam ekran moduna girmeden önce, kullanıcının tarayıcıyı bir monitörden diğerine taşıma olanağı varsa, bu değerler yanlış olabilir. Bu değerleri `Stage.displayState` özelliğini `StageDisplayState.FULL_SCREEN` olarak ayarladığınız olay işleyicisinde alırsanız, değerler doğrudur. Birden çok monitöre sahip kullanıcılar için, SWF içeriği yalnızca bir monitörü dolduracak şekilde genişler. Flash Player ve AIR uygulaması, SWF'nin en büyük bölümünü hangi monitörün içerdiğini belirlemek için bir metrik kullanır ve tam ekran modu için bu monitörü kullanır. fullScreenHeight ve fullScreenWidth özellikleri yalnızca tam ekran modu için kullanılan monitör boyutunu gösterir. Ayrıntılı bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda bulunan `Stage.fullScreenHeight` ve `vStage.fullScreenWidth` bölümüne bakın.

Tam ekran modu için Sahne Alanı ölçekleme davranışı, normal moddakiyle aynıdır; ölçekleme, Stage sınıfının `scaleMode` özelliği tarafından denetlenir. `scaleMode` özelliği `StageScaleMode.NO_SCALE` olarak ayarlanırsa, Sahne Alanı'nın `stageWidth` ve `stageHeight` özellikleri, SWF tarafından kaplanan ekran alanının boyutunu (bu durumda, ekranın tamamı) yansıtacak şekilde değişir; tarayıcıda görüntülenirse, ayarı bu denetimlerin HTML parametreleri denetler.

Tam ekran modu etkinleştirildiğinde veya devre dışı bırakıldığında bunu algılayıp yanıt vermek için Stage sınıfının `fullScreen` olayını kullanabilirsiniz. Örneğin, bu örnekteki gibi, tam ekran moduna girerken veya tam ekran modundan çıkarken ekrandaki öğeleri yeniden konumlandırmak, ekrana öğe eklemek veya ekrandan öğe kaldırmak isteyebilirsiniz:

```
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);
```

Bu kodun da gösterdiği gibi, `fullScreen` olayının olay nesnesi, tam ekran modunun etkinleştirilip (`true`) etkinleştirilmediğini (`false`) belirten `fullScreen` özelliğini içeren bir flash.events.FullScreenEvent sınıfı örneğidir.

Tam ekran modunda klavye desteği

Flash Player bir tarayıcıda çalışırken, klavye olayları ve TextField örneklerindeki metin girişi gibi klavyeyle ilgili tüm ActionScript, tam ekran modunda devre dışı bırakılır. İstisnalar (etkin tuşlar) şunlardır:

- Seçili basılamayan tuşlar, ok tuşları, boşluk tuşu ve sekme tuşu
- Tam ekran modunu sonlandıran klavye kısayolları: Esc (Windows ve Mac), Control-W (Windows), Command-W (Mac) ve Alt-F4

Bu kısıtlamalar, bağımsız Flash Player veya AIR uygulamasında çalışan SWF içeriği için mevcut değildir. AIR, klavye girdisine izin veren etkileşimli bir tam ekran modunu destekler.

Tam ekran modunda fare desteği

Varsayılan olarak, tam ekran modundaki fare olayları aynen tam ekran modunda olunmadığı zamanki gibi çalışır. Ancak, tam ekran modunda fare kilidini etkinleştirmek için `Stage.mouseLock` özelliğini isteğe bağlı olarak ayarlayabilirsiniz. Fare kilidi, imleci devre dışı bırakır ve sınırsız fare hareketini etkinleştirir.

Not: *Fare kilidini, yalnızca masaüstü uygulamalar için tam ekran modunda etkinleştirebilirsiniz. Fare kilidinin tam ekran modunda olmayan uygulamalarda veya mobil aygıtlardaki uygulamalar için ayarlanması özel bir durum oluşturur.*

Şu durumlarda fare kilidi otomatik olarak devre dışı bırakılır ve fare imleci tekrar görünür hale gelir:

- Kullanıcı Escape tuşunu (tüm platformlar), Control-W (Windows) tuşlarını, Command-W (Mac) tuşlarını veya Alt-F4 (Windows) tuşlarını kullanarak tam ekran modundan çıktığında.
- Uygulama penceresi odağı kaybettiğinde.
- Tüm gizlilik iletişim kutuları da dahil olmak üzere ayar kullanıcı arabirimleri görünür olduğunda.
- Dosya yükleme iletişim kutusu gibi bir yerel iletişim kutusu görüntülendiğinde.

`mouseMove` olayı gibi fare hareketiyle ilişkilendirilmiş olaylar, olay nesnesini temsil etmek için MouseEvent sınıfını kullanır. Fare kilidi devre dışı durumdayken, farenin konumunu belirlemek için `MouseEvent.localX` ve `MouseEvent.localY` özelliklerini kullanın. Fare kilidi etkin durumdayken, fare konumunu belirlemek için `MouseEvent.movementX` ve `MouseEvent.movementY` özelliklerini kullanın. `movementX` ve `movementY` özellikleri, fare konumunun kesin koordinatlarını değil son olaydan itibaren fare konumunda gerçekleşmiş olan değişiklikleri içerir.

Tam ekran modunda donanım ölçekleme

Flash Player veya AIR uygulamasını sahne alanının belirli bir bölgesini tam ekran moduna ölçekleyecek şekilde ayarlamak için Stage sınıfının `fullScreenSourceRect` özelliğini kullanabilirsiniz. Flash Player ve AIR, varsa kullanıcının bilgisayarındaki grafik ve video kartını kullanarak donanımda ölçekler ve genellikle içeriği yazılım ölçeklemeden daha hızlı bir şekilde görüntüler.

Donanım ölçeklemeden yararlanmak için, sahne alanının tamamını veya bir kısmını tam ekran moduna ayarlarsınız. Şu ActionScript 3.0 kodu, sahne alanının tamamını tam ekran moduna ayarlar:

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

Bu özellik geçerli bir dikdörtgene ayarlandığında ve `displayState` özelliği tam ekran moduna ayarlandığında, Flash Player ve AIR belirtilen alanı ölçekler. ActionScript içinde piksel olarak gerçek Sahne Alanı boyutu değişmez. Flash Player ve AIR, "Tam ekran modundan çıkmak için Esc düğmesine basın" mesajını barındıracak şekilde dikdörtgenin boyutu için minimum bir sınırı zorlar. Bu sınır genellikle yaklaşık 260 x 30 pikseldir ancak platforma ve Flash Player sürümüne bağlı olarak değişiklik gösterebilir.

`fullScreenSourceRect` özelliği yalnızca Flash Player veya AIR tam ekran modunda olmadığında ayarlanabilir. Bu özelliği düzgün bir şekilde kullanmak için, öncelikle bu özelliği ayarlayın, ardından `displayState` özelliğini tam ekran moduna ayarlayın.

Ölçeklemeyi etkinleştirmek için, `fullScreenSourceRect` özelliğini bir dikdörtgen nesnesine ayarlayın.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

Ölçeklemeyi devre dışı bırakmak için, `fullScreenSourceRect` özelliğini `null` olarak ayarlayın.

```
stage.fullScreenSourceRect = null;
```

Flash Player ile tüm donanım ivmesi özelliklerinden yararlanabilmek için, özelliği Flash Player Ayarları iletişim kutusundan etkinleştirin. İletişim kutusunu yüklemek için, tarayıcınızdaki Flash Player içeriğinin üzerinde sağ tıklatın (Windows) veya Control tuşuna basarak tıklatın (Mac). Birinci sekme olan Görünüm sekmesini seçin ve Donanım ivmesini etkinleştir onay kutusunu tıklatın.

Doğrudan ve GPU birleştirme penceresi modları

Flash Player 10, doğrudan ve GPU birleştirme olmak üzere iki pencere modu sunar, bu pencereleri Flash geliştirme aracındaki yayın ayarlarından etkinleştirebilirsiniz. Bu modlar AIR uygulamasında desteklenmez. Bu modlardan yararlanabilmek için, Flash Player için donanım ivmesini etkinleştirmeniz gerekmektedir.

Doğrudan modu, grafikleri ekrana itmek için en hızlı ve en doğrudan yolu kullanır ve bu da video oynatımı için avantaj sağlar.

GPU birleştirme modu, birleştirme işlemini hızlandırmak için video kartında grafik işleme birimini kullanır. Video birleştirme, tek bir video görüntüsü oluşturmak için birden çok görüntü katmanlama işlemidir. GPU ile birleştirme hızlandırıldığında, YUV dönüştürme, renk düzeltme, döndürme veya ölçekleme ve karışım performansı artırılabilir. YUV dönüştürme, video kameraların ve ekranların kullandığı RGB (kırmızı, yeşil, mavi) renk modeline iletim için kullanılan bileşik analog sinyallerin renk dönüştürmesini ifade eder. Birleştirmeyi hızlandırmak için GPU'nun kullanılması, CPU'ya yüklenecek olan bellek ve hesaplama gereksinimini azaltır. Bu da standart tanımlı videoların daha akıcı şekilde oynatılmasını sağlar.

Bu pencere modlarını uygularken dikkatli olun. GPU birleştirmenin kullanılması bellek ve CPU kaynakları için maliyetli olabilir. Bazı işlemler (örn. karışım modları, filtre uygulama, kırpma veya maskeleme) GPU'da gerçekleştirilemezse, bu işlemler yazılım tarafından yapılır. Adobe, bu modları kullanırken her HTML sayfası için kendinizi bir SWF dosyasıyla sınırlamanızı ve bant reklamlar için bu modları etkinleştirmemenizi önerir. Flash uygulamasının Filmi Test Et işlevi, donanım ivmesini kullanmaz ancak Yayınla Önizle seçeneği üzerinden bu işlevi kullanabilirsiniz.

SWF dosyanızda maksimum ekran yenileme hızı olan 60'tan yüksek bir kare hızının ayarlanması gereksizdir. Kare hızının 50 ile 55 arasında bir değere ayarlanması, zaman zaman çeşitli nedenlerle oluşabilen bırakılan karelere neden olur.

Doğrudan modunu kullanabilmek için Windows'ta 128 MB VRAM ile Microsoft DirectX 9 ve Apple Macintosh, Mac OS X v10.2 veya daha üstü için OpenGL gerekmektedir. GPU birleştirme için Windows'ta 128 MB VRAM ile Microsoft DirectX 9 ve Pixel Shader 2.0 desteği gerekir. Mac OS X ve Linux'ta GPU birleştirme için OpenGL 1.5 ve birçok OpenGL uzantısı (framebuffer nesnesi, çoklu doku, gölgelendirici nesneleri, gölgeleme dili, parça gölgelendirici) gerekir.

Flash sekmesindeki Donanım İvmesi menüsünü kullanarak, Flash uygulamasının Yayın Ayarları iletişim kutusu üzerinden her SWF dosyasında `direct` ve `gpu` ivme modlarını etkinleştirebilirsiniz. Yok seçeneğini belirlerseniz, HTML sekmesindeki Pencere Modu ayarının belirttiği gibi, pencere modu `default`, `transparent` veya `opaque`, ayarına geri döndürülür.

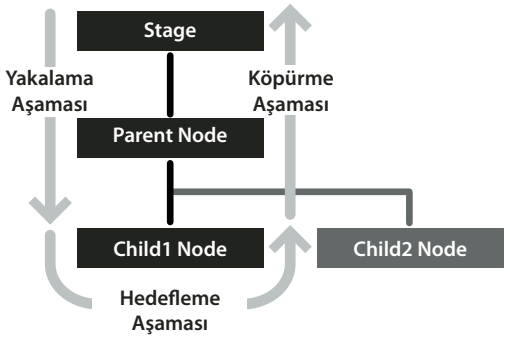
Görüntüleme nesneleri için olayları işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

DisplayObject sınıfı, EventDispatcher sınıfından miras alır. Başka bir deyişle, her görüntüleme nesnesi olay modeline tamamen katılabilir ("Olayları işleme" sayfa 126 bölümünde açıklanmıştır.) Her görüntüleme nesnesi, belirli bir olayı dinlemek için `addEventListener()` yöntemini (EventDispatcher sınıfından miras alınır) kullanabilir ancak yalnızca dinleme nesnesi o olayın olay akışının bir parçasıysa bu gerçekleşebilir.

Flash Player veya AIR bir olay nesnesi gönderdiğinde, söz konusu olay nesnesi Sahne Alanı'ndan olayın gerçekleştiği görüntüleme nesnesine doğru dairesel bir seyahatte bulunur. Örneğin, kullanıcı `child1` adında bir görüntüleme nesnesini tıklatırsa, Flash Player, Sahne Alanı'ndan görüntüleme listesi hiyerarşisinde `child1` görüntüleme nesnesine doğru aşağı yönde bir olay nesnesi gönderir.

Bu diyagramda gösterildiği gibi, olay akışı kavramsal olarak üç aşamaya ayrılır:



Daha fazla bilgi için "Olayları işleme" sayfa 126 bölümünü inceleyin.

Görüntüleme nesnesi olaylarıyla çalışırken dikkate alınması gereken önemli bir nokta, görüntüleme nesnelerinin, görüntüleme listesinden kaldırıldığında otomatik olarak bellekten de kaldırılıp (çöp toplama) kaldırılmaması üzerinde olay dinleyicilerinin etki sahibi olabilmesidir. Görüntüleme nesnesinin olaylarına dinleyici olarak abone olmuş nesneler varsa, söz konusu olay nesnesi halen bu dinleyici nesnelere başvuru içerdiğinden, görüntüleme listesinden kaldırılsa da bellekten kaldırılmaz. Daha fazla bilgi için, bkz. "Olay dinleyicilerini yönetme" sayfa 139.

DisplayObject alt sınıfı seçme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Belirlenebilecek birçok seçenek olduğundan, görüntüleme nesneleriyle çalışırken vereceğiniz önemli kararlardan biri, hangi görüntüleme nesnesini ne amaçla kullanacağınızdır. Aşağıda, karar vermenize yardımcı olacak birkaç yönerge verilmiştir. İster bir sınıf örneğine ihtiyacınız olsun, isterse oluşturduğunuz sınıf için bir temel sınıf seçiyor olun, bu önerilerin aynısı geçerlidir:

- Başka görüntüleme nesneleri için kap olabilecek bir nesneye ihtiyacınız yoksa (başka bir deyişle, bağımsız bir ekran öğesi görevi görecek bir nesneye ihtiyacınız varsa), ne için kullanılacağına bağlı olarak bu DisplayObject veya InteractiveObject alt sınıflarından birini seçin:
  - Bitmap görüntüsünü görüntülemek için Bitmap.
  - Metin eklemek için TextField.
  - Video görüntülemek için Video.
  - Ekran içeriği çizmek üzere "tuval" için Shape öğesi. Özellikle, ekranda şekil çizmek için bir örnek oluşturmak istiyorsanız ve bu örnek diğer görüntüleme nesneleri için bir kap olmayacaksa, Sprite veya MovieClip yerine Shape öğesini kullanmanız çok daha fazla performans avantajı elde etmenizi sağlayacak.
  - Flash geliştirme aracı tarafından oluşturulan öğeler için MorphShape, StaticText veya SimpleButton. (Programlama yoluyla bu sınıfların örneklerini oluşturamazsınız ancak Flash geliştirme aracı kullanılarak oluşturulmuş öğeleri ifade etmek için bu veri türleriyle değişkenler oluşturabilirsiniz.)
- Ana Sahne Alanı'nı ifade etmek için bir değişken gerekiyorsa, veri türü olarak Stage sınıfını kullanın.
- Harici bir SWF dosyasını veya görüntü dosyasını yüklemek için bir kap gerekiyorsa, Loader örneğini kullanın. Yüklenen içerik, Loader örneğinin bir alt öğesi olarak görüntüleme listesine eklenir. Veri türü ise, aşağıdaki gibi, yüklenen içeriğin doğasına bağlı olur:
  - Yüklenen görüntü bir Bitmap örneği olacaktır.
  - ActionScript 3.0'da yazılmış, yüklenen bir SWF dosyası, Sprite veya MovieClip örneği (ya da içerik oluşturucu tarafından belirtildiği şekilde bu sınıfların bir örneği veya alt sınıfı) olacaktır.
  - ActionScript 1.0 veya ActionScript 2.0'da yazılmış, yüklenen bir SWF dosyası, AVM1Movie örneği olacaktır.
- Diğer görüntüleme nesneleri için kap görevi görecek (ActionScript kullanarak görüntüleme nesnesi üzerinde çizim yapacak olsanız da olmasanız da) bir nesne gerekiyorsa, DisplayObjectContainer alt sınıflarından birini seçin:
  - Nesne yalnızca ActionScript kullanılarak oluşturulacaksa veya yalnızca ActionScript ile oluşturulup işlenecek özel bir görüntüleme nesnesinin temel sınıfı olarak oluşturulacaksa: Sprite.
  - Flash geliştirme aracında oluşturulmuş bir film klibini ifade etmek için değişken oluşturuyorsanız: MovieClip.
- Flash kütüphanesinde bir film klibi sembolüyle ilişkilendirilecek bir sınıf oluşturuyorsanız, sınıfınızın temel sınıfı olarak bu DisplayObjectContainer alt sınıflarından birini seçin:
  - İlişkilendirilen film klibi sembolü, birden çok kare üzerinde içerik barındırıyorsa: MovieClip.
  - İlişkilendirilen film klibi sembolü, yalnızca birinci karede içerik barındırıyorsa: Sprite.

Görüntüleme nesnelerini işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Hangi görüntüleme nesnesini kullanmayı seçtiğiniz dikkate alınmaksızın, ekranda görüntülenen öğeler olarak tüm görüntüleme nesnelerinin ortak şekilde sahip olduğu birkaç işleme vardır. Örneğin, bunların tümü ekranda konumlandırılabilir, görüntüleme nesnelerinin yığınlama sırasında ileri veya geri taşınabilir, ölçeklenebilir, döndürülebilir, vb. işlemler yapılabilir. Tüm görüntüleme nesneleri bu işlevleri ortak temel sınıflarından (DisplayObject) aldığından, TextField örneğini, Video örneğini, Shape örneğini ya da başka bir görüntüleme nesnesini de işleseniz, bu işlevler aynı şekilde davranır. İlerleyen bölümlerde bu ortak görüntüleme nesnesi işlemelerinin birçoğu ayrıntılı şekilde ele alınmaktadır.

Konum değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Herhangi bir görüntüleme nesnesinin en temel işlenmesi, ekrandaki konumunun değiştirilmesidir. Görüntüleme nesnesinin konumunu ayarlamak için, nesnenin `x` ve `y` özelliklerini değiştirin.

```
myShape.x = 17;
myShape.y = 212;
```

Görüntüleme nesnesi konumlandırma sistemi, Sahne Alanı'nı Kartezyen koordinat sistemi (yatay x ekseni ve dikey y eksenine sahip ortak ızgara sistemi) olarak değerlendirir. Koordinat sisteminin başlangıç noktası (x ve y eksenlerinin buluştuğu 0,0 koordinatı), Sahne Alanı'nın sol üst köşesindedir. Bu noktadan itibaren, x değerleri pozitif olarak sağa ve negatif olarak sola giderken (tipik grafik sistemlerinin tersine), y değerleri pozitif olarak aşağı ve negatif olarak yukarı gider. Örneğin, kodun önceki satırları `myShape` nesnesini 17 x koordinatına (başlangıç noktasının 17 piksel sağına) ve 212 y koordinatına (başlangıç noktasının 212 piksel aşağısına) taşır.

Varsayılan olarak, ActionScript kullanılarak bir görüntüleme nesnesi oluşturulduğunda, `x` ve `y` özelliklerinin her ikisi de 0 değerine ayarlanır ve böylece nesne, üst içeriğinin sol üst köşesine yerleştirilir.

Sahne Alanı'na göre konum değiştirme

`x` ve `y` özelliklerinin her zaman, üst görüntüleme nesnesinin eksenlerinin 0,0 koordinatına göre görüntüleme nesnesinin konumunu ifade ettiğinin unutulmaması önemlidir. Bu nedenle de, bir Sprite örneğinin içinde yer alan bir Shape örneği (örn. bir daire) için, Shape nesnesinin `x` ve `y` özelliklerinin 0 olarak ayarlanması sonucunda daire Sprite öğesinin sol üst köşesine yerleştirilir ve bunun her zaman Sahne Alanı'nın sol üst köşesi olması gerekmez. Bir nesnesi genel Sahne Alanı koordinatlarına göre konumlandırmak için, koordinatları genel (Sahne Alanı) koordinatlardan yerel (görüntüleme nesnesi kabı) koordinatlarına dönüştürmek için aşağıdaki gibi, herhangi bir görüntüleme nesnesinin `globalToLocal()` yöntemini kullanabilirsiniz:

```
// 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;
```

Aynı şekilde, yerel koordinatları Sahne Alanı koordinatlarına dönüştürmek için de DisplayObject sınıfının `localToGlobal()` yöntemini kullanabilirsiniz.

Görüntü nesnelerini fare ile taşıma

ActionScript'te bir kullanıcının fareyi kullanarak görüntü nesnelerini hareket ettirmesi iki teknikle sağlanır. Her iki durumda da iki fare olayı kullanılır; fare düğmesine basıldığında, nesneye fare imlecini takip etmesi bildirilir ve düğme serbest bırakıldığında, nesneye fare imlecini takip etmeyi durdurması bildirilir.

Not: *Flash Player 11.3 ve üst sürümleri, AIR 3.3 ve üst sürümleri: Ayrıca kullanıcının fare düğmesini öğeyi bulunduran Grafik Sayfasının sınırları dışında bırakması gibi durumlar için MouseEvent.RELEASE_OUTSIDE olayını da kullanabilirsiniz.*

Birinci teknik olan `startDrag()` yöntemi, daha kolay ancak sınırlıdır. Fare düğmesine basıldığında, sürüklenecek görüntüleme nesnesinin `startDrag()` yöntemi çağrılır. Fare düğmesi serbest bırakıldığında, `stopDrag()` yöntemi çağrılır. Bu iki işlev, Sprite sınıfında tanımlandığı için hareket eden nesne bir Sprite ya da onun alt sınıflarından biri olmalıdır.

```
// 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);
```

Bu teknik önemli ölçüde bir sınırlama içerir: `startDrag()` kullanılarak aynı anda yalnızca bir öğe sürüklenebilir. Yalnızca bir görüntüleme nesnesi sürüklenirken `startDrag()` yöntemi başka bir görüntüleme nesnesinde çağrılırsa, birinci görüntüleme nesnesi fareyi takip etmeyi hemen durdurur. Örneğin, `startDragging()` işlevi burada gösterildiği gibi değiştirilirse, `square.startDrag()` yöntemi çağrısına rağmen yalnızca `circle` nesnesi sürüklenir:

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

`startDrag()` kullanılarak yalnızca bir nesnenin sürüklenebilmesinin sonucunda, `stopDrag()` yöntemi herhangi bir görüntüleme nesnesinde çağrılabilir ve geçerli olarak sürüklenen nesneyi durdurur.

Birden çok görüntüleme nesnesi sürüklemeniz gerekiyorsa veya birden çok nesnenin `startDrag()` yöntemini kullanması durumunda çakışma oluşması olasılığını önlemek için, en iyisi sürükleme etkisini oluşturmak için fareyi takip etme tekniğinin kullanılmasıdır. Bu teknikte, fare düğmesine basıldığında, bir işlev, Sahne Alanı'nın `mouseMove` olayına dinleyici olarak abone olur. Daha sonra fare her hareket ettiğinde çağrılacak olan bu işlev, sürüklenen nesnenin, farenin x, y koordinatına atlamasına neden olur. Fare düğmesi serbest bırakıldıktan sonra, işlevin dinleyici olarak aboneliği kaldırılır, başka bir deyişle, artık fare hareket ettiğinde bu işlev çağrılmaz ve nesne fare imlecini takip etmeyi durdurur. Aşağıda, bu tekniği gösteren birkaç kod yer almaktadır:

```
// 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);
```

Görüntüleme nesnesinin fare imlecini takip etmesini sağlamaya ek olarak, sürüklenen nesneyi görüntünün önüne taşımak çoğu zaman tercih edildiği için diğer tüm nesnelerin yukarısında kayıyor gibi görünür. Örneğin bir daire ve bir kare gibi her ikisi de fareyle taşınabilir iki nesneniz olduğunu varsayın. Görüntüleme listesinde daire karenin aşağısında bulunursa ve siz de imlecin karenin üzerine gelmesi için daireyi tıklatıp sürüklerseniz, daire karenin arkasından kayıyor gibi görünür ve bu da sürükleyip bırakma görüntüsünü bozar. Bunun yerine, daire tıklatıldığında görüntüleme listesinin üst kısmına gidecek şekilde ayarlama yapabilir ve böylece dairenin diğer tüm içeriklerin en üstünde görüntülenmesini sağlayabilirsiniz.

Aşağıdaki kod (önceki örnekten alınmış) fareyle taşınacak iki görüntü nesnesi sağlar; bir daire ve bir kare. Fare düğmesine her basıldığında, öğe Sahne Alanı'nın görüntüleme listesinin en üst kısmına taşınır, böylece sürüklenen öğe her zaman en üstte görüntülenir. (Yeni olan veya önceki listeden değiştirilen kod kalın yazı tipinde görüntülenir.)

```
// 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);
```

Bu etkiyi daha fazla genişletmek için (örn. kümeler arasında işaret veya kartların taşındığı bir oyun için), "alındığında" sürüklenen nesneyi Sahne Alanı'nın görüntüleme listesine ekleyebilir ve daha sonra fare düğmesi serbest bırakıldığında bunu başka bir listeye (bırakıldığı "küme" gibi) ekleyebilirsiniz.

Son olarak, etkiyi geliştirmek için, tıklatıldığında (sürüklemeye başladığınızda) görüntüleme nesnesine bir gölge filtresi ekleyebilir ve nesne serbest bırakıldığında gölgeyi kaldırabilirsiniz. ActionScript'te gölge filtresinin ve diğer görüntüleme nesnesi filtrelerinin kullanılmasıyla ilgili ayrıntılar için, bkz. “Görüntüleme nesnelerine filtre uygulama” sayfa 270.

Görüntüleme nesnelerini yatay ve dikey kaydırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntülemek istediğiniz alana sığamayacak kadar büyük olan bir görüntüleme nesneniz varsa, görüntüleme nesnesinin görüntülenebilir alanını tanımlamak için `scrollRect` özelliğini kullanabilirsiniz. Ayrıca, kullanıcı girdisine yanıt olarak `scrollRect` özelliğini değiştirerek içeriğin sola ve sağa veya yukarı ve aşağı kaydırılmasını sağlayabilirsiniz.

`scrollRect` özelliği, bir dikdörtgen alanını tek bir nesne olarak tanımlamak için gerekli değerleri birleştiren bir sınıf niteliğindeki Rectangle sınıfının bir örneğidir. Görüntüleme nesnesinin görüntülenebilir alanını başlangıçta tanımlamak için, yeni bir Rectangle örneği oluşturun ve bunu görüntüleme nesnesinin `scrollRect` özelliğine atayın. Daha sonra, dikey veya yatay kaydırma için, `scrollRect` özelliğini ayrı bir Rectangle değişkenine yazar ve istediğiniz özelliği değiştirirsiniz. (Örneğin, yatay kaydırma için Rectangle örneğinin `x` özelliğini veya dikey kaydırma için `y` özelliğini değiştirirsiniz.) Bunun ardından, değiştirilen değeri görüntüleme nesnesine bildirmek için Rectangle örneğini `scrollRect` özelliğine yeniden atarsınız.

Örneğin, aşağıdaki kod, SWF dosyasının sınırları içine sığamayacak kadar uzun olan `bigText` adında bir TextField nesnesi için görüntülenebilir alanı tanımlar. `up` ve `down` adındaki iki düğme tıklatıldığında, bunlar `scrollRect` Rectangle örneğinin `y` özelliğini değiştirerek TextField nesnesinin içeriklerinin yukarı veya aşağı kaydırılmasına neden olan işlevleri çağırır.

```
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);
```

Bu örnekte de gösterildiği gibi, bir görüntüleme nesnesinin `scrollRect` özelliğiyle çalıştığınızda, en iyisi `cacheAsBitmap` özelliğini kullanarak Flash Player veya AIR uygulamasının, görüntüleme nesnesinin içeriğini bitmap olarak önbelleğe alması gerektiğinin belirtilmesidir. Bunu yaptığınızda, Flash Player veya AIR uygulamasının, görüntüleme nesnesi her kaydırıldığında görüntüleme nesnesinin içeriklerinin tamamını yeniden çizmesi gerekmez ve bunun yerine gerekli bölümü doğrudan ekranda oluşturmak için, önbelleğe alınan bitmap'i kullanması yeterli olur. Ayrıntılar için, bkz. "Görüntüleme nesnelerini önbelleğe alma" sayfa 185.

Boyut ve ölçekleme nesnelerini işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesinin boyutunu iki şekilde ölçüp işleyebilirsiniz: boyut özelliklerini (`width` ve `height`) veya ölçek özelliklerini (`scaleX` ve `scaleY`) kullanarak.

Her görüntüleme nesnesi, başlangıçta piksel olarak nesnenin boyutuna ayarlanan bir `width` özelliğine ve `height` özelliğine sahiptir. Görüntüleme nesnesinin boyutunu ölçmek için bu özelliklerin değerlerini okuyabilirsiniz. Ayrıca aşağıdaki gibi, nesnenin boyutunu değiştirmek için yeni değerler de belirtebilirsiniz:

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

Görüntüleme nesnesinin `height` veya `width` öğesi değiştirildiğinde, nesnenin ölçeklenmesi sağlanır, başka bir deyişle, nesne yeni alana sığacak şekilde genişletilir ya da daraltılır. Görüntüleme nesnesi yalnızca vektör şekilleri içeriyorsa, bu şekiller herhangi bir kalite kaybı olmadan yeni ölçekte yeniden çizilir. Görüntüleme nesnesindeki tüm bitmap grafik öğeleri yeniden çizilmek yerine ölçeklenir. Bu nedenle, örneğin genişlik ve uzunluğu görüntüdeki piksel bilgilerinin gerçek boyutlarından fazlasına artırılan dijital bir fotoğraf pikselleştirilerek fotoğrafın tırtıklı görünmesi sağlanır.

Görüntüleme nesnesinin `width` veya `height` özelliklerini değiştirdiğinizde, Flash Player ve AIR, nesnenin `scaleX` ve `scaleY` özelliklerini de günceller.

Not: *TextField nesneleri, bu ölçekleme davranışına bir istisnadır. Metin alanlarının, metin kaydırma ve font boyutlarını barındıracak şekilde kendini yeniden boyutlandırması gerekir; dolayısıyla metin alanları, yeniden boyutlandırmadan sonra scaleX veya scaleY değerlerini 1 olarak sıfırlar. Ancak, TextField nesnesinin scaleX veya scaleY değerlerini ayarlarsanız, genişlik ve yükseklik değerleri, sağladığınız ölçekleme değerlerini alacak şekilde değiştirilir.*

Bu özellikler, görüntüleme nesnesinin orijinal boyuta göre göreceli boyutunu temsil eder. `scaleX` ve `scaleY` özellikleri, yüzdeyi temsil etmek için kesirli (ondalık) değerleri kullanır. Örneğin, bir görüntüleme nesnesinin `width` öğesi, orijinal boyutunun yarısı kadar olacak şekilde değiştirilirse, nesnenin `scaleX` özelliği yüzde 50 anlamına gelen `.5` değerine sahip olur. Yüksekliği iki katına çıkarılırsa, `scaleY` özelliği yüzde 200 anlamına gelen `2` değerine sahip olur.

```
// 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
```

Boyut değişiklikleri orantısaldır. Başka bir deyişle, bir karenin `height` öğesini değiştirir ancak `width` öğesini değiştirmezseniz, karenin oranları aynı olmaz ve bu artık bir kare değil dikdörtgen olur. Görüntüleme nesnesinin boyutunda göreceli değişiklikler yapmak isterseniz, nesneyi yeniden boyutlandırmak için `width` veya `height` özelliklerini ayarlamaya alternatif olarak, `scaleX` ve `scaleY` özelliklerinin değerleri ayarlayabilirsiniz. Örneğin, bu kod, `square` adındaki görüntüleme nesnesinin `width` özelliğini değiştirir ve sonra karenin orantısı bozulmayacak şekilde yatay ölçekle eşleşmesi için dikey ölçeği (`scaleY`) değiştirir.

```
// 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;
```

Ölçekleme sırasında deformasyonu denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Normalde, bir görüntüleme nesnesi ölçeklendiğinde (örneğin, yatay olarak genişletildiğinde), ortaya çıkan deformasyon nesneye eşit şekilde yayılır, böylece her bölüm eşit miktarda genişletilir. Grafik ve tasarım öğeleri için büyük ihtimalle istediğiniz budur. Ancak bazen, görüntüleme nesnesinin hangi bölümlerinin genişleyip hangi bölümlerinin değişmeden kalacağını denetlemeyi tercih edebilirsiniz. Buna verilebilecek yaygın bir örnek, yuvarlak köşeli dikdörtgen şeklinde bir düğmedir. Normal ölçeklemede, düğmenin köşeleri genişler ve düğme yeniden boyutlandırıldıkça köşe yarıçapı değişir.

Ancak bu durumda, görünebilir bir deformasyon olmadan ölçeklemenin gerçekleşmesi için (ölçeklenmesi gereken (düz kenarlar ve orta kısım) ve ölçeklenmemesi gereken belirli alanları (köşeler) belirleyebilmek için), ölçekleme üzerinde denetim elde edilmesi istenir.

Nesnelerin ölçeklenme şekli üzerinde denetim sahibi olduğunuz görüntüleme nesneleri oluşturmak için, 9 dilimli ölçeklemeyi (Ölçek 9) kullanabilirsiniz. 9 dilimli ölçeklemeyle, görüntüleme nesnesi dokuz ayrı dikdörtgene bölünür (tic-tac-toe panosunun ızgarası gibi 3/3 ızgara). Dikdörtgenlerin aynı boyutta olması gerekmez—ızgara çizgilerinin nereye yerleştirileceğini siz atarsınız. Görüntüleme nesnesi ölçeklendiğinde, dört köşe dikdörtgeninde (örn. düğmenin yuvarlak köşeleri) bulunan içerik genişletilmez veya sıkıştırılmaz. Üst orta ve alt orta dikdörtgenleri dikey olarak değil yatay olarak ölçeklenirken, sol orta ve sağ orta dikdörtgenleri yatay olarak değil dikey olarak ölçeklenir. Orta dikdörtgen hem yatay hem de dikey olarak ölçeklenir.

Bunu göz önünde bulundurarak, bir görüntüleme nesnesi oluşturuyorsanız ve belirli bir içeriğin asla ölçeklenmemesini istiyorsanız, 9 dilimli ölçekleme ızgarasının bölme çizgilerinin, köşe dikdörtgenlerinin birinde içerik sona erecek şekilde yerleştirildiğinden emin olmanız gerekir.

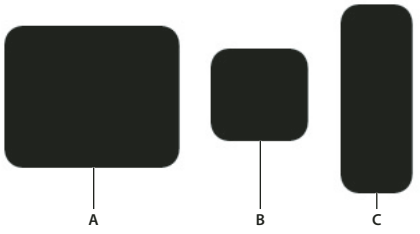
ActionScript'te, görüntüleme nesnesinin `scale9Grid` özelliği için bir değer ayarlandığında, nesne için 9 dilimli ölçekleme etkinleştirilir ve nesnenin Ölçek 9 ızgarasında dikdörtgenlerin boyutu tanımlanır. Aşağıdaki gibi, `scale9Grid` özelliğinin değeri olarak bir Rectangle sınıfı örneğini kullanırsınız:

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

Rectangle yapıcısının dört parametresi, x koordinatı, y koordinatı, genişlik ve yükseklikdir. Bu örnekte, dikdörtgenin sol üst köşesi, `myButton` adındaki görüntüleme nesnesinde x: 32, y: 27 noktasına yerleştirilir. Dikdörtgen 71 piksel genişlikte ve 64 piksel uzunluktadır (Böylece sağ kenarı görüntüleme nesnesindeki 103 x koordinatında ve alt kenarı da görüntüleme nesnesindeki 92 y koordinatındadır.)

Rectangle örneğinin tanımladığı bölgede bulunan gerçek alan, Ölçek 9 ızgarasının orta dikdörtgenini temsil eder. Diğer dikdörtgenler, burada gösterildiği gibi, Rectangle örneğinin kenarları genişletilerek Flash Player ve AIR tarafından hesaplanır:

Bu durumda, düğme yukarı veya aşağı ölçeklendikçe, yuvarlak köşeler genişletilmez veya daraltılmaz ancak diğer alanlar ölçeklemeyi barındıracak şekilde ayarlanır.



A. *myButton.width = 131;myButton.height = 106;* ***B.*** *myButton.width = 73;myButton.height = 69;* ***C.*** *myButton.width = 54;myButton.height = 141;*

Görüntüleme nesnelerini önbelleğe alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İster bir uygulama, isterse karmaşık komut dosyaları yazılmış animasyonlar oluşturun, Flash uygulamasındaki tasarımlarınızın boyutu arttıkça, performans ve en iyileştirmeyi göz önünde bulundurmanız gerekir. Statik kalan bir içeriğiniz (örn. dikdörtgen bir Shape örneği) olduğunda, Flash Player ve AIR uygulaması içeriği en iyileştirmez. Bu nedenle, dikdörtgenin konumunu değiştirdiğinizde, Flash Player veya AIR uygulaması Shape örneğinin tamamını yeniden çizer.

SWF dosyanızın performansını artırmak için, belirtilen görüntüleme nesnelerini önbelleğe alabilirsiniz. Görüntüleme nesnesi, bir *yüzeydir*, temel olarak örneğin vektör verilerinin bir bitmap sürümü olup SWF dosyanızın akışı sırasında çok da değiştirmek istemediğiniz verilerdir. Bu nedenle, önbelleğe almanın etkinleştirilmiş olduğu örnekler, SWF dosyası oynatıldıkça sürekli olarak yeniden çizilmez, bu da SWF dosyasının hızlı şekilde oluşturulmasını sağlar.

Not: *Vektör verilerini güncelleyebilirsiniz, bu sürede yüzey yeniden oluşturulur. Dolayısıyla, yüzeyde önbelleğe alınan vektör verilerinin SWF dosyasının tamamında aynı kalması gerekmez.*

Görüntüleme nesnesinin `cacheAsBitmap` özelliği `true` değerine ayarlandığında, görüntüleme nesnesinin kendi bitmap temsilini önbelleğe alması sağlanır. Flash Player veya AIR uygulaması, örnek için bir yüzey nesnesi oluşturur ve bu nesne vektör verisi yerine önbelleğe alınmış bir bitmap'tir. Görüntüleme nesnesinin sınırlarını değiştirirseniz, yüzey yeniden boyutlandırılmaz, yeniden oluşturulur. Yüzeyler diğer yüzeylerin içinde yuvalanabilir. Alt yüzey, bitmap'ini üst yüzeyinin üzerine kopyalar. Daha fazla bilgi için, bkz. "Bitmap önbelleğe almayı etkinleştirme" sayfa 187.

DisplayObject sınıfının `opaqueBackground` özelliği ile `scrollRect` özelliği, `cacheAsBitmap` özelliği kullanılarak bitmap'in önbelleğe alınmasıyla ilgilidir. Bu üç özellik birbirinden bağımsız olsa da, `opaqueBackground` ve `scrollRect` özellikleri, bir nesne önbelleğe alındığında en iyi şekilde çalışır—yalnızca `cacheAsBitmap` öğesini `true` değerine ayarladığınızda `opaqueBackground` ve `scrollRect` özellikleri için performans avantajı elde edersiniz. Görüntüleme nesnesi içeriğini kaydırma hakkında daha fazla bilgi için, bkz. "Görüntüleme nesnelerini yatay ve dikey kaydırma" sayfa 181. Opak bir arka plan ayarlama hakkında daha fazla bilgi almak için, bkz. "Opak bir arka plan rengi ayarlama" sayfa 188.

`cacheAsBitmap` özelliğini `true` değerine ayarlamanızı gerektiren, alfa kanalı maskeleme hakkında bilgi almak için, bkz. "Görüntüleme nesnelerini maskeleme" sayfa 193.

Önbelleğe alma ne zaman etkinleştirilmeli

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesi için önbelleğe alma etkinleştirildiğinde bir yüzey oluşturulur ve bu da karmaşık vektör animasyonlarının daha hızlı oluşturulmasına yardımcı olma gibi birçok avantaj sağlar. Önbelleği etkinleştirmek isteyeceğiniz birçok senaryo vardır. SWF dosyalarınızın performansını artırmak için önbelleğe almayı her zaman etkinleştirmek isteyeceğiniz düşünülebilir ancak önbelleğe almanın etkinleştirilmesinin performansı artırmadığı, hatta düşürdüğü durumlar da vardır. Bu bölümde, önbelleğe almanın kullanılması gereken senaryolar açıklanmakta ve normal görüntüleme nesnelerinin ne zaman kullanılması gerektiği anlatılmaktadır.

Önbelleğe alınan verilerin genel performansı, örneğinizin vektör verilerinin ne kadar karmaşık olduğuna, verinin ne kadarını değiştirmek istediğinize ve `opaqueBackground` özelliğini ayarlayıp ayarlamamanıza bağlıdır. Küçük bölgeleri değiştiriyorsanız, yüzey kullanma ile vektör verilerini kullanma arasındaki fark göz ardı edilebilir. Uygulamayı konuşlandırmadan önce çalışmanızla iki senaryoyu da test etmek isteyebilirsiniz.

Bitmap önbelleğe alma ne zaman kullanılmalı

Aşağıda, bitmap önbelleğe almayı etkinleştirdiğinizde önemli avantajlar elde edebileceğiniz tipik senaryolara yer verilmiştir.

- Karmaşık·arka·plan·görüntüsü:·Vektör·verilerinin·ayrıntılı·ve·karmaşık·bir·arka·plan·görüntüsünü·içeren·bir·uygulama·(belki·de·bitmap'i izle komutunu uyguladığınız bir görüntü ya da Adobe Illustrator® uygulamasında oluşturduğunuz bir resim). Karakterlere arka plan üzerinde animasyon uygulayabilirsiniz, arka planın devamlı olarak vektör verilerini yeniden oluşturması gerektiğinden bu animasyonu yavaşlatır. Performansı artırmak için, arka plan görüntüleme nesnesinin `opaqueBackground` özelliğini `true` değerine ayarlayabilirsiniz. Arka plan bitmap olarak oluşturulur ve hızlı şekilde yeniden çizilebilir, böylece animasyonunuz da daha hızlı oynatılır.
- Metin alanını kaydırma: Kaydırılan metin alanında büyük miktarda metin görüntüleyen bir uygulama. Kaydırma sınırlarıyla (`scrollRect` özelliği) metin alanını kaydırılabilir olarak ayarladığınız bir görüntüleme nesnesine yerleştirebilirsiniz. Böylece belirtilen örnek için hızlı piksel kaydırması sağlanır. Kullanıcı bir görüntüleme nesnesi örneğini kaydırdığında, Flash Player veya AIR uygulaması kaydırılan pikselleri yukarı kaydırır ve metin alanının tamamını yeniden oluşturmak yerine yeni gösterilen bölgeyi oluşturur.

- Pencereleme sistemi: Örtüşen pencerelerin karmaşık sistemini içeren bir uygulama. Her pencere açılabilir veya kapatılabilir (örneğin, web tarayıcısı pencereleri). Pencerelerin her birini bir yüzey olarak işaretlerseniz (`cacheAsBitmap` özelliğini `true` değerine ayarlayarak), her pencere ayrılır ve önbelleğe alınır. Kullanıcılar, pencereleri birbiriyle örtüşecek şekilde sürükleyebilir ve pencerelerin her birinin vektör içeriğini yeniden oluşturması gerekmez
- Alfa kanalı maskeleme: Alfa kanalı maskelemeyi kullandığınızda, `cacheAsBitmap` özelliğini `true` değerine ayarlamanız gerekir. Daha fazla bilgi için, bkz. "Görüntüleme nesnelerini maskeleme" sayfa 193.

Bu senaryoların tümünde bitmap önbelleğe almanın etkinleştirilmesi, vektör grafiklerini en iyileştirerek uygulamanın yanıt verme hızını artırıp etkileşimini geliştirir.

Ayrıca bir görüntüleme nesnesine her filtre uyguladığınızda, `cacheAsBitmap` öğesini açıkça `false` değerine ayarlasanız da bu öğe otomatik olarak `true` değerine ayarlanır. Görüntüleme nesnesinden tüm filtreleri temizlerseniz, `cacheAsBitmap` özelliği en son ayarlandığı değere döndürülür.

Bitmap önbelleğe almayı kullanmaktan ne zaman kaçınmalı

Bu özelliğin yanlış koşullarda kullanılması SWF dosyanızın performansını olumsuz yönde etkileyecektir. Bitmap önbelleğe almayı kullanırken şu yönergeleri unutmayın:

- Yüzeyleri aşırı kullanmayın (önbellek etkinleştirilmiş görüntüleme nesneleri). Her yüzey normal bir görüntüleme nesnesinden daha fazla bellek tüketir, başka bir deyişle, yalnızca oluşturma performansını artırmanız gerektiğinde yüzeyleri etkinleştirmeniz gerekir.

  Önbelleğe alınan bir bitmap, normal bir görüntüleme nesnesinden çok daha fazla bellek kullanabilir. Örneğin, Sahne Alanı'ndaki bir Sprite örneğinin boyutu 250 piksel x 250 pikselse, bu örnek önbelleğe alındığında, normal (önbelleğe alınmamış) bir Sprite örneği olduğunda kullandığı 1 KB yerine 250 KB bellek kullanabilir.
- Önbelleğe alınan yüzeyleri yakınlaştırmaktan kaçının Bitmap önbelleğe almayı fazla kullanırsanız, özellikle içeriğin üzerine yakınlaştırırsanız çok miktarda bellek kullanılır. (Bir önceki madde imine bakın.)
- Büyük ölçüde statik olan (animasyon uygulanmamış) görüntüleme nesnesi örnekleri için yüzeyleri kullanın. Örneği sürükleyebilir veya taşıyabilirsiniz ancak örneğin içeriklerine çok fazla animasyon uygulanmaması veya içeriklerin değiştirilmemesi gerekir. (Animasyon veya değişen içerik, animasyon ya da Video örneği içeren bir MovieClip örneğine çok benzer.) Örneğin, bir örneği döndürürseniz veya dönüştürürseniz, örnek, yüzey ve vektör verisi arasında değişir, bunu işlemek zordur ve SWF dosyanızı olumsuz yönde etkiler.
- Yüzeyleri vektör verileriyle karıştırırsanız, Flash Player ve AIR (ve bazen de bilgisayar) için gerekli işleme miktarı artırılır. Yüzeyleri olabildiğince (örneğin, pencereleme uygulamaları oluşturduğunuzda) bir arada gruplayın.
- Grafikleri sık aralıklarla değişen nesneleri ön belleğe almayın. Görüntüleme nesnesi her ölçeklendirildiğinde, eğildiğinde, döndürüldüğünde, alfa veya renk dönüştürme değeri değiştirildiğinde, görüntülenen nesnelerin alt öğeleri taşındığında veya grafik özelliği kullanılarak çizim yapıldığında, bitmap ön belleği yeniden çizilir. Bunun her karede olması durumunda, çalışma zamanının nesneyi bir bitmap'e çizmesi ve ardından bu bitmap'i sahne alanına kopyalaması gerekir -ki bu da sadece ön belleğe alınmamış nesnenin sahne alanına çizilmesine kıyasla fazladan çalışmaya neden olur. Güncelleme sıklığı ile ön belleğe alma arasındaki performans farkı görüntü nesnesinin karmaşıklığına ve boyutuna bağlıdır ve sadece belirli bir içerik test edilerek belirlenebilir.

Bitmap önbelleğe almayı etkinleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir görüntüleme nesnesi için bitmap önbelleğe almayı etkinleştirmek üzere, nesnenin `cacheAsBitmap` özelliğini `true` değerine ayarlarsınız:

```
mySprite.cacheAsBitmap = true;
```

`cacheAsBitmap` özelliğini `true` değerine ayarladıktan sonra, görüntüleme nesnesinin otomatik olarak pikselleri koordinatların tamamına birleştirdiğini fark edebilirsiniz. SWF dosyasını test ettiğinizde, karmaşık bir vektör görüntüsünde gerçekleştirilen herhangi bir animasyonun daha hızlı şekilde oluşturulduğunu görmeniz gerekir.

`cacheAsBitmap` öğesi `true` ayarına getirilse de, aşağıdakilerden biri veya birkaçı oluştuğunda bir yüzey (önbelleğe alınmış bitmap) oluşturulmaz:

- Bitmap, en az 2880 piksel yükseklikte veya genişlikte olduğunda.
- Bitmap bellek ayırmayı başaramadığında (yetersiz bellek hatası nedeniyle).

Ön bellekteki bitmap dönüştürme matrisleri

Adobe AIR 2.0 ve üstü (taşınabilir profil)

Taşınabilir cihazlar için AIR uygulamalarında, `cacheAsBitmap` özelliğini her ayarladığınızda `cacheAsBitmapMatrix` özelliğini de ayarlamanız gerekir. Bu özelliği ayarlamanız daha geniş bir aralıktaki dönüştürmeleri, yeniden oluşturmayı tetiklemeden, görüntü nesnesine uygulayabilmenizi sağlar.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

Bu matrix özelliğini ayarladığınızda aşağıdaki ek dönüştürmeleri de görüntü nesnesini yeniden ön belleğe almadan, nesneye uygulayabilirsiniz.

- Piksel yapışması olmadan taşıma veya çevirme
- Döndürme
- Ölçekleme
- Eğme
- Alfa değerini değiştirme (%0 ila 100 arasında saydamlık)

Bu dönüştürmeler ön belleğe alınan bitmap'e doğrudan uygulanır.

Opak bir arka plan rengi ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesi için opak bir arka plan ayarlayabilirsiniz. Örneğin, SWF dosyanızın karmaşık vektör resmi içeren bir arka planı varsa, `opaqueBackground` özelliğini belirtilen bir renge (genellikle Sahne Alanı ile aynı renge) ayarlayabilirsiniz. Sayı olarak belirtilen değer (genellikle onaltılık bir renk değeri). Daha sonra arka plan bir bitmap olarak değerlendirilir ve bu da performansın en iyileştirilmesine yardımcı olur.

`cacheAsBitmap` öğesini `true` değerine ayarlayıp `opaqueBackground` özelliğini de belirtilen bir renge ayarladığınızda, `opaqueBackground` özelliği, dahili bitmap'in opak olmasını ve daha hızlı oluşturulmasını sağlar. `cacheAsBitmap` öğesini `true` değerine ayarlamazsanız, `opaqueBackground` özelliği, görüntüleme nesnesinin arka planına opak bir vektör kare şekli ekler. Bu, bitmap'i otomatik şekilde oluşturmaz.

Aşağıdaki örnek, performansı en iyileştirmek için görüntüleme nesnesinin arka planının nasıl ayarlandığını gösterir:

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

Bu durumda, `myShape` adındaki Shape öğesinin arka plan rengi kırmızı (`0xFF0000`) olarak ayarlanır. Shape örneğinin yeşil bir üçgen çizimi içerdiği varsayılırsa, beyaz arka planın bulunduğu bir Sahne Alanı'nda bu, Shape örneğinin sınırlama kutusunda (Shape örneğini tamamen kapsayan dikdörtgen) boş alanın kırmızı olduğu yeşil bir üçgen olarak görünür.

Elbette bu kodun düz kırmızı arka planın yer aldığı bir Sahne Alanı ile kullanılması daha mantıklı olur. Başka bir renkli arka planda, bu renk belirtilir. Örneğin, beyaz arka plana sahip bir SWF'de, `opaqueBackground` özelliği büyük ihtimalle `0xFFFFFF` değerine veya sade beyaza ayarlanır.

Karışım modları uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Karışım modları, üçüncü bir renk oluşturmak için bir görüntünün renklerinin (temel görüntü) başka bir görüntünün renkleriyle (karışım görüntüsü) birleştirilmesini içerir—sonuçta elde edilen görüntü, gerçekten ekranda görüntülenen görüntüdür. Sonuçta elde edilen görüntüdeki aynı konum için bir piksel değeri üretmek üzere, bir görüntüdeki her piksel değeri, diğer görüntünün karşılık gelen piksel değeriyle işlenir.

Her görüntüleme nesnesi, aşağıdaki karışım modlarından birine ayarlanabilen bir `blendMode` özelliğine sahiptir. Bu sabitler, BlendMode sınıfında tanımlanır. Alternatif olarak, sabitlerin gerçek değerleri olan String değerlerini (parantez içinde) kullanabilirsiniz.

- `BlendMode.ADD` (`"add"`): Genellikle, iki görüntü arasında animasyon uygulanmış bir aydınlatıcı erime efekti oluşturmak için kullanılır.
- `BlendMode.ALPHA` (`"alpha"`): Genellikle, arka plana ön planın saydamlığını uygulamak için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.DARKEN` (`"darken"`): Genellikle tür eklemek için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.DIFFERENCE` (`"difference"`): Genellikle, daha canlı renkler oluşturmak için kullanılır.
- `BlendMode.ERASE` (`"erase"`): Genellikle, ön plan alfasını kullanarak arka planın parçasını kesmek (silmek) için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.HARDLIGHT` (`"hardlight"`): Genellikle gölge efekti oluşturmak için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.INVERT` (`"invert"`): Arka planı ters çevirmek için kullanılır.
- `BlendMode.LAYER` (`"layer"`): Belirli bir görüntüleme nesnesine yönelik ön oluşturma için geçici arabellek oluşturulmasını zorlamak için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.LIGHTEN` (`"lighten"`): Genellikle, tür eklemek için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.MULTIPLY` (`"multiply"`): Genellikle, gölgeler ve derinlik efektleri oluşturmak için kullanılır.
- `BlendMode.NORMAL` (`"normal"`): Karışım görüntüsünün piksel değerlerinin temel görüntünün piksel değerlerini geçersiz kıldığını belirtmek için kullanılır.

- `BlendMode.OVERLAY` (`"overlay"`): Genellikle, gölge efekti oluşturmak için kullanılır. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.SCREEN` (`"screen"`): Genellikle, vurgular ve mercek parlaması oluşturmak için kullanılır.
- `BlendMode.SHADER` (`"shader"`): Özel bir karışım efekti oluşturmak üzere Pixel Bender gölgelendiricisinin kullanıldığını belirtmek için kullanılır. Gölgelendiriciler hakkında daha fazla bilgi için bkz. "Pixel Bender gölgelendiricileriyle çalışma" sayfa 302. (GPU görüntü oluşturma altında desteklenmez.)
- `BlendMode.SUBTRACT` (`"subtract"`): Genellikle, iki görüntü arasında animasyon uygulanmış bir karartıcı erime efekti oluşturmak için kullanılır.

DisplayObject renklerini ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir görüntüleme nesnesinin rengini ayarlamak için ColorTransform sınıfının (flash.geom.ColorTransform) yöntemlerini kullanabilirsiniz. Her görüntüleme nesnesi, Transform sınıfının bir örneği olan `transform` özelliğine sahip olup görüntüleme nesnesine uygulanan çeşitli dönüştürmeler (örn. döndürme, ölçek ve konumdaki değişiklikler, vb.) hakkında bilgi içerir. Transform sınıfı, geometrik dönüştürmeleri hakkındaki bilgilere ek olarak, ColorTransform sınıfının bir örneği olan `colorTransform` özelliğini de içerir ve renk ayarlamaları yapılması için görüntüleme nesnesine erişilmesini sağlar. Bir görüntüleme nesnesinin renk dönüştürme bilgilerine erişmek için, bunun gibi bir kod kullanabilirsiniz:

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

Bir ColorTransform örneği oluşturduktan sonra, hangi renk dönüştürmelerinin uygulanmış olduğunu öğrenmek için bu örneğin özellik değerlerini okuyabilir veya görüntüleme nesnesine renk değişiklikleri yapmak için bu değerleri ayarlayabilirsiniz. Herhangi bir değişiklikten sonra görüntüleme nesnesini güncellemek için, ColorTransform özelliğini `transform.colorTransform` özelliğine yeniden atamanız gerekir.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

Kod ile renk değerlerini ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesine belirli bir kırmızı, yeşil, mavi (RGB) renk değeri atamak için ColorTransform sınıfının `color` özelliği kullanılabilir. Aşağıdaki kod, kullanıcı `blueBtn` adında bir düğmeyi tıklattığında `square` adındaki görüntüleme nesnesinin rengini değiştirmek için `color` özelliğini kullanır:

```
// 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);
```

`color` özelliğini kullanarak görüntüleme nesnesinin rengini değiştirdiğinizde, nesnenin önceden birden çok renk içermiş olup olmadığına bakılmaksızın, bunun nesnenin tamamının rengini değiştireceğini unutmayın. Örneğin, üst kısmında siyah metnin bulunduğu yeşil bir daire içeren bir görüntüleme nesnesi varsa, bu nesnenin ilişkilendirilmiş ColorTransform örneğinin `color` özelliğinin kırmızı bir gölgeye ayarlanması, nesnenin tamamını, daireyi ve metni kırmızıya dönüştürür. (Böylece metin artık nesnenin geri kalanından ayırt edilemez.)

Kod ile renk ve parlaklık efektlerini değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Birden çok renk içeren bir görüntüleme nesneniz (örn. dijital bir fotoğraf) olduğunu ve nesneyi tamamen yeniden renklendirmek istemediğinizi; varolan renkleri esas alarak yalnızca bir görüntüleme nesnesinin rengini ayarlamak istediğinizi varsayın. Bu senaryoda, ColorTransform sınıfı bu tür ayarlama yapmak için kullanabileceğiniz bir dizi çarpan ve uzaklık özelliklerini içerir. `redMultiplier`, `greenMultiplier`, `blueMultiplier` ve `alphaMultiplier` adı verilen çarpan özellikleri, görüntüleme nesnesindeki belirli renkleri çoğaltıp azaltarak renkli fotoğraf filtreleri (veya renkli güneş gözlükleri) gibi çalışır. Uzaklık özellikleri (`redOffset`, `greenOffset`, `blueOffset` ve `alphaOffset`), nesneye fazladan miktarda belirli bir renk eklemek veya belirli bir rengin sahip olabileceği minimum değeri belirtmek için kullanılabilir.

Bu çarpan ve uzaklık özellikleri, Özellik denetçisindeki Renk açılır menüsünden Gelişmiş'i seçtiğinizde Flash geliştirme aracında film klibi sembolleri için kullanılabilir olan gelişmiş renk ayarlarının aynısıdır.

Aşağıdaki kod, bir JPEG görüntüsü yükler ve buna fare işaretçisi x ekseni ve y ekseni boyunca hareket ettikçe kırmızı ve yeşil kanallarını ayarlayan bir renk dönüştürmesi uygular. Bu durumda, herhangi bir uzaklık değeri belirtilmediğinden, ekranda görüntülenen her renk kanalının renk değeri, görüntüdeki orijinal renk değerinin bir yüzdesi olur, başka bir deyişle belirli bir pikselde görüntülenen en yüksek kırmızı veya yeşil, o pikseldeki orijinal kırmızı ya da yeşil miktarı olur.

```
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);
```

Nesneleri döndürme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesneleri, `rotation` özelliği kullanılarak döndürülebilir. Bir nesnenin döndürülüp döndürülmediğini öğrenmek için veya bu özelliği ayarlayabileceğiniz nesneyi, nesneye uygulanacak döndürme miktarını temsil eden bir sayıya (derece cinsinden) döndürmek için bu değeri okuyabilirsiniz. Örneğin, bu kod satırı, `square` adındaki nesneyi 45 derece (tam bir dönüşün sekizde biri) döndürür:

```
square.rotation = 45;
```

Alternatif olarak, "Geometriyle çalışma" sayfa 213 bölümünde açıklanan bir dönüştürme matrisini kullanarak görüntüleme nesnesini döndürebilirsiniz.

Nesneleri soldurma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesini kısmen veya tamamen saydamlaştırmak için görüntüleme nesnesinin saydamlığını kontrol edebilir ya da nesnenin yavaşça görünmesini veya kaybolmasını sağlamak için saydamlığı değiştirebilirsiniz. DisplayObject sınıfının `alpha` özelliği, görüntüleme nesnesinin saydamlığını tanımlar. `alpha` özelliği, 0 ile 1 arasında herhangi bir değere ayarlanabilir, burada 0 tamamen saydam ve 1 tamamen opaktır. Örneğin, bu kod satırları, fare ile tıklatıldığında `myBall` adındaki nesneyi kısmen (yüzde 50) saydam hale getirir:

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

ColorTransform sınıfı üzerinden mevcut olan renk ayarlamalarını kullanarak görüntüleme nesnesinin saydamlığını da değiştirebilirsiniz. Daha fazla bilgi için, bkz. “DisplayObject renklerini ayarlama” sayfa 190.

Görüntüleme nesnelerini maskeleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Başka bir görüntüleme nesnesinin içeriklerinin görünebildiği bir delik oluşturmak için, bir görüntüleme nesnesini maske olarak kullanabilirsiniz.

Maskeyi tanımlama

Bir görüntüleme nesnesinin başka bir görüntüleme nesnesi için maske olacağını belirtmek üzere, maske nesnesini, maskelenecek görüntüleme nesnesinin `mask` özelliği olarak ayarlayın:

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

Maskelenen görüntüleme nesnesi, görüntüleme nesnesinin maske olarak hareket eden tüm opak (saydam olmayan) alanları altında gösterilir. Örneğin, şu kod, 100 x 100 piksel kare kırmızı içeren bir Shape örneği ve 25 piksel yarıçapında mavi bir daire içeren Sprite örneği oluşturur. Daire tıklatıldığında, karenin maskesi olarak ayarlanır, böylece karenin gösterilen tek bölümü, dairenin düz bölümü tarafından kaplanan kısımdır. Başka bir deyişle, yalnızca kırmızı daire görünebilir olur.

```
// 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);
```

Maske olarak hareket eden görüntüleme nesnesi sürüklenebilir, dinamik olarak yeniden boyutlandırılabilir, tek bir maske içinde ayrı şekiller kullanabilir ve bu nesneye animasyon uygulanabilir. Maske görüntüleme nesnesinin mutlaka görüntüleme listesine eklenmesi gerekmez. Ancak, Sahne Alanı ölçeklendiğinde maske nesnesinin ölçeklenmesini istiyorsanız veya maske ile kullanıcı etkileşimini (örn. kullanıcı tarafından denetlenen sürükleme ve yeniden boyutlandırma) etkinleştirmek istiyorsanız, maske nesnesinin görüntüleme listesine eklenmesi gerekir. Maske nesnesi, görüntüleme listesine eklendiği sürece, görüntüleme nesnelerinin gerçek z dizininin (önden arkaya sıralama) önemi yoktur. (Maske nesnesi, maske olarak görüntülenmek dışında ekranda görüntülenmez.) Maske nesnesi birden çok kare içeren bir MovieClip örneğiyse, zaman çizelgesinde tüm kareleri oynatır; maske nesnesi maske görevi görmeseydi de bu durum aynı olacaktır. `mask` özelliğini `null` değerine ayarlayarak bir maskeyi kaldırabilirsiniz:

```
// remove the mask from mySprite
mySprite.mask = null;
```

Bir maskeyi, başka bir maskeyi maskelemek için kullanamazsınız. Bir maske görüntüleme nesnesinin `alpha` özelliğini ayarlayamazsınız. Maske olarak kullanılan bir görüntüleme nesnesinde yalnızca dolgular kullanılır; konturlar yok sayılır.

AIR 2

Maskeli görüntüleme nesnesi `cacheAsBitmap` ve `cacheAsBitmapMatrix` özellikleri kullanılarak önbelleğe alınmışsa maske, maskeli görüntüleme nesnesinin alt öğesi olmalıdır. Benzer şekilde, maskelenmiş görüntüleme nesnesi önbelleğe alınmış bir görüntüleme nesnesi kabının alt öğesiyse, hem maske hem de görüntüleme nesnesi o kabın alt öğesi olmalıdır. Maskelenmiş nesne birden fazla önbelleğe alınmış görüntüleme nesnesi kabının alt öğesiyse, maske görüntüleme listesindeki maskelenmiş nesneye en yakın önbelleğe alınmış kabın alt öğesi olmalıdır.

Aygıt fontlarını maskeleme hakkında

Bir aygıt fontunda ayarlanan metni maskelemek için görüntüleme nesnesi kullanabilirsiniz. Bir aygıt fontunda ayarlanan metni maskelemek için görüntüleme nesnesi kullandığınızda, maskeleme şekli olarak maskenin dikdörtgen sınırlama kutusu kullanılır. Başka bir deyişle, aygıt font metni için dikdörtgen olmayan bir görüntüleme nesnesi maskesi oluşturursanız, SWF dosyasında görüntülenen maske, maskenin kendi şekli değil, maskenin dikdörtgen sınırlama kutusunun şeklidir.

Alfa kanalı maskeleme

Alfa kanalı maskeleme, burada gösterildiği gibi hem maske hem de maskelenen görüntüleme nesneleri bitmap önbelleğe almayı kullandığında desteklenir:

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

Örneğin, alfa kanalı maskelemenin bir uygulaması, maskelenen görüntüleme nesnesine uygulanan bir filtreden bağımsız olarak maske nesnesinde bir filtre kullanılmasıdır.

Aşağıdaki örnekte, Sahne Alanı'na harici bir görüntü dosyası yüklenir. Bu görüntü (daha net olmak gerekirse, yüklenen Loader örneği), maskelenen görüntüleme nesnesidir. Görüntünün üzerine bir oval degrade çizilir (kenarlardaki saydama ortadan düz siyah soldurma); bu alfa maskesi olacaktır. Her iki görüntüleme nesnesinde bitmap önbelleğe alma etkindir. Oval, görüntü için bir maske olarak ayarlanır ve daha sonra sürüklenebilir duruma getirilir.

```
// 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);
```

Nesnelere animasyon uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Animasyon, bir şeyi hareket ettirme veya alternatif olarak, bir şeyin zamanla değişmesini sağlama işlemidir. Komut dosyası içeren animasyon, video oyunlarının temel parçası olup genellikle diğer uygulamalara düzgün ve kullanışlı etkileşim ipuçları eklemek için kullanılır.

Komut dosyası içeren animasyonun arkasındaki asıl düşünce, bir değişim gerçekleşmesi ve değişimin zamanla artış birimlerine ayrılmasının gerektiğidir. Yaygın bir döngü deyimi kullanarak bir şeyin ActionScript'te yinelenmesini sağlamak kolaydır. Ancak, görüntü güncellenmeden önce bir döngü tüm yinelemelerini gerçekleştirir. Komut dosyası içeren animasyon oluşturmak için, zamanla aynı eylemi art arda gerçekleştiren ve her çalıştırıldığında ekranı güncelleyen ActionScript yazmanız gerekir.

Örneğin, bir topun ekranda gezinmesini sağlamak gibi basit bir animasyon oluşturmak istediğinizi varsayın. ActionScript, zaman akışını izlemenize ve uygun şekilde ekranı güncellemenize olanak sağlayan basit bir mekanizma içerir, başka bir deyişle, hedefine ulaşıncaya kadar topun her seferinde küçük bir miktar hareket etmesini sağlayan bir kod yazabilirsiniz. Her hareketten sonra ekran güncellenerek Sahne Alanı üzerindeki hareketin izleyen tarafından görülebilmesini sağlar.

Pratik açıdan ele alındığında, SWF dosyasının kare hızı, Flash Player veya AIR'in ekranı güncelleme hızı olduğundan, komut dosyası içeren animasyonun, SWF dosyasının kare hızıyla senkronize edilmesi (başka bir deyişle, her yeni bir kare görüntülendiğinde veya görüntüleneceğinde bir animasyonun değişmesinin sağlanması) akıllıcadır. Her görüntüleme nesnesi, kare başına bir olay olmak üzere SWF dosyasının kare hızına göre gönderilen bir `enterFrame` olayına sahiptir. Komut dosyası içeren animasyonlar oluşturan geliştiricilerin çoğu, zamanla yinelenen eylemler oluşturmanın bir yolu olarak `enterFrame` olayını kullanır. `enterFrame` olayını dinleyip animasyon uygulanmış topu her karede belirli bir miktar hareket ettiren bir kod yazabilirsiniz ve ekran güncellendikçe (her karede), top yeni konumunda yeniden çizilerek hareket oluşturulur.

Not: *Zamanla art arda bir eylem gerçekleştirmenin başka bir yolu da Timer sınıfının kullanılmasıdır. Belirtilen bir zaman miktarı her geçtiğinde, Timer örneği bir olay bildirimini tetikler. Timer sınıfının zamanlayıcı olayını işleyerek animasyon gerçekleştiren kod yazabilir, zaman aralığını daha küçük bir zaman aralığına (saniye kesrine) ayarlayabilirsiniz. Timer sınıfını kullanma hakkında daha fazla bilgi için, bkz. "Zaman aralıklarını denetleme" sayfa 4.*

Aşağıdaki örnekte, Sahne Alanı'nda `circle` adında bir daire Sprite örneği oluşturulur. Kullanıcı daireyi tıklattığında, komut dosyası içeren bir animasyon dizisi başlayarak `circle` öğesinin tamamen saydam oluncaya kadar solmasını (`alpha` özelliği azaltılır) sağlar:

```
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);
```

Kullanıcı daireyi tıklattığında, `fadeCircle()` işlevi, `enterFrame` olayının dinleyicisi olarak abone olur, başka bir deyişle, her karede çağrılmaya başlar. Bu işlev, `alpha` özelliğini değiştirerek `circle` öğesini soldurur, böylece her karede bir defa dairenin `alpha` değeri 0,05 (yüzde 5) azalır ve ekran güncellenir. Sonuç olarak, `alpha` değeri 0 olduğunda (`circle` tamamen saydam olduğunda), `fadeCircle()` işlevi bir olay dinleyicisi olarak kaldırılır ve animasyon sona erer.

Aynı kod, örneğin, solma yerine animasyon uygulanmış hareket oluşturmak için de kullanılabilir. `enterFrame` olay dinleyicisi olan bir işlevde `alpha` için farklı bir özelliğin getirilmesiyle bunun yerine söz konusu özelliğe animasyon uygulanır. Örneğin, bu satır

```
    circle.alpha -= .05;
```

bu koda dönüştürüldüğünde,

```
    circle.x += 5;
```

`x` özelliğine animasyon uygulanarak dairenin Sahne Alanı üzerinde sağa doğru hareket etmesi sağlanır. İstenen `x` koordinatına ulaşıldığında, animasyonu durdurmak için (başka bir deyişle, `enterFrame` dinleyicisinin aboneliğini kaldırmak için), animasyonu sonlandıran koşul değiştirilebilir.

Sahne Alanı yönlendirmesi

AIR 2.0 ve üstü

Taşınabilir cihazlar, kullanıcı cihazı döndürdüğünde görüntüyü düz tutmak için genellikle kullanıcı arabirimini yeniden yönlendirir. Uygulamanızda otomatik yönlendirmeyi etkinleştirirseniz aygıt görüntünün yönünü düzgün bir şekilde korur, ancak sahne alanının en boy oranı değiştiğinde içeriğinizin düzgün görünmesini sağlamak size bağlıdır. Otomatik yönlendirmeyi devre dışı bırakırsanız, aygıttaki görüntü siz yönlendirmeyi el ile değiştirmediğiniz sürece aynı kalır.

AIR uygulamaları belli bir sayıda farklı taşınabilir cihazlarda ve işletim sistemlerinde çalışır. Temel alınan yönlendirme davranışı işletim sistemleri ve hatta aynı işletim sistemindeki farklı cihazlar arasında değişiklik gösterebilir. Tüm cihazlarda ve işletim sistemlerinde çalışan basit bir tasarım stratejisi, uygulama mizanpajını ne zaman yenilemeniz gerektiğini belirlemek üzere, otomatik yönlendirmeyi etkinleştirmenizi ve Stage `resize` olaylarını dinlemenizi sağlar.

Alternatif olarak, uygulamanız sadece dikey en boy oranını veya sadece yatay en boy oranını destekliyorsa, otomatik yönlendirmeyi kapatıp desteklenen en boy oranını AIR uygulama tanımlayıcısından ayarlayabilirsiniz. Bu tasarım stratejisi tutarlı bir davranış sağlar ve seçilen en boy oranı için "en iyi" yönlendirmeyi seçer. Örneğin, yatay en boy oranını belirtirseniz, seçilen yönlendirme yatay mod ve dışa kaydırmalı klavyelere sahip aygıtlar için uygundur.

Geçerli Sahne Alanı yönlendirmesini ve en boy oranını alma

Yönlendirme, aygıtın normal konumuna göre bildirilir. Çoğu aygıtta açık, yukarı doğru bir konum vardır. Bu konum *varsayılan* yönlendirme olarak görülür. Diğer üç olası yönlendirme şu şekildedir: *sola döndürülmüş*, *sağa döndürülmüş* ve *ters*. Yönlendirme değerleri ayarlanırken veya bunları karşılaştırırken StageOrientation sınıfı dize sabitlerini tanımlar.

Stage sınıfı yönlendirme bildiren iki sınıf tanımlar:

- Stage.deviceOrientation: Aygıtın varsayılan konuma göre fiziksel yönlendirmesini bildirir.

  Not: *Uygulamanız ilk başladığında veya aygıt düz olarak bırakılmışsa deviceOrientation her zaman mevcut değildir. Bu durumlarda, aygıt yönlendirmesi bilinmiyor olarak bildirilir.*

- Stage.deviceOrientation: Sahne Alanı'nın varsayılan konuma göre fiziksel yönlendirmesini bildirir. Otomatik yönlendirme etkinleştirildiğinde, sahne alanı yukarı doğru kalmak için aygıtın tersine doğru döner. Bu nedenle, `orientation` özelliği tarafından bildirilen sağ ve sol konumları `deviceOrientation` özelliği tarafından bildirilenlerin tam tersidir. Örneğin, `deviceRotation`*sağa döndürülmüş* olarak bildirirse, `orientation`*sola döndürülmüş* olarak bildirir.

Sahne alanının geçerli genişliğini ve yüksekliğini karşılaştırarak sahne alanının en boy oranı elde edilebilir:

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ?
StageAspectRatio.LANDSCAPE : StageAspectRatio.PORTRAIT;
```

Otomatik yönlendirme

Otomatik yönlendirme açık olduğunda ve bir kullanıcı aygıtı döndürdüğünde işletim sistemi sistem görev çubuğu ve uygulamanız da dahil olmak üzere tüm kullanıcı arabirimini yeniden yönlendirir. Sonuç olarak, sahne alanının en boy oranı dikeyden yataya veya yataydan dikeye değişir. En boy oranı değiştiğinde sahne alanı boyutları da değişir.

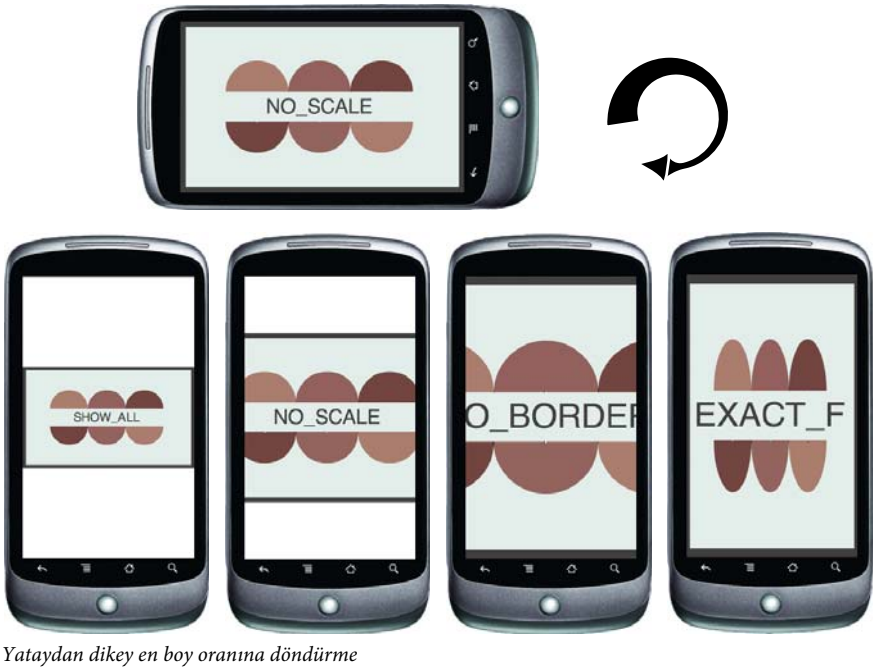
Sahne Alanı'nın `autoOrients` özelliğini `true` veya `false` olarak ayarlayarak otomatik yönlendirmeyi çalışma zamanında etkinleştirin veya devre dışı bırakın. `<autoOrients>` öğesiyle AIR uygulama tanımlayıcısında bu özelliğin başlangıç değerini ayarlayabilirsiniz. (AIR 2.6'dan önce `autoOrients` öğesinin salt okunur bir özellik olduğunu ve yalnızca uygulama açıklayıcısında ayarlanabileceğini unutmayın.)

Yatay veya dikey bir en boy oranı belirtir ve aynı zamanda otomatik yönlendirmeyi de etkinleştirirseniz, AIR otomatik yönlendirmeyi belirtilen en boy oranına sınırlar.

Sahne Alanı boyutu değişiklikleri

Sahne alanı boyutları değiştiğinde, sahne alanı içerikleri Stage nesnesinin `scaleMode` ve `align` özellikleri tarafından belirtildiği şekilde ölçeklenir ve yeniden konumlandırılır. Çoğu durumda Sahne Alanı'nın `scaleMode` ayarları tarafından sağlanan otomatik davranışa güvenmek iyi sonuçlar oluşturmaz. Bunun yerine birden fazla en boy oranını desteklemek için grafiklerinizin ve bileşenlerinizin mizanpajını yeniden ayarlamanız veya yeniden çizmeniz gerekir. (Esnek mizanpaj mantığı sağlamak uygulamanızın farklı ekran boyutları ve en boy oranlarına sahip aygıtlarda daha iyi çalışacağı anlamına gelir.)

Aşağıdaki çizim tipik bir mobil aygıtı döndürürken farklı `scaleMode` ayarlarının etkisini gösterir:



Yataydan dikey en boy oranına döndürme

Çizim farklı ölçüm modlarıyla yatay en boy oranından dikey en boy oranına döndürürken ortaya çıkan farklı ölçekleme davranışlarını gösterir. Dikeyden yataya döndürme işlemi de benzer etkiler oluşturur.

Yönlendirme değişikliği olayları

Stage nesnesi yönlendirme değişikliklerini belirlemeniz ve tepki göstermeniz için iki tür olay gönderir. Otomatik yönlendirme etkinleştirildiğinde sahne alanı `resize` ve `orientationChange` olayları gönderilir.

Ekranı dik tutması için otomatik yönlendirmeye güveniyorsanız kullanılacak en uygun olay *resize* olayıdır. Sahne alanı bir `resize` olayı gönderdiğinde gerektiği şekilde içeriğin mizanpajını yeniden düzenleyin veya içeriği yeniden çizin. `resize` olayı yalnızca sahne alanı ölçek modu `noScale` olarak ayarlandığında gönderilir.

Yönlendirme değişikliklerini belirlemek için `orientationChange` olayı da kullanılabilir. `orientationChange` olayı yalnızca otomatik yönlendirme etkinleştirildiğinde gönderilir.

Not: *Bazı mobil platformlarda sahne alanı resize veya orientationChange olaylarını göndermeden önce iptal edilebilir* `orientationChanging` *olayını gönderir. Olay tüm platformlarda desteklenmediğinden bu olaya güvenmeyin.*

El ile yönlendirme

AIR 2.6 ve üstü

Sahne Alanı'nın `setOrientation()` veya `setAspectRatio()` yöntemlerini kullanarak sahne alanının yönlendirmesini kontrol edebilirsiniz.

Sahne alanının yönlendirmesini ayarlama

Stage nesnesinin `setOrientation()` yöntemini kullanarak çalışma zamanında sahne alanı yönlendirmesini ayarlayabilirsiniz. İstenilen yönlendirmeyi belirtmek için StageOrientation sınıfı tarafından tanımlanan dize sabitlerini kullanın:

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

Olası her yönlendirmeyi tüm aygıtlar veya işletim sistemleri desteklemez. Örneğin, Android 2.2 standardı dikey olan aygıtlarda programlama yoluyla sola döndürülmüş yönlendirmeyi seçmeyi desteklemez ve baş aşağı yönlendirmeyi hiçbir şekilde desteklemez. Sahne alanının `supportedOrientations` özelliği, `setOrientation()` yöntemine iletilebilecek yönlendirme listesi sağlar:

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

Sahne alanının en boy oranını ayarlama

Öncelikli olarak sahne alanının en boy oranıyla ilgileniyorsanız, en boy oranını dikey veya yatay olarak ayarlayabilirsiniz. Sahne Alanı'nın `setAspectRatio()` yöntemini kullanarak en boy oranını AIR uygulama tanımlayıcısında veya çalışma zamanında ayarlayabilirsiniz:

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

Çalışma zamanı belirtilen en boy oranı için olası iki yönlendirmeden birini seçer. Bu geçerli aygıt yönlendirmesiyle eşleşmeyebilir. Örneğin, ters yönlendirmeye tercihen varsayılan yönlendirme seçilir (AIR 3.2 veya öncesi) ve zıt yönlendirmeye tercihen kaydırmalı klavye için uygun olan yönlendirme seçilir.

(AIR 3.3 ve üstü) AIR 3.3'ten itibaren (SWF sürümü 16), `StageAspectRatio.ANY` sabitini de kullanabilirsiniz. `Stage.autoOrients`, `true` olarak ayarlandıysa ve `setAspectRatio(StageAspectRatio.ANY)` öğesini çağırırsanız, uygulamanız tüm yönlendirmeleri (yatay-sol, yatay-sağ, dikey ve dikey-ters) yeniden yönlendirme özelliğine sahip olur. Ayrıca AIR 3.3'teki bir başka yenilik en boy oranının kalıcı olması ve cihazın diğer döndürme işlemlerinin belirtilen yönlendirmeyle sınırlı olmasıdır.

Örnek: Aygıt yönlendirmesiyle eşleştirmek için sahne alanı yönlendirmesini ayarlama

Aşağıdaki örnekte, sahne alanı yönlendirmesini geçerli aygıt yönlendirmesiyle eşleştirmek için güncelleyen bir işlev gösterilmektedir. Sahne alanının `deviceOrientation` özelliği, otomatik yönlendirme kapatılmış olsa bile aygıtın fiziksel yönlendirmesini gösterir.

```
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
    }
}
```

Yönlendirme değişimi senkronize değildir. Değişikliğin tamamlandığını algılamak için sahne alanı tarafından gönderilen `orientationChange` olayını dinleyebilirsiniz. Yönlendirme bir aygıtta desteklenmiyorsa, `setOrientation()` çağrısı hata göndermeden başarısız olur.

Görüntüleme içeriğini dinamik olarak yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Şu harici görüntüleme varlıklarından herhangi birini ActionScript 3.0 uygulamasına yükleyebilirsiniz:

- ActionScript 3.0'da geliştirilmiş bir SWF dosyası—Bu dosya bir Sprite, MovieClip veya Sprite öğesini genişleten herhangi bir sınıf olabilir. iOS'deki AIR uygulamalarında yalnızca ActionScript bayt kodunu içermeyen SWF dosyaları yüklenebilir. Bu, görüntü ve ses gibi gömülü veri içeren SWF dosyalarının yüklenebileceği, ancak yürütülebilir kod içeren SWF dosyalarının yüklenemeyeceği anlamına gelir.
- Görüntü dosyası—Bunlar arasında JPG, PNG ve GIF dosyaları yer alır.
- AVM1 SWF dosyası—Bu, ActionScript 1.0 veya 2.0'da yazılmış bir SWF dosyasıdır. (Mobil AIR uygulamalarında desteklenmez)

Loader sınıfını kullanarak bu varlıkları yüklersiniz.

Görüntüleme nesnelerini yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Loader nesneleri, bir uygulamaya SWF dosyaları ve grafik dosyaları yüklemek için kullanılır. Loader sınıfı, DisplayObjectContainer sınıfının bir alt sınıfıdır. Loader nesnesi, görüntüleme listesinde (yüklediği SWF veya grafik dosyasını temsil eden görüntüleme nesnesi) yalnızca bir alt görüntüleme nesnesi içerebilir. Görüntüleme listesine bir Loader nesnesi eklediğinizde, şu kodda olduğu gibi, yüklenen alt görüntüleme nesnesini yüklemenin ardından görüntüleme listesine ekleyebilirsiniz:

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

SWF dosyası veya görüntü yüklendikten sonra, bu örnekteki `container` DisplayObjectContainer nesnesi gibi, yüklenen görüntüleme nesnesini başka bir görüntüleme nesnesi kabına da taşıyabilirsiniz:

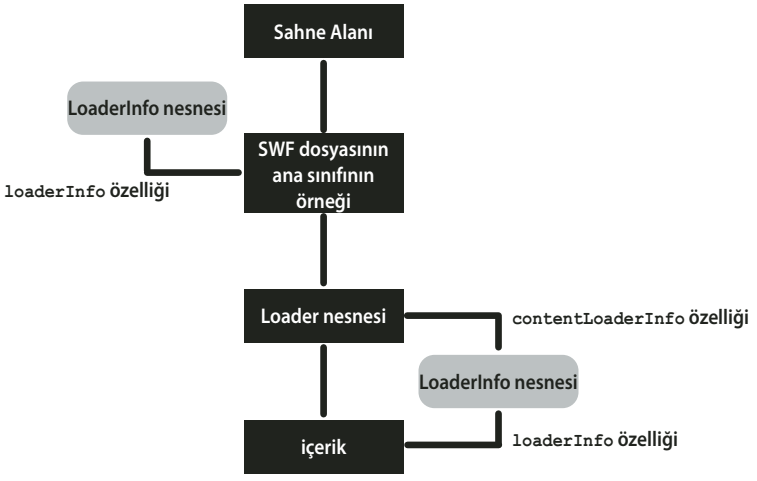
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

Yükleme ilerlemesini izleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dosya yüklenmeye başladıktan sonra, bir LoaderInfo nesnesi oluşturulur. LoaderInfo nesnesi, yükleme ilerlemesi, yükleyici ve yüklenenin URL'leri, medyanın toplam bayt sayısı ve medyanın nominal yüksekliği ve genişliği gibi bilgiler sağlar. LoaderInfo nesnesi, yükleme ilerlemesinin izlenmesine yönelik olaylar da gönderir.

Aşağıdaki diyagram, LoaderInfo nesnesinin farklı kullanımlarını gösterir—örneğin, bir Loader nesnesi için ve Loader nesnesi tarafından yüklenen nesne için SWF dosyasının ana sınıfı:



LoaderInfo nesnesine, hem Loader nesnesinin hem de yüklenen görüntüleme nesnesinin bir özelliği olarak erişilebilir. Yükleme başladığı anda, Loader nesnesinin `contentLoaderInfo` özelliği üzerinden LoaderInfo nesnesine erişilebilir. Görüntüleme nesnesinin yüklemesi sona erdiğinde, yüklenen görüntüleme nesnesinin bir özelliği olarak görüntüleme nesnesinin `loaderInfo` özelliği üzerinden de LoaderInfo nesnesine erişilebilir. Yüklenen görüntüleme nesnesinin `loaderInfo` özelliği, Loader nesnesinin `contentLoaderInfo` özelliği olarak aynı LoaderInfo nesnesini ifade eder. Başka bir deyişle, yüklenen bir nesne ile o nesneyi yükleyen Loader nesnesi arasında (yükleyen ile yüklenen arasında) LoaderInfo nesnesi paylaşılır.

Yüklenen içeriğin özelliklerine erişmek için, aşağıdaki kodda olduğu gibi, LoaderInfo özelliğine bir olay dinleyicisi eklemek istersiniz:

```
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;
}
```

Daha fazla bilgi için “Olayları işleme” sayfa 126 bölümünü inceleyin.

Yükleme içeriği belirtme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Loader sınıfının `load()` veya `loadBytes()` yöntemi üzerinden Flash Player ya da AIR uygulamasına harici bir dosya yüklediğinizde, isteğe bağlı olarak bir `context` parametresi belirtebilirsiniz. Bu parametre bir LoaderContext nesnesidir.

LoaderContext sınıfı, yüklenen içeriğin kullanılabileceği bağlamı tanımlamanızı sağlayan üç özellik içerir:

- `checkPolicyFile`: Yalnızca bir görüntü dosyası (SWF dosyası değil) yüklerken bu özelliği kullanın. Bu özelliği `true` değerine ayarlarsanız, Loader öğesi ilke dosyası için kaynak sunucuyu kontrol eder (bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062). Bu yalnızca Loader nesnesini içeren SWF dosyasının etki alanları dışındaki etki alanlarından ortaya çıkan içerik için gereklidir. Sunucu, Loader etki alanına izin verirse, Loader etki alanında bulunan SWF dosyalarındaki ActionScript, yüklenen görüntüdeki verilere erişebilir; başka bir deyişle, yüklenen görüntüdeki verilere erişmek için `BitmapData.draw()` komutunu kullanabilirsiniz.

  Loader nesnesinin etki alanı dışındaki etki alanlarında bulunan bir SWF dosyasının, belirli bir etki alanına izin vermek için `Security.allowDomain()` öğesini çağırabildiğini unutmayın.

- `securityDomain`: Yalnızca bir SWF dosyası (görüntü değil) yüklerken bu özelliği kullanın. Loader nesnesini içeren dosyanın etki alanı dışındaki bir etki alanında bulunan SWF dosyası için bunu belirtin. Bu seçeneği belirttiğinizde, Flash Player, ilke dosyasının var olup olmadığını kontrol eder ve varsa, çapraz ilke dosyasında izin verilen etki alanlarında bulunan SWF dosyaları, yüklenen SWF içeriğinin çapraz komut dosyasını oluşturabilir. Bu parametre için `flash.system.SecurityDomain.currentDomain` belirtebilirsiniz.

- `applicationDomain`: Yalnızca ActionScript 3.0'da yazılmış bir SWF dosyası (ActionScript 1.0 veya 2.0'da yazılmış bir görüntü ya da SWF dosyası değil) yüklenirken bu özelliği kullanın. Dosyayı yüklerken `applicationDomain` parametresini `flash.system.ApplicationDomain.currentDomain` öğesine ayarlayarak, dosyanın Loader nesnesinin etki alanıyla aynı uygulama etki alanında bulunacağını belirtebilirsiniz Yüklenen SWF dosyasını aynı uygulama etki alanına koyarak, doğrudan sınıflarına erişebilirsiniz. İlişkilendirilebilir sınıf adları üzerinden erişebileceğiniz, gömülü medya içeren bir SWF dosyası yüklüyorsanız bu kullanışlı olabilir. Ayrıntılı bilgi için bkz. "Uygulama etki alanlarıyla çalışma" sayfa 149.

Aşağıda, başka bir etki alanından bitmap yüklenirken ilke dosyasının denetlenmesine yönelik bir örnek yer almaktadır:

```
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);
```

Aşağıda, dosyayı Loader nesnesiyle aynı güvenlik sanal alanına yerleştirmek için başka bir etki alanından SWF yüklenirken ilke dosyasının denetlenmesine yönelik bir örnek yer almaktadır. Ayrıca kod, yüklenen SWF dosyasındaki sınıfları, Loader nesnesinin etki alanıyla aynı uygulama etki alanına ekler:

```
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);
```

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda bulunan LoaderContext sınıfına bakın.

AIR for iOS'ta SWF dosyalarını yükleme

Adobe AIR 3.6 ve üstü, yalnızca iOS

iOS aygıtlarında, çalışma zamanında kod yükleme ve derleme konusunda kısıtlamalar bulunur. Bu kısıtlamalar nedeniyle, harici SWF dosyalarının uygulamanıza yüklenmesi görevinde bazı gerekli farklılıklar bulunur:

- ActionScript kodu içeren tüm SWF dosyaları uygulama paketinde olmalıdır. Kod içeren hiçbir SWF, harici bir kaynaktan (örn. bir ağ üzerinden) yüklenemez. Uygulamayı paketlemenin bir parçası olarak, uygulama paketindeki tüm SWF dosyalarında bulunan ActionScript kodlarının tamamı, iOS aygıtları için yerel kod olarak derlenir.
- Bir SWF dosyasını yükleyip kaldırdıktan sonra yeniden yükleyemezsiniz. Bunu yapmaya çalışırsanız bir hata oluşur.
- Belleğe yüklenip ardından kaldırılması davranışı masaüstü platformlardaki ile aynıdır. Bir SWF dosyasını yükleyip ardından kaldırırsanız, bu SWF dosyasında bulunan tüm görsel varlıklar bellekten kaldırılır. Ancak, yüklü SWF'deki ActionScript sınıfına yönelik tüm sınıf başvuruları bellekte kalır ve bunlara ActionScript kodunda erişilebilir.
- Yüklü tüm SWF dosyaları, ana SWF dosyası ile aynı uygulama etki alanına yüklenmelidir. Bu, varsayılan davranış olmadığından, yüklediğiniz her bir SWF için, ana uygulama etki alanını belirten bir LoaderContext nesnesi oluşturmalı ve söz konusu LoaderContext nesnesini Loader.load() yöntem çağrısına iletmelisiniz. Ana SWF uygulama etki alanından farklı bir uygulama etki alanına SWF yüklemeye çalışırsanız bir hata oluşur. Yüklü SWF'de yalnızca görsel varlıklar bulunuyor ve hiçbir ActionScript kodu bulunmuyor olsa da bu true değerine sahiptir.

  Aşağıdaki örnekte, uygulama paketindeki bir SWF'nin ana SWF uygulama etki alanına yüklenmesi için kullanılacak kod gösterilmektedir:

```
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);
```

Yalnızca varlıklar içeren, hiçbir kod içermeyen bir SWF dosyası uygulama paketinden veya bir ağ üzerinden yüklenebilir. Her iki durumda da SWF dosyasının ana uygulama etki alanına yüklenmesi gerekir.

AIR 3.6'dan önceki AIR sürümleri için tüm kodlar, derleme süreci sırasında ana uygulama SWF'sinin dışındaki SWF'lerden kaldırılır. Yalnızca görsel varlıklar içeren SWF dosyaları uygulama paketine eklenebilir ve çalışma zamanında yüklenebilir. Ancak bu durum kod için söz konusu değildir. ActionScript kodu içeren bir SWF yüklemeye çalışırsanız bir hata oluşur. Bu hata, uygulamada bir “Derlenmeyen ActionScript” hatası iletişim kutusu görünmesine neden olur.

Ayrıca bkz.
Packaging and loading multiple SWFs in AIR apps on iOS (iOS'ta birden fazla SWF paketleme ve AIR'e yükleme)

ProLoader ve ProLoaderInfo sınıflarını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü ve Flash Professional CS5.5 gerektirir

Uzak paylaşılan kütüphane (RSL) önyüklemesine yardımcı olmak için Flash Professional CS5.5 fl.display.ProLoader ve fl.display.ProLoaderInfo sınıflarını sunar. Bu sınıflar, flash.display.Loader ve flash.display.LoaderInfo sınıflarını yansıtır, ancak daha tutarlı bir yükleme deneyimi sağlar.

Özellikle, ProLoader sınıfı RSL önyüklemesine sahip Metin Mizanpajı Çerçevesi (TLF) kullanan SWF dosyalarını yüklemenize yardımcı olur. Çalışma zamanında, TLF gibi diğer SWF dosyalarını veya SWZ dosyalarını önyükleyen SWF dosyaları, yalnızca dahili olan SWF sarıcı dosyası gerektirir. SWF sarıcı dosyası tarafından uygulanan ekstra karmaşıklık katmanı, istenmeyen davranışlara neden olabilir. ProLoader, bu dosyaları normal SWF dosyalarıymış gibi yüklemek için bu karmaşıklığı çözer. ProLoader sınıfı tarafından kullanılan çözüm kullanıcı tarafından görülebilir ve ActionScript'te özel işlem gerektirmez. Ek olarak, ProLoader normal SWF içeriğini düzgün bir şekilde yükler.

Flash Professional CS5.5 ve üstünde, Loader sınıfının tüm kullanımlarını ProLoader sınıfı ile güvenli bir şekilde değiştirebilirsiniz. Ardından, ProLoader sınıfının gerekli ActionScript işlevselliğine erişebilmesini sağlamak için uygulamanızı Flash Player 10.2 veya üstüne dışa aktarın. Ayrıca, ProLoader sınıfını Flash Player'ın ActionScript 3.0'ı destekleyen önceki sürümlerini hedef alırken de kullanabilirsiniz. Ancak, ProLoader özelliklerinden yalnızca Flash Player 10.2 veya üstünde tam olarak faydalanabilirsiniz. Flash Professional veya üstünde TLF kullanırken her zaman ProLoader sınıfını kullanın. ProLoader'a Flash Professional dışındaki ortamlarda ihtiyaç yoktur.

Önemli: *Flash Professional CS5.5 ve üstünde yayınlanan SWF dosyaları için her zaman flash.display.Loader ve flash.display.LoaderInfo sınıfları yerine fl.display.ProLoader ve fl.display.ProLoaderInfo sınıflarını kullanabilirsiniz.*

ProLoader sınıfı tarafından ele alınan sorunlar

ProLoader sınıfı, eski Loader sınıfının işlemek üzere tasarlanmadığı sorunlarla ilgilenir. Bu sorunlar TLF kütüphanelerinin RSL önyüklemesinden kaynaklanır. Özellikle, diğer SWF dosyalarını yüklemek için bir Loader nesnesi kullanan SWF dosyaları için geçerlidir. Ele alınan sorunlar şunları içerir:

- **Yükleyen dosya ve yüklenen dosya arasındaki komut dosyası beklendiği şekilde davranmaz.** ProLoader sınıfı yükleyen SWF dosyasını otomatik olarak yüklenen SWF dosyasının üst öğesi olarak ayarlar. Bu nedenle, yükleyen SWF dosyasındaki iletişim bilgileri doğrudan yüklenen SWF dosyasına gider.
- **SWF uygulaması yükleme sürecini etkin bir şekilde yönetmelidir.** Bunu gerçekleştirmek `added`, `removed`, `addedToStage` ve `removedFromStage` olayları gibi ek olayların uygulanmasını gerektirir. Uygulamanız Flash Player 10.2 veya daha üstünü hedefliyorsa, ProLoader bu ekstra işleme olan ihtiyacı ortadan kaldırır.

Kodu Loader yerine ProLoader'ı kullanacak şekilde güncelleme

ProLoader sınıfı Loader sınıfını yansıttığından kodunuzda iki sınıf arasında kolaylıkla geçiş yapabilirsiniz. Şu örnek varolan kodun yeni sınıfı kullanmak üzere nasıl güncelleneceğini gösterir:

```
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!');
}
```

Bu kod ProLoader sınıfını kullanmak üzere aşağıdaki şekilde güncellenebilir:

```
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!');
}
```

Görüntüleme nesnesi örneği: SpriteArranger

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SpriteArranger örnek uygulaması, *ActionScript 3.0'ı Öğrenme* bölümünde ayrıca açıklanan GeometricShapes örnek uygulamasına göre geliştirilmiştir.

SpriteArranger örnek uygulaması, görüntüleme nesneleriyle ilgili birçok kavramı gösterir:

- Görüntüleme nesnesi sınıflarını genişletme
- Görüntüleme listesine nesneler ekleme
- Görüntüleme nesnelerini katmanlama ve görüntüleme nesnesi kaplarıyla çalışma
- Görüntüleme nesnesi olaylarını yanıtlama
- Görüntüleme nesnelerinin özelliklerini ve yöntemlerini kullanma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. SpriteArranger uygulama dosyalarını Examples/SpriteArranger klasörü içinde bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
SpriteArranger.mxml veya SpriteArranger.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/SpriteArranger/CircleSprite.as	Ekranda daire oluşturan bir Sprite nesnesi türünü tanımlayan bir sınıf.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	GeometricSprite nesnelerini içeren bir görüntüleme nesnesi kabı niteliğindeki tuvali tanımlayan bir sınıf.
com/example/programmingas3/SpriteArranger/SquareSprite.as	Ekranda kare oluşturan bir Sprite nesnesi türünü tanımlayan bir sınıf.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	Ekranda üçgen oluşturan bir Sprite nesnesi türünü tanımlayan bir sınıf.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	Ekrandaki bir şekli tanımlamak için kullanılan Sprite nesnesini genişleten bir sınıf. CircleSprite, SquareSprit ve TriangleSprite öğelerinin her biri bu sınıfı genişletir.

File	Açıklama
com/example/programmingas3/geometricshapes/IGeometricShape.as	Tüm geometrik şekil sınıfları tarafından uygulanacak yöntemlerin tanımlanmasına yönelik temel arabirim.
com/example/programmingas3/geometricshapes/IPolygon.as	Birden çok kenarı olan geometrik şekil sınıfları tarafından uygulanacak yöntemleri tanımlayan bir arabirim.
com/example/programmingas3/geometricshapes/RegularPolygon.as	Şeklin merkezi etrafında simetrik olarak konumlandırılmış eşit uzunlukta kenarlara sahip bir geometrik şekil türü.
com/example/programmingas3/geometricshapes/Circle.as	Bir daireyi tanımlayan geometrik şekil türü.
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	Eşit kenar bir üçgeni tanımlayan bir RegularPolygon alt sınıfı.
com/example/programmingas3/geometricshapes/Square.as	Kareyi tanımlayan bir RegularPolygon alt sınıfı.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	Belirli bir tür ve boyutta şekiller oluşturulması için "fabrika yöntemini" içeren bir sınıf.

SpriteArranger Sınıflarını Tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SpriteArranger uygulaması, kullanıcının ekrandaki "tuvale" çeşitli görüntüleme nesneleri eklemesine olanak sağlar.

DrawingCanvas sınıfı, kullanıcının, üzerine ekran şekilleri ekleyebileceği bir görüntüleme nesnesi kabı türü olan bir çizim alanı ekler. Bu ekran şekilleri, GeometricSprite sınıfının alt sınıflarından birinin örnekleridir.

DrawingCanvas sınıfı

Flex'te, Container nesnesine eklenen tüm alt görüntüleme nesnelerinin, mx.core.UIComponent sınıfının alt öğeleri olan bir sınıfa ait olması gerekir. Bu uygulama, SpriteArranger.mxml dosyasındaki MXML kodunda tanımlandığı gibi, DrawingCanvas sınıfının bir örneğini mx.containers.VBox nesnesinin bir alt öğesi olarak ekler. Bu miras alma işlemi, DrawingCanvas sınıfı bildiriminde şu şekilde tanımlanır:

```
public class DrawingCanvas extends UIComponent
```

UIComponent sınıfı, DisplayObject, DisplayObjectContainer ve Sprite sınıflarından miras alır ve DrawingCanvas sınıfındaki kod, bu sınıfların yöntemlerini ve özelliklerini kullanır.

DrawingCanvas sınıfı Sprite sınıfını genişletir ve bu miras, DrawingCanvas sınıfı bildiriminde şu şekilde tanımlanır:

```
public class DrawingCanvas extends Sprite
```

Sprite sınıfı, DisplayObjectContainer ve DisplayObject sınıflarının bir alt sınıfıdır ve DrawingCanvas sınıfı, bu sınıfların yöntemlerini ve özelliklerini kullanır.

`DrawingCanvas()` yapıcı yöntemi, bir Rectangle nesnesi olan `bounds` öğesini ayarlar ve bu özellik daha sonra tuvalin anahattının çiziminde kullanılır. Ardından, şu şekilde `initCanvas()` yöntemini çağırır:

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

Aşağıdaki örnekte de gösterildiği gibi, `initCanvas()` yöntemi, argüman olarak yapıcı işlevine iletilen DrawingCanvas nesnesinin çeşitli özelliklerini tanımlar:

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

Daha sonra `initCanvas()` yöntemi, DrawingCanvas sınıfının `graphics` özelliğini kullanarak tuval çizen `drawBounds()` yöntemini çağırır. `graphics` özelliği, Shape sınıfından miras alınır.

```
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();
```

DrawingCanvas sınıfının şu ek yöntemleri, uygulamayla etkileşime bağlı olarak çağrılır:

- `addShape()` ve `describeChildren()` yöntemleri, bkz. "Tuvale görüntüleme nesneleri ekleme" sayfa 210
- `moveToBack()`, `moveDown()`, `moveToFront()` ve `moveUp()` yöntemleri, bkz. "Görüntüleme nesnesi katmanlamasını yeniden düzenleme" sayfa 212
- `onMouseUp()` yöntemi, bkz. "Görüntüleme nesnelerini tıklatıp sürükleme" sayfa 211

GeometricSprite sınıfı ve alt sınıfları

Kullanıcının tuvale ekleyebildiği her görüntüleme nesnesi, GeometricSprite sınıfının şu alt sınıflarından birinin örneğidir:

- CircleSprite
- SquareSprite
- TriangleSprite

GeometricSprite sınıfı, flash.display.Sprite sınıfını genişletir:

```
public class GeometricSprite extends Sprite
```

GeometricSprite sınıfı, tüm GeometricSprite nesneleri için ortak olan bir çok özelliği içerir. Bunlar, işleve iletilen parametreler esas alınarak yapıcı işlevinde ayarlanır. Örneğin:

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

GeometricSprite sınıfının `geometricShape` özelliği, şeklin matematiksel özelliklerini tanımlayıp görsel özelliklerini tanımlamayan bir IGeometricShape arabirimini tanımlar. IGeometricShape arabirimini uygulayan arabirimler, *ActionScript 3.0'ı Öğrenme* bölümünde açıklanan GeometricShapes örnek uygulamasında tanımlanmıştır.

GeometricSprite sınıfı, GeometricSprite öğesinin her bir alt sınıfındaki geçersiz kılma tanımlarında daha fazla işlenen `drawShape()` yöntemini tanımlar. Daha fazla bilgi için, ilerleyen bölümlerde bkz. "Tuvale görüntüleme nesneleri ekleme".

GeometricSprite sınıfı ayrıca şu yöntemleri sağlar:

- `onMouseDown()` ve `onMouseUp()` yöntemleri, bkz. "Görüntüleme nesnelerini tıklatıp sürükleme" sayfa 211
- `showSelected()` ve `hideSelected()` yöntemleri, bkz. "Görüntüleme nesnelerini tıklatıp sürükleme" sayfa 211

Tuvale görüntüleme nesneleri ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı Şekil Ekle düğmesini tıklattığında, uygulama, DrawingCanvas sınıfının `addShape()` yöntemini çağırır. Aşağıdaki örnekte gösterildiği gibi, GeometricSprite alt sınıflarından birinin uygun yapıcı işlevini çağırarak yeni bir GeometricSprite öğesini başlatır:

```
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);
}
```

Her yapıcı yöntemi, `drawShape()` yöntemini çağırır, bu da uygun vektör grafiğini çizmek için sınıfın `graphics` özelliğini (Sprite sınıfından miras alınan) kullanır. Örneğin, CircleSprite sınıfının `drawShape()` yöntemi şu kodu içerir:

```
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);
```

`addShape()` işlevinin ikinci satırı ile sonuncu satırı arasındaki satırlar, görüntüleme nesnesinin `alpha` özelliğini (DisplayObject sınıfından miras alınan) ayarlar, böylece tuvale eklenen her görüntüleme nesnesi biraz daha saydam olduğundan kullanıcı bunun ardındaki nesneleri görebilir.

`addChild()` yönteminin son satırı, zaten görüntüleme listesinde bulunan DrawingCanvas sınıfı örneğinin alt öğe listesine yeni görüntüleme nesnesini ekler. Bu da, yeni görüntüleme nesnesinin Sahne Alanı'nda görüntülenmesini sağlar.

Uygulamanın arabirimi iki metin alanı içerir: `selectedSpriteTxt` ve `outputTxt`. Bu metin alanlarının metin özellikleri, tuvale eklenmiş veya kullanıcı tarafından seçilmiş olan GeometricSprite nesneleri hakkındaki bilgilerle güncellenir. GeometricSprite sınıfı, aşağıdaki gibi, `toString()` yöntemini geçersiz kılarak bu bilgi raporlama görevini işler:

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

`shapeType` özelliği, her bir GeometricSprite alt sınıfının yapıcı yönteminde uygun değere ayarlanır. Örneğin, `toString()` yöntemi, DrawingCanvas örneğine henüz eklenmiş olan bir CircleSprite örneği için şu değeri döndürebilir:

```
Circle of size 50 at 0, 0
```

DrawingCanvas sınıfının `describeChildren()` yöntemi, `for` döngüsünün sınırını ayarlamak için `numChildren` özelliğini (DisplayObjectContainer sınıfından miras alınan) kullanarak tuvalin alt öğe listesi üzerinden döngü oluşturur. Bu, aşağıdaki gibi, alt öğelerin her birini listeleyen bir dize oluşturur:

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

Sonuçta elde edilen dize, `outputTxt`metin alanının `text` özelliğini ayarlamak için kullanılır.

Görüntüleme nesnelerini tıklatıp sürükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı bir GeometricSprite örneğini tıklattığında, uygulama, `onMouseDown()` olay işleyicisini çağırır. Aşağıda gösterildiği gibi, bu olay işleyicisi, GeometricSprite sınıfının yapıcı işlevindeki fare basılı olaylarını dinleyecek şekilde ayarlanmıştır:

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

`onMouseDown()` yöntemi daha sonra GeometricSprite nesnesinin `showSelected()` yöntemini çağırır. Bu yöntem nesne için ilk defa çağrılıyorsa, yöntem `selectionIndicator` adında yeni bir Shape nesnesi oluşturur ve aşağıdaki gibi kırmızı bir vurgu dikdörtgenini çizmek için Shape nesnesinin `graphics` özelliğini kullanır:

```
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);
```

`onMouseDown()` yöntemi ilk defa çağrılmıyorsa, aşağıdaki gibi, yöntem, `selectionIndicator` şeklinin `visible` özelliğini (DisplayObject sınıfından miras alınan), ayarlar:

```
this.selectionIndicator.visible = true;
```

`hideSelected()` yöntemi, önceden seçilen nesnenin `visible` özelliğini `false` değerine ayarlayarak bu nesnenin `selectionIndicator` şeklini gizler.

`onMouseDown()` olay işleyicisi yöntemi ayrıca şu kodu içeren `startDrag()` yöntemini (Sprite sınıfından miras alınan) çağırır:

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

Bu, `boundsRect` dikdörtgeni tarafından ayarlanan sınırlar içinde kullanıcının seçili nesneyi tuval etrafında sürüklemesine olanak sağlar.

Kullanıcı fareyi serbest bıraktığında, `mouseUp` olayı gönderilir. DrawingCanvas öğesinin yapıcı yöntemi, şu olay dinleyicisini ayarlar:

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

Bu olay dinleyicisi, GeometricSprite nesneleri için değil, DrawingCanvas nesnesi için ayarlanır. Bunun nedeni, GeometricSprite nesnesi sürüklenirken, fare serbest bırakıldığında bu nesnenin başka bir görüntüleme nesnesinin (başka bir GeometricSprite nesnesi) arkasında kalabilmesidir. Ön plandaki görüntüleme nesnesi, fare basılı olmama olayını alır ancak kullanıcının sürüklemekte olduğu görüntüleme nesnesi bu olayı almaz. DrawingCanvas nesnesine dinleyici eklenmesi, olayın her zaman işlenmesini sağlar.

`onMouseUp()` yöntemi, GeometricSprite nesnesinin `onMouseUp()` yöntemini çağırır, bu da daha sonra GeometricSprite nesnesinin `stopDrag()` yöntemini çağırır.

Görüntüleme nesnesi katmanlamasını yeniden düzenleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanın kullanıcı arabirimi, Geriye Taşı, Aşağı Taşı, Yukarı Taşı ve Öne Taşı etiketli düğmeleri içerir. Kullanıcı bu düğmelerden birini tıklattığında, uygulama DrawingCanvas sınıfının karşılık gelen yöntemini çağırır: `moveToBack()`, `moveDown()`, `moveUp()` veya `moveToFront()`. Örneğin, `moveToBack()` yöntemi şu kodu içerir:

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

Bu yöntem, DrawingCanvas örneğinin (`this`) alt öğe listesinde 0 dizin konumuna görüntüleme nesnesini yerleştirmesi için `setChildIndex()` yöntemini (DisplayObjectContainer sınıfından miras alınan) kullanır.

`moveDown()` yöntemi benzer şekilde çalışır, tek farkı, DrawingCanvas örneğinin alt öğe listesinde görüntüleme nesnesinin dizin konumunu 1'er 1'er azaltmasıdır:

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

`moveUp()` ve `moveToFront()` yöntemleri, `moveToBack()` ve `moveDown()` yöntemlerine benzer şekilde çalışır.

Bölüm 11: Geometriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

flash.geom paketi, nokta, dikdörtgen ve dönüştürme matrisleri gibi geometrik şekilleri tanımlayan sınıfları içerir. Diğer sınıflarda kullanılan nesnelerin özelliklerini tanımlamak için bu sınıfları kullanırsınız.

Daha fazla Yardım konusu

flash.geom paketi

Geometri temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

flash.geom paketi, nokta, dikdörtgen ve dönüştürme matrisleri gibi geometrik şekilleri tanımlayan sınıfları içerir. Bu sınıfların mutlaka işlevleri kendi başına sağlaması gerekmez; bunlar diğer sınıflarda kullanılan nesnelerin özelliklerini tanımlamak için kullanılır.

Tüm geometri sınıfları, ekrandaki konumların iki boyutlu bir düzlem olarak temsil edildiği varsayımına dayanmaktadır. Ekran, bir yatay (x) eksenin ve bir dikey (y) eksenin bulunduğu düz bir grafik olarak değerlendirilir. Ekrandaki tüm konumlar (veya *point*)bir x ve y değer çifti (o konumun *koordinatları*) olarak temsil edilebilir.

Sahne alanı da dahil olmak üzere her görüntü nesnesinin kendi*koordinat alanı* vardır. Koordinat alanı, görüntü nesnelerinin, çizimlerin vs. alt öğelerinin yerlerinin işaretlenmesi için nesnenin kendi grafiğidir. *orijin* 0,0 koordinat konumundadır (x ve y'nin buluşma noktası), ve görüntü nesnesinin üst sol köşesinde yer alır. Bu orijin noktası Sahne Alanı için her zaman true değerinde olsa da, diğer görüntüleme nesneleri için mutlaka true değerinde olmak zorunda değildir. X eksenindeki değerler sağa doğru gidildikçe artarken sola doğra gidildikçe azalır. Orijin noktasının solundaki konumlar için x koordinatı negatiftir. Ancak, geleneksel koordinat sistemlerinin tersine y eksenindeki Flash çalışma zamanı koordinat değerleri ekrandan aşağı doğru büyür ve ekrandan yukarı doğru küçülür. Orijin noktasının üzerindeki değerlerin negatif koordinat değeri vardır). Sahne alanının üst sol köşesi kendi koordinat aralığının orijin noktası olduğu için, Sahne alanındaki nesnelerin çoğunun koordinatı sıfırdan büyük, Sahne alanının genişliğinden küçüktür. Aynı nesnenin y koordinatı sıfırdan büyüktür ve Sahne aralığının yüksekliğinden küçüktür.

Koordinat alanında tek tek noktaları temsil etmek için Point sınıfı örneklerini kullanabilirsiniz. Koordinat alanında dikdörtgen bir alanı temsil etmek için Rectangle örneği oluşturabilirsiniz. İleri düzey kullanıcılar için, bir görüntüleme nesnesine birden çok veya karmaşık dönüştürmeler uygulamak için Matrix örneğini kullanabilirsiniz. Görüntüleme nesnesinin özellikleri kullanılarak, dönüş, konum ve ölçek değişiklikleri gibi basit dönüştürmelerin çoğu doğrudan o nesneye uygulanabilir. Görüntüleme nesnesi özelliklerini kullanarak dönüştürmeler uygulama hakkında daha fazla bilgi için, bkz. “Görüntüleme nesnelerini işleme” sayfa 176.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi, önemli geometrik terimleri içerir:

Kartezyen koordinatlar Koordinatlar genellikle sayı çiftleri olarak yazılır (örn. 5, 12 veya 17, -23). Bu iki sayı sırasıyla x koordinatını ve y koordinatını belirtir.

Koordinat alanı Görüntüleme nesnesinde bulunan ve üzerinde nesnenin alt öğelerinin konumlandırıldığı koordinatlar grafiği.

Kaynak Bir koordinat aralığındaki nokta, x ekseniyle y ekseninin buluştuğu yerdir. Bu noktanın koordinatı 0, 0'dır.

Nokta Koordinat alanında tek bir konum. ActionScript'te kullanılan 2B koordinat sisteminde x ve y eksenindeki konum (noktanın koordinatları), noktayı tanımlar.

Kayıt noktası Görüntüleme nesnesinde, koordinat alanının başlangıç noktası (0, 0 koordinatı).

Ölçekleme Nesnenin, orijinal boyutuna göre boyutu. Fiil olarak kullanıldığında bir nesnenin ölçeklenmesi, nesneyi genişleterek veya daraltarak nesnenin boyutunun değiştirilmesi anlamına gelir.

Çevirme Noktanın koordinatlarını bir koordinat alanından diğerine değiştirme.

Dönüştürme Nesneyi döndürme, ölçeğini değiştirme, şeklini eğme veya deforme etme ya da rengini değiştirme şeklinde bir grafiğin görsel özellikleri üzerinde yapılan düzenleme.

X ekseni ActionScript'te kullanılan 2-b koordinat sistemindeki yatay eksen.

Y ekseni ActionScript'te kullanılan 2-b koordinat sistemindeki dikey eksen.

Point nesnelerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Point nesnesi bir kartezyan koordinatı çifti tanımlar. İki boyutlu koordinat sisteminde konumu temsil eder, burada *x* yatay ekseni temsil ederken *y* de dikey ekseni temsil eder.

Bir Point nesnesini tanımlamak için, nesnenin `x` ve `y` özelliklerini şu şekilde ayarlarsınız:

```
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;
```

İki nokta arasındaki mesafeyi bulma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Koordinat alanında iki nokta arasındaki mesafeyi bulmak için, Point sınıfının `distance()` yöntemini kullanabilirsiniz. Örneğin, aşağıdaki kod, aynı görüntüleme nesnesi kabında bulunan `circle1` ve `circle2` adındaki iki görüntüleme nesnesinin kayıt noktaları arasındaki mesafeyi bulur:

```
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);
```

Koordinat alanlarını çevirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İki görüntüleme nesnesi, farklı görüntüleme nesnesi kabındaysa, bunlar farklı koordinat aralıklarında olabilir. Koordinatları Sahne Alanı'nın koordinat alanıyla aynı (genel) koordinat alanına çevirmek için, DisplayObject sınıfının `localToGlobal()` yöntemini kullanabilirsiniz. Örneğin, aşağıdaki kod, farklı görüntüleme nesnesi kaplarında bulunan `circle1` ve `circle2` adındaki iki görüntüleme nesnesinin kayıt noktaları arasındaki mesafeyi bulur:

```
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);
```

Aynı şekilde, Sahne Alanı üzerindeki belirli bir noktadan `target` adındaki bir görüntüleme nesnesinin kayıt noktası arasındaki mesafeyi bulmak için, DisplayObject sınıfının `localToGlobal()` yöntemini kullanabilirsiniz:

```
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);
```

Görüntüleme nesnesini belirtilen bir açıya ve mesafeye göre taşıma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesini belirli bir açıya göre belirli bir mesafe boyunca taşımak için, Point sınıfının `polar()` yöntemini kullanabilirsiniz. Örneğin aşağıdaki kod `myDisplayObject` nesnesini 60°'den 100 piksel taşır.

```
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;
```

Point sınıfının diğer kullanımları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Point nesnelerini şu yöntem ve özelliklerle kullanabilirsiniz:

Sınıf	Yöntemler veya özellikler	Açıklama
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()` `getObjectsUnderPoint()`	Görüntü nesnesi kabında bir noktanın altındaki nesnelerin listesini döndürmek için kullanılır.
BitmapData	`hitTest()`	Vuruş için kontrol ettiğiniz noktanın yanı sıra, BitmapData nesnesinde pikseli tanımlamak için kullanılır.

Sınıf	Yöntemler veya özellikler	Açıklama

BitmapData	`applyFilter()`
	`copyChannel()`
	`merge()`
	`paletteMap()`
	`pixelDissolve()`
	`threshold()`
		İşlemleri tanımlayan dikdörtgen konumlarını tanımlamak için kullanılır.

Matris	`deltaTransformPoint()` `transformPoint()`	Dönüştürme uygulamak istediğiniz noktaları tanımlamak için kullanılır.

Rectangle	`bottomRight`
	`size`
	`topLeft`
		Bu özellikleri tanımlamak için kullanılır.

Rectangle nesnelerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Rectangle nesnesi bir dikdörtgen alanını tanımlar. Dikdörtgen nesnesi, sol üst köşesinin*x* ve *y* koordinatlarının tanımladığı bir konuma, bir`width` özelliğine ve`height` özelliğine sahiptir. `Dikdörtgen()` yapıcı işlevini aşağıdaki şekilde çağırarak yeni bir Dikdörtgen nesnesi için bu özellikleri tanımlayabilirsiniz:

```
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);
```

Dikdörtgen nesnelerini yeniden boyutlandırma ve yeniden konumlandırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Rectangle nesnelerini yeniden boyutlandırmanın ve yeniden konumlandırmanın birçok yolu vardır.

`x` ve `y` özelliklerini değiştirerek Rectangle nesnesini doğrudan yeniden konumlandırabilirsiniz. Bu değişikliğin, Rectangle nesnesinin genişliği veya yüksekliği üzerinde bir etkisi yoktur.

```
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)
```

Aşağıdaki kodun gösterdiği gibi, Dikdörtgen nesnesinin `left` veya `top` özelliğini değiştirdiğinizde dikdörtgen yeniden konumlandırılır. Dikdörtgenin `x` ve `y` özellikleri sırasıyla `left` ve`top` özellikleriyle eşleşir. Diğer taraftan, Dikdörtgen nesnesinin sol alt köşesinin konumu değişmez; böylece yeniden boyutlandırılmış olur.

```
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)
```

Aynı şekilde, aşağıdaki örnekte gösterildiği gibi dikdörtgen nesnesinin `bottom` veya `right` özelliğini değiştirirseniz, sol üst konumu değişmez. Dikdörtgen aşağıdaki gibi yeniden boyutlandırılır:

```
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)
```

Aşağıdaki gibi, `offset()` yöntemini kullanarak da Rectangle nesnesini yeniden konumlandırabilirsiniz:

```
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)
```

`offsetPt()` yöntemi benzer şekilde çalışır, tek farkı, *x* ve *y* uzaklık değerlerini değil, bir Point nesnesini parametresi olarak almasıdır.

`dx` ve `dy` adındaki iki parametreyi içeren `inflate()` yöntemini kullanarak da Rectangle nesnesini yeniden boyutlandırabilirsiniz. `dx` parametreleri dikdörtgenin merkezden sola ve sağa taşınan kenarlarının piksel sayısını gösterir. `dy` parametresi merkezden taşınan üst ve alt kenarlarının piksek sayısını gösterir.

```
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)
```

`inflatePt()` yöntemi benzer şekilde çalışır, tek farkı, `dx` ve `dy` değerlerini değil, bir Point nesnesini parametresi olarak almasıdır.

Rectangle nesnelerinin birleşimlerini ve kesişimlerini bulma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İki dikdörtgenin sınırlarıyla oluşturulan dikdörtgen bölgesini bulmak için `union()` yöntemini kullanırsınız:

```
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)
```

İki dikdörtgenin örtüşen alanıyla oluşturulan dikdörtgen bölgesini bulmak için `intersection()` yöntemini kullanırsınız:

```
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)
```

İki dikdörtgenin kesişip kesişmediğini öğrenmek için `intersects()` yöntemini kullanırsınız. Bir görüntüleme nesnesinin Sahne Alanı'nın belirli bir bölgesinde olup olmadığını öğrenmek için `intersects()` yöntemini kullanabilirsiniz. Örneğin aşağıdaki kodda `circle` nesnesini içeren görüntüleme nesnesi kabının koordinat alanının Sahne alanının koordinat alanıyla aynı olduğunu varsayın. Bu örnek, bir görüntüleme nesnesinin (`circle`), `target1` ve `target2` Rectangle nesneleri tarafından tanımlanan Sahne Alanı'nın belirtilen bölgeleriyle kesişip kesişmediğini belirlemek için `intersects()` yönteminin nasıl kullanıldığını gösterir:

```
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
```

Aynı şekilde, iki görüntüleme nesnesinin sınırlama dikdörtgenlerinin örtüşüp örtüşmediğini öğrenmek için `intersects()` yöntemini kullanabilirsiniz. Görüntüleme nesnesinin konturlarının bir sınırlama bölgesine ekleyebileceği herhangi bir ek bir alanı dahil etmek için, DisplayObject sınıfının `getRect()` yöntemini kullanın.

Dikdörtgen nesnelerinin diğer kullanımları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Rectangle nesneleri, şu yöntemlerde ve özelliklerde kullanılır:

Sınıf	Yöntemler veya özellikler	Açıklama
BitmapData	`applyFilter()`, `colorTransform()`, `copyChannel()`, `copyPixels()`, `draw()`, `drawWithQuality()`, `encode()`, `fillRect()`, `generateFilterRect()`, `getColorBoundsRect()`, `getPixels()`, `merge()`, `paletteMap()`, `pixelDissolve()`, `setPixels()`, and `threshold()`	BitmapData nesnesinin bir bölgesini tanımlamak için bazı parametrelerin türü olarak kullanılır.
DisplayObject	`getBounds()`, `getRect()`, `scrollRect`, `scale9Grid`	Özelliğin veri türü veya döndürülen veri türü olarak kullanılır.
PrintJob	`addPage()`	`printArea` parametresini tanımlamak için kullanılır.
Sprite	`startDrag()`	`bounds` parametresini tanımlamak için kullanılır.
TextField	`getCharBoundaries()`	Döndürme veri türü olarak kullanılır.
Transform	`pixelBounds`	Veri türü olarak kullanılır.

Matrix nesnelerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Matrix sınıfı, bir koordinat boşluğundan diğerine nasıl eşleme yapılacağını belirleyen bir dönüşüm matrisini temsil eder. Bir Matrix nesnesinin özelliklerini ayarlayarak bir görüntüleme nesnesi üzerinde çeşitli grafik dönüştürmeleri gerçekleştirebilirsiniz. Bunu yapmak için söz konusu Matrix nesnesini Transform nesnesinin `matrix` özelliğine uygulamanız ve ardından bu Transform nesnesini görüntüleme nesnesinin `transform` özelliği olarak uygulamanız gerekir. Bu dönüştürme işlevleri çevirmeyi (*x* ve *y* yeniden konumlandırması), döndürmeyi, ölçeklemeyi ve eğmeyi içerir.

Bir Matrix nesnesinin özelliklerini (`a`, `b`, `c`, `d`, `tx`, `ty`) ayarlayarak bir matrisi doğrudan tanımlayabilseniz de, `createBox()` yönteminin kullanılması daha kolaydır. Bu yöntem, sonuç matrisinin ölçeklemesini, dönüşünü ve çevirme efektlerini doğrudan tanımlamanıza olanak sağlayan parametreleri içerir. Örneğin, aşağıdaki kod bir nesneyi 45 derece sağa doğru 10 piksel ve 20 piksel aşağı taşıyarak (çeviren) yatay olarak 2.0; dikey olarak 3.0 ölçeklendiren bir Matris nesnesi oluşturur.

```
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);
```

`scale()`, `rotate()` ve `translate()` yöntemlerini kullanarak da Matrix nesnesinin ölçeklemesini, dönüşünü ve çevirme efektlerini ayarlayabilirsiniz. Bu yöntemlerin, varolan Matrix nesnesinin değerleriyle birleştiğini unutmayın. Örneğin, `scale()` ve `rotate()` yöntemleri iki defa çağrıldığından, aşağıdaki kod, bir nesneyi 4 faktörüyle ölçekleyip 60 derece döndüren bir Matrix nesnesini belirler:

```
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;
```

Bir Matrix nesnesine eğme dönüştürmesi uygulamak için, nesnenin `b` veya `c` özelliğini ayarlayın. `b` özelliği ayarlandığında, matris dikey olarak eğilir ve `c` özelliği ayarlandığında da matris yatay olarak eğilir. Aşağıdaki kod, `myMatrix` Matrix nesnesini 2 faktörüyle dikey olarak eğilir:

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

Görüntüleme nesnesinin `transform` özelliğine bir matrix dönüştürmesi uygulayabilirsiniz. Örneğin, aşağıdaki kod, `myDisplayObject` adındaki görüntüleme nesnesine bir matris dönüştürmesi uygular:

```
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;
```

Birinci satır, `myDisplayObject` görüntüleme nesnesi tarafından kullanılan varolan dönüştürme matrisine bir Matrix nesnesi ayarlar (`myDisplayObject` görüntüleme nesnesinin `transformation` özelliğinin `matrix` özelliği). Bu şekilde, çağırdığınız Matrix sınıfı yöntemlerinin, görüntüleme nesnesinin varolan konumu, ölçeği ve dönüşü üzerinde birikimli bir etkisi olur.

Not: *flash.geometry paketine ColorTransform sınıfı da dahil edilmiştir. Bir Transform nesnesinin `colorTransform` özelliğini ayarlamak için bu sınıf kullanılır. Herhangi bir geometrik dönüştürme türüne uygulanmadığı için, bu sınıf burada ayrıntılı olarak ele alınmamıştır. Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda bulunan ColorTransform sınıfına bakın.*

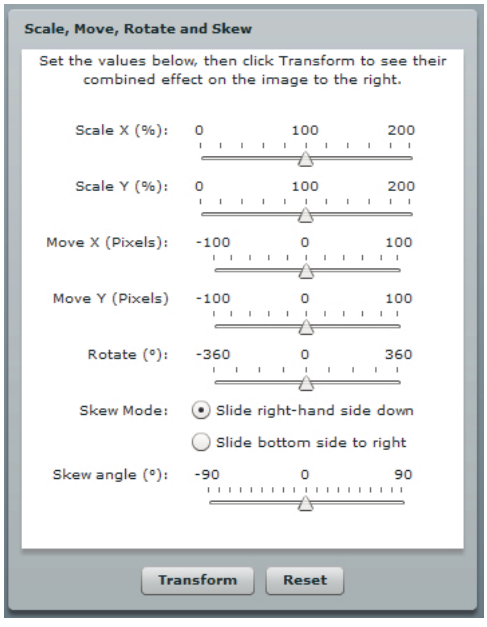
Geometri örneği: Görüntüleme nesnesine matris dönüştürmesi uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

DisplayObjectTransformer örnek uygulaması, görüntüleme nesnesini dönüştürmek için matrix sınıfının özelliklerinin kullanılmasının birçok özelliğini gösterir, bu özellikler arasında şunlar yer alır:

- Görüntüleme nesnesini döndürme
- Görüntüleme nesnesini ölçekleme
- Görüntüleme nesnesini çevirme (yeniden konumlandırma)
- Görüntüleme nesnesini eğme

Bu uygulama, aşağıdaki gibi, matris dönüştürmesinin parametrelerinin ayarlanması için bir arabirim sağlar:



Kullanıcı Dönüştürme düğmesini tıklattığında, uygulama ilgili dönüştürmeyi uygular.

Orijinal görüntüleme nesnesi ve -45° döndürülmüş ve %50 ölçeklenmiş görüntüleme nesnesi

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. DisplayObjectTransformer uygulama dosyalarını Samples/DisplayObjectTransformer klasörü içinde bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
DisplayObjectTransformer.mxml veya DisplayObjectTransformer.fla	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası
com/example/programmingas3/geometry/MatrixTransformer.as	Matris dönüştürmelerinin uygulanmasına yönelik yöntemleri içeren bir sınıf.
img/	Uygulama tarafından kullanılan örnek görüntü dosyalarını içeren bir dizin.

MatrixTransformer sınıfını tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

MatrixTransformer sınıfı, Matrix nesnelerinin geometrik dönüştürmelerini uygulayan statik yöntemleri içerir.

transform() yöntemi

`transform()` yöntemi, aşağıdakilerden her biri için parametreler içerir:

- `sourceMatrix`—Yöntemin dönüştürdüğü girdi matrisi
- `xScale` ve `yScale`—*x* ve *y* ölçek faktörü
- `dx` ve `dy`—Piksel cinsinden *x* ve *y* çevirme miktarları
- `rotation`—Derece cinsinden dönüş miktarı
- `skew`—Yüzde olarak eğme faktörü
- `skewType`—`"right"` veya `"left"` olmak üzere eğme yönü

Döndürme değeri, sonuçta elde edilen matristir.

`transform()` yöntemi, sınıfın şu statik yöntemlerini çağırır:

- `skew()`
- `scale()`
- `translate()`
- `rotate()`

Her biri, dönüştürme uygulanmış şekilde kaynak matrisi döndürür.

skew() yöntemi

`skew()` yöntemi, matrisin `b` ve `c` özelliklerini ayarlayarak matrisi eğer. İsteğe bağlı bir parametre olan `unit`, eğme açısını tanımlamak için birimleri belirler ve gerekirse bu yöntem `angle` değerini radyanlara dönüştürür:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

`skewMatrix` Matrix nesnesi oluşturulur ve eğme dönüştürmesini uygulamak için ayarlanır. Başlangıçta bu aşağıdaki gibi kimlik matrisidir:

```
var skewMatrix:Matrix = new Matrix();
```

`skewSide` parametresi, eğimin uygulandığı kenarı belirler. Bu `"right"` olarak ayarlanırsa, aşağıdaki kod, matrisin `b` özelliğini ayarlar:

```
skewMatrix.b = Math.tan(angle);
```

Aksi takdirde, Matrix öğesinin `c` özelliği ayarlanarak aşağıdaki gibi alt kenar eğilir:

```
skewMatrix.c = Math.tan(angle);
```

Daha sonra aşağıdaki örnekte de gösterildiği gibi iki matris bitiştirilerek, sonuçta elde edilen eğim, varolan matrise uygulanır:

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

scale() yöntemi

Aşağıdaki örnekte gösterildiği gibi, `scale()` yöntemi öncelikle yüzde olarak sağlanmış bir ölçek faktörü varsa bu ölçek faktörünü ayarlar ve sonra da matris nesnesinin `scale()` yöntemini kullanır:

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

translate() yöntemi

`translate()` yöntemi yalnızca aşağıdaki gibi, matris nesnesinin `translate()` yöntemini çağırarak `dx` ve `dy` çevirme faktörlerini uygular:

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

rotate() yöntemi

`rotate()` yöntemi, girdi dönüşü faktörünü radyanlara dönüştürür (derece veya degrade olarak sağlanmışsa), sonra da matris nesnesinin `rotate()` yöntemini çağırır:

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

Uygulamadan MatrixTransformer.transform() yöntemini çağırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulama, kullanıcıdan dönüştürme parametrelerinin alınması için bir kullanıcı arabirimi içerir. Daha sonra bu değerleri, aşağıda görüldüğü gibi, görüntüleme nesnesinin`transform` özelliğinin`matrix` özelliğiyle birlikte `Matrix.transform()` yöntemine iletir:

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

Daha sonra uygulama, döndürme değerini, görüntü nesnesinin `transform` özelliğinin `matrix` özelliğine uygulayarak dönüştürmeyi tetikler:

```
img.content.transform.matrix = tempMatrix;
```

Bölüm 12: Çizim API'sini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İçe aktarılan görüntü ve resimler önemli olsa da, ActionScript'te çizgiler ve şekiller çizmenize olanak sağlayan ve çizim API'si olarak adlandırılan işlevsellik, size üzerinde istediğiniz görüntüleri oluşturabileceğiniz bilgisayar modeli bir boş tuvalle uygulamayı başlatma özgürlüğünü verir. Kendi grafiklerinizi oluşturma yeteneğiniz, uygulamalarınıza yönelik geniş olanaklar sağlar. Burada ele alınan tekniklerle bir çizim programı oluşturabilir, animasyon uygulanmış etkileşimli resimler oluşturabilir veya programlama yoluyla kendi kullanıcı arabirimi öğelerinizi oluşturabilir ve daha birçok şey yapabilirsiniz.

Daha fazla Yardım konusu

flash.display.Graphics

API çizmeyle ilgili temel bilgiler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çizim API'si, vektör grafikleri (çizgiler, eğriler, şekiller, dolgular ve degradeler) oluşturmanıza ve ActionScript kullanarak bunları ekranda görüntülemenize olanak sağlayan ActionScript'teki yerleşik işlevselliğin adıdır. flash.display.Graphics sınıfı bu işlevselliği sağlar. ActionScript ile bu sınıfların her birinde tanımlanan `graphics` özelliğini kullanarak herhangi bir Shape, Sprite veya MovieClip örneğinde çizim yapabilirsiniz. (Bu sınıfların her birinin `graphics` özelliği, aslında Graphics sınıfının bir örneğidir.)

Yalnızca kodla çizim yapma işlemine başlıyorsanız, Graphics sınıfı, daire, elips, dikdörtgen ve yuvarlak köşeli dikdörtgen gibi yaygın şekillerin çizilmesini kolaylaştıran çok sayıda yöntem içerir. Bunları boş çizgiler veya doldurulmuş şekiller olarak çizebilirsiniz. Daha gelişmiş işlevselliğe ihtiyacınız olduğunda, Graphics sınıfı ayrıca ihtiyaç duyduğunuz şekilleri oluşturmak için Math sınıfındaki trigonometri işlevleriyle birlikte kullanabileceğiniz, çizgi ve quadratic Bézier eğrileri çizilmesine yönelik yöntemler de içerir.

Flash çalışma zamanları (Flash Player 10 ve Adobe AIR 1.5 ve daha sonraki sürümleri gibi), çizim için ek bir API ekler ve bu, şekillerin tamamını programlama yoluyla tek bir komutla çizmenize olanak sağlar. Graphics sınıfını ve "Çizim API'sini kullanma temelleri" bölümünde açıklanan görevleri öğrendikten sonra, bu çizim API'si özellikleri hakkında daha fazla bilgi almak için "Çizim API'sinin ileri düzey kullanımı" sayfa 238 başlığıyla devam edin.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi çizim API'sini kullanırken karşılaşacağınız önemli terimleri içerir:

Tutturma noktası Quadratic Bézier eğrisinin iki uç noktasından biri.

Denetim noktası Quadratic Bézier eğrisinin yönünü ve miktarını tanımlayan nokta. Eğri çizgi asla denetim noktasına ulaşmaz; ancak çizgi eğrileri denetim noktasına doğru çiziliyor gibi olur.

Koordinat alanı Görüntüleme nesnesinde bulunan ve üzerinde nesnenin alt öğelerinin konumlandırıldığı koordinatlar grafiği.

Dolgu Bir şeklin renkle doldurulmuş bir çizgi içeren düz renkli iç kısmı veya şeklin anahat içermeyen tüm bölümü.

Degrade Bir renkten başka bir renge veya birkaç renge doğru aşamalı geçiş içeren bir renk (düz rengin aksine).

Nokta Koordinat alanında tek bir konum. ActionScript'te kullanılan 2b koordinat sisteminde nokta, x ekseni ve y ekseni (noktanın koordinatları) üzerindeki konumuyla tanımlanır.

Quadratic Bézier eğrisi Belirli bir matematik formülüyle tanımlanan bir eğri türü. Bu eğri türünde, eğrinin şekli, tutturma noktalarının (eğrinin uç noktalarının) konumları ve eğrinin miktarını ve yönünü tanımlayan bir denetim noktası esas alınarak hesaplanır.

Ölçekleme Nesnenin, orijinal boyutuna göre boyutu. Fiil olarak kullanıldığında bir nesnenin ölçeklenmesi, nesneyi genişleterek veya daraltarak nesnenin boyutunun değiştirilmesi anlamına gelir.

Kontur Şeklin, renkle doldurulmuş anahat kısmı veya doldurulmamış bir şeklin çizgileri.

Çevirme Noktanın koordinatlarını bir koordinat alanından diğerine değiştirme.

X ekseni ActionScript'te kullanılan 2-b koordinat sistemindeki yatay eksen.

Y ekseni ActionScript'te kullanılan 2-b koordinat sistemindeki dikey eksen.

Grafikler sınıfı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Her Shape, Sprite ve MovieClip nesnesi, Graphics sınıfının bir örneği olan `graphics` özelliğine sahiptir. Graphics sınıfı, çizgi, dolgu ve şekiller çizilmesi için özellikler ve yöntemler içerir. Yalnızca çizim içeriği için tuval olarak kullanmak üzere bir görüntüleme nesnesi istiyorsanız, bir Shape örneğini kullanabilirsiniz. Shape örneği, Sprite ve MovieClip sınıflarındaki ek işlevsellik yükünü barındırmadığından, çizim için diğer görüntüleme nesnelerinden daha iyi performans gösterir. Üzerinde grafiksel içerik çizebileceğiniz bir görüntüleme nesnesi istiyorsanız ve bu nesnenin başka görüntüleme nesnelerini içermesini istiyorsanız, bir Sprite örneği kullanabilirsiniz. Çeşitli görevler için kullanılacak görüntüleme nesnesinin belirlenmesiyle ilgili daha fazla bilgi için, bkz. “DisplayObject alt sınıfı seçme” sayfa 175.

Çizgi ve eğriler çizme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir Graphics örneğiyle yapacağınız tüm çizimler, çizgi ve eğrilerle yapılan temel çizimi esas alır. Bunun neticesinde tüm ActionScript çiziminin, aynı adım dizisi kullanılarak gerçekleştirilmesi gerekir:

- Çizgi ve dolgu stillerini tanımlama
- Başlangıç çizim konumunu ayarlama
- Çizgi, eğri ve şekiller çizme (isteğe bağlı olarak çizim noktasını taşıma)
- Gerekirse, dolgu oluşturulmasını tamamlama

Çizgi ve dolgu stillerini tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir Shape, Sprite veya MovieClip örneğinin `graphics` özelliğiyle çizim yapmak için, öncelikle çizim sırasında kullanılacak stili (çizgi boyutu ve rengi, dolgu rengi) tanımlamanız gerekir. Adobe® Flash® Professional veya başka bir çizim uygulamasındaki çizim araçlarını kullandığınızda olduğu gibi, çizim yapmak için ActionScript'i kullandığınızda da konturlu veya kontursuz ya da dolgu rengi ile veya dolgu rengi olmadan çizim yapabilirsiniz. `lineStyle()` veya `lineGradientStyle()` yöntemini kullanarak konturun görünümünü belirtirsiniz. Düz çizgi oluşturmak için, `lineStyle()` yöntemini kullanın. Bu yöntemi çağırırken en yaygın değer olarak şu üç parametreyi belirtirsiniz: çizgi kalınlığı, rengi ve alfası. Örneğin, bu kod satırı, `myShape` adındaki Shape öğesine, 2 piksel kalınlığında, kırmızı (0x990000) ve %75 opak çizgiler çizmesini bildirir:

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

Alfa parametresinin varsayılan değeri 1,0 (%100) şeklindedir, bu nedenle tamamen opak bir çizgi istiyorsanız bu parametreyi devre dışı bırakabilirsiniz. `lineStyle()` yöntemi, piksel ipucu ve ölçek modu için iki ek parametre daha kabul eder. Bu parametrelerin kullanımı hakkında daha fazla bilgi almak için, `http://help.adobe.com/tr_TR/FlashPlatform/reference/actionscript/3/flash/display/Graphics.html#lineStyle()`'ndaki Graphics.lineStyle() yönteminin açıklamasına bakın.

Degrade çizgi oluşturmak için, `lineGradientStyle()` yöntemini kullanın. Bu yöntem, "Degrade çizgiler ve dolgular oluşturma" sayfa 230 bölümünde açıklanır.

Dolgu uygulanmış bir şekil oluşturmak istiyorsanız, çizime başlamadan önce `beginFill()`, `beginGradientFill()`, `beginBitmapFill()` veya `beginShaderFill()` yöntemlerini çağırırsınız. Bunlar arasında en temel olan `beginFill()` yöntemi iki parametre kabul eder: dolgu rengi ve (isteğe bağlı olarak) dolgu rengi için bir alfa değeri. Örneğin, düz yeşil dolgu içeren bir şekil çizmek istiyorsanız, şu kodu kullanırsınız (`myShape` adında bir nesne üzerinde çizim yaptığınız varsayılarak):

```
myShape.graphics.beginFill(0x00FF00);
```

Herhangi bir dolgu yöntemi çağrıldığında, yeni bir dolguya başlanmadan önce bir önceki dolgu örtük olarak sonlandırılır. Kontur stili belirten bir yöntemin çağrılması, bir önceki konturu değiştirir ancak önceden belirtilen dolguyu değiştirmez veya tam tersi de olabilir.

Çizgi stilini ve dolgu özelliklerini belirttikten sonraki adım, çiziminizin başlangıç noktasını belirtmektir. Graphics örneği, kağıttaki kalem ucu gibi bir çizim noktasına sahiptir. Çizim noktasının konumlandırıldığı her yer, bir sonraki çizim eyleminin başlayacağı yeri belirtir. Başlangıçta Graphics nesnesinin çizim noktası, üzerinde çizildiği nesnenin koordinat alanında 0, 0 noktasında bulunur. Farklı bir noktada çizimi başlatmak için, çizim yöntemlerinden birini çağırmadan önce ilk olarak `moveTo()` yöntemini çağırabilirsiniz. Bu, kalemin ucunu kağıttan kaldırıp yeni bir konuma taşımaya benzer.

Çizim noktası yerinde dururken, `lineTo()` (düz çizgi çizmek için) ve `curveTo()` (eğri çizgiler çizmek için) çizim yöntemlerine yapılan çağrılar dizisini kullanarak çizim yaparsınız.

Çizim yaparken, çizim yapmadan çizim noktasını yeni bir konuma taşımak için istediğiniz zaman `moveTo()` yöntemini çağırabilirsiniz.

Çizerken, belirli bir dolgu renginiz varsa, dolguyu `endFill()` yöntemini çağırarak kapatabilirsiniz. Kapalı bir şekil çizmediyseniz (başka bir deyişle, `endFill()` yöntemini çağırdığınız anda, çizim noktası şeklin başlangıç noktasında değilse), `endFill()` yöntemini çağırdığınızda, Flash çalışma zamanı geçerli çizim noktasından en son `moveTo()` çağrısında belirtilen konuma doğru düz bir çizgi çizerek otomatik olarak şekli kapatır. Bir dolgu başlattıysanız ve `endFill()` yöntemini çağırmadıysanız, `beginFill()` yöntemi (veya diğer dolgu yöntemlerinden biri) çağrıldığında, geçerli dolgu kapanır ve yeni bir dolgu başlatılır.

Düz çizgiler çizme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`lineTo()` yöntemini çağırdığınızda, Graphics nesnesi, geçerli çizim noktasından yöntem çağrısında iki parametre olarak belirttiğiniz koordinatlara, belirttiğiniz çizgi stiliyle düz bir çizgi çizer. Örneğin, bu kod satırı çizim noktasını 100, 100 noktasına yerleştirir ve sonra 200, 200 noktasına doğru bir çizgi çizer:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

Aşağıdaki örnek, 100 piksel yüksekliğinde kırmızı ve yeşil üçgen çizer:

```
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);
```

Eğriler çizme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`curveTo()` yöntemi, quadratic Bézier eğrisi çizer. Bu, iki noktayı (tutturma noktaları olarak adlandırılır) birleştirip üçüncü bir noktaya (denetim noktası olarak adlandırılır) doğru eğerek bir yay çizer. Graphics nesnesi, birinci tutturma noktası olarak geçerli çizim konumunu kullanır. `curveTo()` yöntemini çağırdığınızda, dört parametre iletirsiniz: denetim noktasının x ve y koordinatları, ardından da ikinci tutturma noktasının x ve y koordinatları. Örneğin, aşağıdaki kod, 100, 100 noktasında başlayıp 200, 200 noktasında biten bir eğri çizer. Denetim noktası, 175, 125 noktasında bulunduğundan, bu biraz sağa ve aşağı doğru hareket eden bir eğri oluşturur:

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

Aşağıdaki örnek, 100 piksel genişlik ve yükseklikte kırmızı ve yeşil dairesel nesneler çizer. quadratic Bézier eşitliğinin doğası gereği bunların mükemmel daireler olmadığını unutmayın:

```
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);
```

Yerleşik yöntemleri kullanarak şekiller çizme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Daire, elips, dikdörtgen ve yuvarlak köşeli dikdörtgen gibi yaygın şekilleri çizerken kolaylık sağlamak için, ActionScript 3.0'da bu yaygın şekilleri sizin için çizen yöntemler vardır. Bunlar, Graphics sınıfının `drawCircle()`, `drawEllipse()`, `drawRect()` ve `drawRoundRect()` yöntemleridir. Bu yöntemler, `lineTo()` ve `curveTo()` yöntemleri yerine kullanılabilir. Ancak, bu yöntemleri çağırmadan önce de çizgi ve dolgu stilleri belirtmeniz gerektiğini unutmayın.

Aşağıdaki örnek, 100 piksel genişlik ve yükseklikte kırmızı, yeşil ve mavi kare çizilmesi örneğini yeniden oluşturur. Bu kod, `drawRect()` yöntemini kullanır ve ek olarak dolgu renginin %50 (0,5) alfa değerine sahip olduğunu belirtir:

```
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);
```

Sprite veya MovieClip nesnesinde, `graphics` özelliğiyle oluşturulan çizim içeriği her zaman nesnenin içerdiği tüm alt görüntüleme nesnelerinin arkasında görünür. Ayrıca, `graphics` özelliğinin içeriği ayrı bir görüntüleme nesnesi olmadığından, Sprite veya MovieClip nesnesinin alt öğeleri listesinde görüntülenmez. Örneğin, aşağıdaki Sprite nesnesi `graphics` özelliğiyle çizilmiş bir daire içerir ve bunun alt görüntüleme nesneleri listesinde bir TextField nesnesi bulunur:

```
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);
```

TextField öğesinin, grafik nesnesiyle çizilmiş dairenin en üst kısmında görüntülendiğini unutmayın.

Degrade çizgiler ve dolgular oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Grafik nesnesi, düz renkler yerine degradeler içeren kontur ve dolgular da çizebilir. Degrade kontur, `lineGradientStyle()` yöntemiyle ve degrade dolgu da `beginGradientFill()` yöntemiyle oluşturulur.

Her iki yöntem de aynı parametreleri kabul eder. Bunların ilk dördü zorunludur: tür, renkler, alfalar ve oranlar. Kalan dördü ise isteğe bağlı olup ileri düzey özelleştirmelerde kullanışlıdır.

- Birinci parametre, oluşturduğunuz degrade türünü belirtir. Kabul edilebilir değerler `GradientType.LINEAR` veya `GradientType.RADIAL` değeridir.
- İkinci parametre, kullanılacak renk değerlerinin dizisini belirtir. Doğrusal degradede, renkler soldan sağa ayarlanır. Radyal degradede, renkler içten dışa ayarlanır. Renklerin dizideki sırası, renklerin degradede çizileceği sırayı temsil eder.
- Üçüncü parametre, önceki parametrede karşılık gelen renklerin alfa saydamlığı değerlerini belirtir.
- Dördüncü parametre, oranları veya her rengin degradede ne kadar vurguya sahip olduğunu belirtir. Kabul edilebilir değerler 0-255 arasındadır. Bu değerler herhangi bir genişlik veya yüksekliği değil, degrade içindeki konumu temsil eder; 0, degradenin başını temsil ederken, 255 de degradenin sonunu temsil eder. Oranlar dizisi sırayla artmalı ve ikinci ve üçüncü parametrelerde belirtilen renk ve alfa dizileriyle aynı sayıda girişe sahip olmalıdır.

Beşinci parametre olan dönüştürme matrisi isteğe bağlı olsa da, degradenin görünümünü denetlemeye yönelik kolay ve güçlü bir yöntem sağladığından yaygın olarak kullanılır. Bu parametre bir Matrix örneğini kabul eder. Degrade için bir Matrix nesnesi oluşturmanın en kolay yolu, Matrix sınıfının `createGradientBox()` yönteminin kullanılmasıdır.

Degradeyle kullanılacak bir Matrix nesnesini tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Şekillerde kullanılacak degradeleri tanımlamak için flash.display.Graphics sınıfının `beginGradientFill()` ve `lineGradientStyle()` yöntemlerini kullanırsınız. Bir degrade tanımladığınızda, bu yöntemlerin parametrelerinden biri olarak bir matris sağlarsınız.

Matrisi tanımlamanın en kolay yolu, degradeyi tanımlamak için kullanılacak bir matris oluşturan Matrix sınıfının `createGradientBox()` yönteminin kullanılmasıdır. `createGradientBox()` yöntemine iletilen parametreleri kullanarak degradenin ölçeğini, dönüşünü ve konumunu tanımlarsınız. `createGradientBox()` yöntemi şu parametreleri kabul eder:

- Degrade kutusu genişliği: degradenin yayılacağı genişlik (piksel cinsinden)
- Degrade kutusu yüksekliği: degradenin yayılacağı yükseklik (piksel cinsinden)
- Degrade kutusu dönüşü: degradeye uygulanacak dönüş (radyan olarak)
- Yatay çevirme: degradenin yatay olarak kaydırılacağı uzaklık (piksel cinsinden)
- Dikey çevirme: degradenin dikey olarak kaydırılacağı uzaklık (piksel cinsinden)

Örneğin, aşağıdaki özelliklere sahip bir degrade olduğunu varsayın:

- `GradientType.LINEAR`
- `ratios` dizisi `[0, 255]` olarak ayarlanmış yeşil ve mavi olmak üzere iki renk
- `SpreadMethod.PAD`
- `InterpolationMethod.LINEAR_RGB`

Aşağıdaki örnekler, `createGradientBox()` yönteminin `rotation` parametresinin, belirtildiği şekilde farklılık gösterdiği ancak diğer tüm ayarların aynı kaldığı degradeleri gösterir:

`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;`	

Aşağıdaki örnekler, `createGradientBox()` yönteminin `rotation`, `tx` ve `ty` parametrelerinin, belirtildiği şekilde farklılık gösterdiği ancak diğer tüm ayarların aynı kaldığı yeşilden maviye doğrusal degradedeki efektleri gösterir:

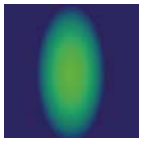
`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 0;`
`ty = 0;`	

`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;`	

`createGradientBox()` yönteminin `width`, `height`, `tx` ve `ty` parametreleri, aşağıdaki örnekte gösterildiği gibi, *radyal* degrade dolgunun boyutunu ve konumunu da etkiler:

`width = 50;`
`height = 100;`
`rotation = 0;`
`tx = 25;`
`ty = 0;`
	

Aşağıdaki kod, gösterilen son radyal degradeyi oluşturur:

```
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);
```

Degrade dolgunun genişlik ve yüksekliğinin, Graphics nesnesi kullanılarak çizilen genişlik ve yükseklikle değil, degrade matrisinin genişlik ve yüksekliğiyle belirlendiğini unutmayın. Graphics nesnesiyle çizim yaparken, degrade matrisindeki bu koordinatlarda bulunanları çizersiniz. `drawRect()` gibi Graphics nesnesinin şekil yöntemlerinden birini de kullansanız, degrade çizilen şeklin boyutuna genişlemez—degradenin boyutu, degrade matrisinde belirtilmelidir.

Aşağıda, degrade matrisinin boyutları ile çizimin kendi boyutları arasındaki görsel fark gösterilmektedir:

```
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);
```

Bu kod, eşit kırmızı, yeşil ve mavi dağılımıyla belirtilen aynı dolgu stiline sahip üç degrade çizer. Degradeler, sırayla 50, 100 ve 150 piksel genişliyle `drawRect()` yöntemi kullanılarak çizilir. `beginGradientFill()` yönteminde belirtilen degrade matris, 100 piksel genişliğinde oluşturulur. Başka bir deyişle, birinci degrade, degrade spektrumunun yalnızca yarısını içerir, ikinci degrade tamamını içerir ve üçüncü degrade de tamamını ve ek olarak sağa genişleyen 50 piksel mavi içerir.

`lineGradientStyle()` yöntemi, `beginGradientFill()` öğesine benzer şekilde çalışır, tek farkı, çizime başlamadan önce degradeyi tanımlamanın yanı sıra `lineStyle()` yöntemini kullanarak kontur kalınlığını da belirtmenizin gerekmesidir. Aşağıdaki kod, kırmızı, yeşil ve mavi degrade konturu ile bir kutu çizer:

```
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);
```

Matrix sınıfı hakkında daha fazla bilgi almak için, bkz. "Matrix nesnelerini kullanma" sayfa 219.

Çizim yöntemleriyle Math sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Graphics nesnesi daireler ve kareler çizer ancak özellikle de çizim yöntemleri, Math sınıfının özellik ve yöntemleriyle birlikte kullanıldığında daha karmaşık formlar da çizebilir. Math sınıfı, dairenin çevresinin çapına oranını belirten bir sabit olan `Math.PI` (yaklaşık 3.14159265...) gibi matematikle ilgili yaygın sabitleri içerir. Bu sınıf ayrıca `Math.sin()`, `Math.cos()` ve `Math.tan()` gibi trigonometri işlevlerine yönelik yöntemleri de içerir. Şekillerin bu yöntemler ve sabitler kullanılarak çizilmesi, özellikle de tekrarlama ve yineleme ile kullanıldığında daha dinamik görsel efektler oluşturur.

Math sınıfının birçok yöntemi, derece olarak değil radyan birimleri olarak dairesel ölçüler bekler. Math sınıfı yaygın olarak bu iki birim türü arasında dönüştürme yapmak için kullanılır.

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

Aşağıdaki örnek, belirli bir değer için `Math.sin()` ve `Math.cos()` yöntemleri arasındaki farkı vurgulamak üzere bir sinüs dalgası ve kosinüs dalgası oluşturur.

```
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);
}
```

Çizim API'si ile animasyon uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çizim API'si ile içerik oluşturmanın bir avantajı, içeriğinizi yalnızca bir defa konumlandırma sınırınızın olmamasıdır. Çizim yapmak için kullandığınız değişkenler korunarak veya değiştirilerek, çizdikleriniz değiştirilebilir. Değişkenleri değiştirip yeniden çizim yaparak, kareler dizisi üzerinden veya bir zamanlayıcı ile animasyon yürütebilirsiniz.

Örneğin, aşağıdaki kod, geçerli derece sayısını artırarak her ilerleyen kareyle görüntüyü değiştirir (`Event.ENTER_FRAME` olayını dinleyip) ve grafik nesnelerine silinmesini ve güncellenmiş konumla yeniden çizilmesini bildirir.

```
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;
}
```

Çok farklı bir sonuç üretmek için, kodun başındaki başlangıç tohum değişkenlerini (`currentDegrees`, `radius` ve `satelliteRadius`) değiştirerek deneme yapabilirsiniz. Örneğin, yarıçap değişkenini daraltmayı ve/veya totalSatellites değişkenini artırmayı deneyin. Bu, çizim API'sinin nasıl karmaşıklığı nedeniyle oluşturma basitliği gizlenen bir görüntü oluşturabildiğini gösteren örneklerden yalnızca biridir.

API çizme örneği: Algoritmik Görsel Oluşturucu

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Algoritmik Görsel Efekt Oluşturucu örneği, dinamik olarak sahne alanına birçok "uydu" veya dairesel bir yörüngede hareket eden daireler çizer. Keşfedilen özellikler arasında şunlar yer alır:

- Dinamik görünüme sahip temel bir şekil çizmek için çizim API'sini kullanma
- Çizimde kullanılan özelliklerle kullanıcı etkileşimini bağlama
- Her karede sahne alanını temizleyip yeniden çizim yaparak animasyon yürütme

Önceki alt bölümde verilen örnek, `Event.ENTER_FRAME` olayını kullanarak tek bir "uyduya" animasyon uygulamıştı. Bu örnek, birçok uydu görünümünü hemen güncelleyen kaydırıcıların bulunduğu bir kontrol paneli oluşturarak bu işlemi genişletir. Bu örnek, kodu harici sınıflar olarak biçimlendirir ve uydu oluşturma kodunu bir döngüye sararak `satellites` dizisindeki her uydunun başvurusunu saklar.

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Uygulama dosyalarını Samples/AlgorithmicVisualGenerator klasöründe bulabilirsiniz. Bu klasör şu dosyaları içerir:

File	Açıklama
AlgorithmicVisualGenerator.fla	Flash Professional'daki (FLA) ana uygulama dosyası.
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	Sahne alanına uyduları çizme ve uydu çizimini etkileyen değişkenleri güncellemek için kontrol panelinden gelen olayları yanıtlama gibi ana uygulama işlevlerini sağlayan sınıf.
com/example/programmingas3/algorithmic/ControlPanel.as	Birçok kaydırıcıyla kullanıcı etkileşimini yöneten ve kullanıcı etkileşimi oluştuğunda olaylar gönderen bir sınıf.
com/example/programmingas3/algorithmic/Satellite.as	Bir merkez noktası etrafındaki yörüngede dönen görüntüleme nesnesini temsil eden ve bu görüntüleme nesnesinin geçerli çizim durumuyla ilgili özellikleri içeren bir sınıf.

Dinleyicileri ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulama ilk olarak üç dinleyici oluşturur. Birinci dinleyici, kontrol panelinden gönderilen ve uyduların yeniden oluşturulmasının gerektiği olayları dinler. İkinci dinleyici, SWF dosyasının sahne alanının boyutuna yapılan değişiklikleri dinler. Üçüncü dinleyici, `doEveryFrame()` işlevi kullanılarak yeniden çizim yapılması için, SWF dosyasındaki ilerleyen her kareyi dinler.

Uyduları oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dinleyiciler ayarlandıktan sonra, `build()` işlevi çağrılır. Bu işlev, `satellites` dizisini boşaltıp sahne alanına yapılan önceki çizimleri temizleyen `clear()` işlevini çağırır. Bunun gerekli olmasının nedeni, kontrol paneli `build()` işlevini yeniden çağırmak için bir olay gönderdiğinde (örn. renk ayarları değiştirildiğinde) söz konusu işlevin yeniden çağrılabilmesidir. Bu durumda uyduların kaldırılıp yeniden oluşturulması gerekir.

Bu işlev daha sonra, yörüngede rastgele bir konumda başlayan `position` değişkeni ve bu örnekte uydu oluşturulduktan sonra değişmeyen `color` değişkeni gibi, oluşturma için gerekli başlangıç özelliklerini ayarlayarak uyduları oluşturur.

Her uydu oluşturulduğunda, `satellites` dizisine o uydunun başvurusu eklenir. `doEveryFrame()` işlevi çağrıldığında, bu dizideki tüm uydulara güncellenir.

Uydu konumunu güncelleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`doEveryFrame()` işlevi, uygulamanın animasyon işleminin kalbidir. SWF dosyasının kare hızına eşit bir hızda her kare için çağrılır. Çizim değişkenleri değiştiğinden, bu, animasyonun görünümünü yürütür.

Bu işlev ilk önce tüm çizimleri temizler ve arka planı yeniden çizer. Daha sonra bu işlev her uydu kabında döngü gerçekleştirir, her uydunun `position` özelliğini artırır ve kontrol paneliyle kullanıcı etkileşimi sonucunda değiştirilmiş olabilecek `radius` ve `orbitRadius` özelliklerini günceller. Son olarak uydu, Satellite sınıfının `draw()` yöntemini çağırarak yeni konumuna güncellenir.

Sayacın (`i`) yalnızca `visibleSatellites` değişkenine kadar artış yaptığını unutmayın. Bunun nedeni, kullanıcının kontrol paneli üzerinden görüntülenen sınırlı sayıda uyduya sahip olması durumunda, döngüdeki kalan uyduların yeniden çizilmeyip gizlenmesinin gerekmesidir. Bu, çizimden sorumlu döngüden hemen sonra gelen bir döngüde gerçekleşir.

doEveryFrame() işlevi tamamlandığında, ekranda `visibleSatellites` öğelerinin konumu güncellenir.

Kullanıcı etkileşimine yanıt verme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı etkileşimi, ControlPanel sınıfı tarafından yönetilen kontrol paneli aracılığıyla gerçekleşir. Bu sınıf her kaydırıcının minimum, maksimum ve varsayılan değerleriyle birlikte bir dinleyici ayarlar. Kullanıcı bu kaydırıcıları hareket ettirdikçe, `changeSetting()` işlevi çağrılır. Bu işlev, kontrol panelinin özelliklerini günceller. Değişiklik, görüntünün yeniden oluşturulmasını gerektiriyorsa, daha sonra ana uygulama dosyasında işlenen bir olay gönderilir. Kontrol paneli ayarları değiştikçe, `doEveryFrame()` işlevi her uyduyu güncellenmiş değişkenlerle çizer.

Daha fazla özelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnek, çizim API'si kullanılarak nasıl görsel efektler oluşturulduğuna yönelik temel bir şemadır. Bu, karmaşık görünen bir etkileşim deneyimi oluşturmak için nispeten daha az kod satırı kullanır. Yine de bu örnek küçük değişikliklerle genişletilebilir. Birkaç fikir:

- `doEveryFrame()` işlevi, uydunun renk değerini artırabilir.
- `doEveryFrame()` işlevi, uydu yarıçapını daraltabilir veya genişletebilir.
- Uydu yarıçapının dairesel olması gerekmez; örneğin bir sinüs dalgasına göre hareket etmek için Math sınıfını kullanabilir.
- Uydular diğer uydularla vuruş algılama özelliğini kullanabilir.

Çizim API'si, çalışma zamanında temel şekiller çizilerek, Flash geliştirme ortamında görsel efektler oluşturulmasına alternatif olarak kullanılabilir. Diğer yandan, elle oluşturulması mümkün olmayan çeşit ve kapsamda görsel efektler oluşturmak için de kullanılabilir. ActionScript yazarı, çizim API'si ve biraz matematik kullanarak birçok beklenmedik oluşturma işlemini gerçekleştirebilir.

Çizim API'sinin ileri düzey kullanımı

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player 10, Adobe AIR 1.5, ve daha sonraki Flash çalışma zamanları gelişmiş bir çizim özellikleri kümesini destekler. Bu çalışma zamanlarına ilişkin çizim API'si geliştirmeleri, önceki sürümlerdeki çizim yöntemleri esas alınarak genişletilmiştir, bu nedenle şekiller oluşturmak için veri kümeleri oluşturabilir, çalışma zamanında şekilleri değiştirebilir ve üç boyutlu efektler oluşturabilirsiniz. Çizim API'si geliştirmeleri, mevcut yöntemleri alternatif komutlarda birleştirir. Bu komutlar, çizim yöntemleri için veri kümeleri sağlamak üzere vektör dizilerini ve numaralandırma sınıflarını kullanır. Vektör dizilerinin kullanılması, daha karmaşık şekillerin hızlı şekilde oluşturulmasına ve geliştiricilerin çalışma zamanında dinamik şekil oluşturulması için programlama yoluyla dizi değerlerini değiştirmesine olanak sağlar.

Flash Player 10'da sunulan çizim özellikleri şu bölümlerde açıklanmıştır: "Yol Çizme" sayfa 239, "Sargı kurallarını tanımlama" sayfa 240, "Grafik verisi sınıflarını kullanma" sayfa 242 ve "drawTriangles() öğesini kullanma hakkında" sayfa 244.

Aşağıda, ActionScript'te ileri düzey çizim API'sini kullanarak gerçekleştirmek isteyebileceğiniz görevler yer almaktadır:

- Çizim yöntemlerine yönelik verileri saklamak için Vector nesnelerini kullanma
- Programlama yoluyla çizim şekillerinin yollarını tek bir işlemde tanımlama
- Örtüşen şekillerin nasıl doldurulduğunu belirlemek için sarma kurallarını tanımlama
- Bir görüntüleme nesnesinin vektör grafik içeriğini okuma: Grafik verilerini serileştirme ve kaydetme, çalışma zamanında hareketli grafik sayfası oluşturma veya vektör grafik içeriğinin bir kopyasını çizme vb.
- Üç boyutlu efektler için üçgenleri ve çizim yöntemlerini kullanma

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesinde, bu bölümde karşınıza çıkacak önemli terimler bulunmaktadır:

- Vektör: Tümü aynı veri türünde olan değerlerin oluşturduğu bir dizi. Vector nesnesi, çizim yöntemlerinin tek bir komutla çizgi ve şekil oluşturmak için kullandığı değerlerden oluşan bir diziyi saklayabilir. Vector nesneleri hakkında daha fazla bilgi için, bkz. "Dizinlenmiş diziler" sayfa 26.
- Yol: Yol, bir veya daha fazla düz ya da eğri parçadan oluşur. Her parçanın başlangıç ve bitişi, bir teli yerinde tutan çiviler gibi çalışan koordinatlarla işaretlenir. Bir yol kapalı (örneğin bir daire) olabilir veya açık olup ayrı uç noktalarına sahip olabilir (örneğin dalgalı bir çizgi).
- Sargı: Oluşturucu tarafından yorumlandığı haliyle yolun yönü; bu pozitif (saat yönünde) veya negatif (saat yönünün tersi) olabilir.
- GraphicsStroke: Çizgi stilinin ayarlanması için bir sınıf. "kontur" terimi, çizim API'si geliştirmelerinin bir parçası olmasa da, kendi dolgu özelliği olan bir çizgi stili belirlemek için bir sınıf kullanmak yeni çizim API'lerinin bir parçasıdır. GraphicsStroke sınıfını kullanarak bir çizginin stilini dinamik olarak ayarlayabilirsiniz.

- Fill nesnesi: `Graphics.drawGraphicsData()` çizim komutuna iletilen flash.display.GraphicsBitmapFill ve flash.display.GraphicsGradientFill gibi görüntüleme sınıfları kullanılarak oluşturulmuş nesneler. Fill nesneleri ve geliştirilen çizim komutları, `Graphics.beginBitmapFill()` ve `Graphics.beginGradientFill()` öğelerinin çoğaltılmasına daha nesne tabanlı bir programlama yaklaşımı sunar.

Yol Çizme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Çizgi ve eğrilerin çizilmesiyle ilgili bölümde (bkz.“Çizgi ve eğriler çizme” sayfa 226), tek bir çizgi (`Graphics.lineTo()`) veya eğri (`Graphics.curveTo()`) çizilmesine ve ardından bir şekil oluşturmak için çizginin başka bir noktaya hareket ettirilmesine (`Graphics.moveTo()`) yönelik komutlar açıklanmıştır. `Graphics.drawPath()` ve `Graphics.drawTriangles()` yöntemleri, söz konusu çizim komutlarının aynısını parametre olarak temsil eden bir nesne kümesini kabul eder. Bu yöntemlerle, Flash çalışma zamanının tek bir ifade içinde çalışması için `Graphics.lineTo()`, `Graphics.curveTo()` veya `Graphics.moveTo()` komutlarından bir seri sağlayabilirsiniz.

GraphicsPathCommand numaralandırma sınıfı, çizim komutlarına karşılık gelen bir sabit kümesini tanımlar. Bu sabitlerden oluşan bir seriyi (bir Vektör örneğinde sarılı) `Graphics.drawPath()` yöntemi için bir parametre olarak iletin. Daha sonra tek bir komutla bir tam bir şekil veya birçok şekil oluşturabilirsiniz. Ayrıca varolan bir şekli değiştirmek için bu yöntemlere iletilen değerleri değiştirebilirsiniz.

Çizim komutları Vektörü'nün yanı sıra, `drawPath()` yöntemi de her bir çizim komutu koordinatlarına karşılık gelen bir koordinat kümesi gerektirir. Koordinatlar (Numara örnekleri) içeren bir Vektör örneği oluşturun ve bunu `drawPath()` yöntemine ikinci (`data`) argüman olarak iletin.

Not: *Vektördeki değerler Point nesneleri değildir; vektör, iki sayıdan oluşan her grubun bir x/y koordinat çiftini temsil ettiği sayılar dizisidir.*

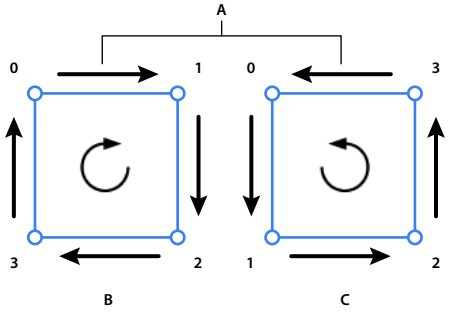
`Graphics.drawPath()` yöntemi, Graphics nesnesinde bir yol oluşturmak için her komutu ilgili nokta değerleriyle (iki veya dört sayıdan oluşan bir koleksiyon) eşleştirir:

```
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);
        }
    }
}
```

Sargı kurallarını tanımlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Gelişmiş çizim API'leri ayrıca yol "sarması" öğesini (bir yola giden yön) tanıtır. Bir yolun sargısı pozitif (saat yönünde) veya negatif (saat yönünün tersi) olur. Oluşturucunun veri parametreleri için vektör tarafından sağlanan koordinatları yorumlama sırası, sargıyı belirler.



Pozitif ve negatif sargı
A. *Oklar çizim yönünü belirtir* ***B.*** *Pozitif sargılı (saat yönünde)* ***C.*** *Negatif sargılı (saat yönünün tersi)*

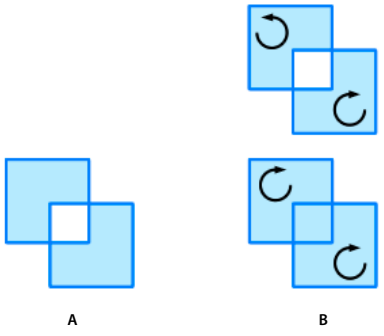
Ayrıca, `Graphics.drawPath()` yönteminin "sargı" adında isteğe bağlı üçüncü bir parametre içerdiğine dikkat edin:

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

Bu bağlamda üçüncü parametre, kesişen yollar için sargı veya dolgu kuralını belirten bir dize veya sabittir. (GraphicsPathWinding sınıfında sabit değerleri `GraphicsPathWinding.EVEN_ODD` veya `GraphicsPathWinding.NON_ZERO` olarak tanımlanır.) Yollar kesiştiğinde sargı kuralı önemlidir.

Çift-tek kuralı standart sargı kuralı olup eski çizim API'leri tarafından kullanılır. Çift-tek kuralı aynı zamanda `Graphics.drawPath()` yöntemi için de varsayılan kuraldır. Çift-tek kuralıyla, kesişen yollar açık ve kapalı dolgular arasında geçiş yapar. İki kare aynı dolgu kesişimiyle çizilirse, kesişimin oluştuğu alan doldurulur. Genellikle bitişik alanların ya ikisi de doldurulur ya ikisi de doldurulmaz.

Sıfır dışı kuralı ise, kesişen yollar tarafından tanımlanan alanların doldurulup doldurulmayacağını belirlemek için sargıyı (çizim yönü) esas alır. Zıt sargı yolları kesiştiğinde, çift-tek kuralına benzer şekilde tanımlanan alan doldurulmaz. Aynı sargıya sahip yollar için, doldurulmayacak alan doldurulur:

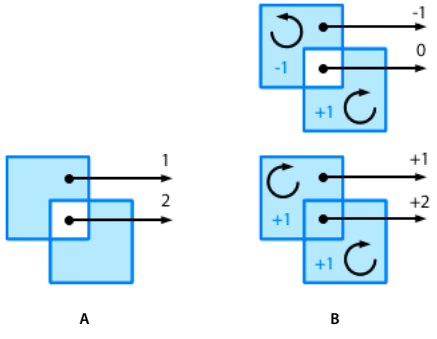


Kesişen alanlar için sargı kuralları
***A.** Çift-tek sargı kuralı **B.** Sıfır dışı sargı kuralı*

Sargı kuralı adları

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Adlar, bu dolguların nasıl yönetildiğini tanımlayan daha belirli bir kuralı ifade eder. Pozitif sargılı yollara +1 değeri atanır; negatif sargılı yollara ise -1 değeri atanır. Bir şeklin çevrelenmiş alanındaki noktadan başlayarak o noktadan itibaren belirsiz şekilde genişleyen bir çizgi çizin. Dolguyu belirlemek için çizginin bir yoldan geçme sayısı ve bu yolların birleştirilmiş değerleri kullanılır. Çift-tek sargısı için, çizginin bir yoldan geçme sayısı kullanılır. Sayı tek olduğunda alan doldurulur. Çift sayı olduğunda alan doldurulmaz. Sıfır dışı sargı için, yollara atanan değerler kullanılır. Yolun birleştirilmiş değerleri 0 olmadığında, alan doldurulur. Birleştirilmiş değer 0 olduğunda alan doldurulmaz.



Sargı kuralı sayıları ve dolgular
***A.** Çift-tek sargı kuralı **B.** Sıfır dışı sargı kuralı*

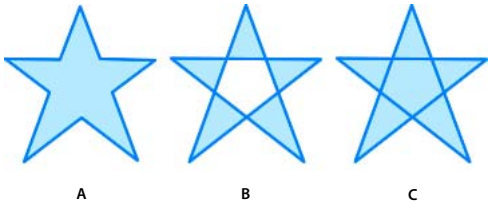
Sargı kurallarını kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bu dolgu kuralları karmaşıktır ancak bazı durumlarda gereklidir. Örneğin, bir yıldız şekli çizdiğinizi düşünün. Standart çift-tek kuralıyla şekil için on çizgi gerekir. Sıfır dışı sargı kuralıyla bu on çizgi dokuza indirilir. Burada beş çizgili bir yıldız için ActionScript ve sıfır dışı sargı kuralı verilmiştir:

```
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);
```

Ve burada da yıldız şekli verilmiştir:



Farklı sargı kuralları kullanan bir yıldız şekli
***A.** Çift-tek 10 çizgi **B.** Çift-tek 5 çizgi **C.** Sıfır dışı 5 çizgi*

Ve, görüntülere animasyon uygulandıkça veya görüntüler üç boyutlu nesnelerde doku olarak kullanılıp örtüştükçe, sargı kuralları daha çok önem kazanır.

Grafik verisi sınıflarını kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Gelişmiş çizim API'si, IGraphicsData arabirimini başlatan flash.display paketinde bir sınıf kümesi içerir. Bu sınıflar, çizim API'sinin çizim yöntemlerini temsil eden değer nesneleri (veri kapsayıcıları) olarak davranırlar.

Şu sınıflar, IGraphicsData arabirimini uygular:

- GraphicsBitmapFill
- GraphicsEndFill
- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

Bu sınıflarla, IGraphicsData (Vector.<IGraphicsData>) türündeki bir Vektör nesnesinde eksiksiz bir çizimdepolayabilirsiniz. Ardından, bu grafik verilerini başka şekil örnekleri için veya çizim bilgilerini daha sonra kullanmak üzere depolama amacıyla veri kaynağı olarak yeniden kullanabilirsiniz.

Her stil veya dolgu için birden çok dolgu sınıfınız olduğu halde yalnızca bir kontur sınıfına sahip olduğunuza dikkat edin. Kontur sınıfı kendi stilini tanımlamak için dolgu sınıflarını kullandığından, ActionScript, yalnızca bir kontur IGraphicsData sınıfına sahiptir. Bu nedenle her kontur, kontur sınıfı ve dolgu sınıfıyla birlikte tam olarak tanımlanır. Ya da, bu grafik veri sınıflarının API'si, flash.display.Graphics sınıfında temsil ettikleri yöntemleri yansıtır:

Graphics Yöntemi	Karşılık Gelen Sınıf
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

Ayrıca, GraphicsPath sınıfının GraphicsPath örneği için bu komutları kolayca tanımlamak üzere kendi `GraphicsPath.moveTo()`, `GraphicsPath.lineTo()`, `GraphicsPath.curveTo()`, `GraphicsPath.wideLineTo()` ve `GraphicsPath.wideMoveTo()` yardımcı program yöntemleri vardır. Bu yardımcı program yöntemleri, komutların ve veri değerlerinin doğrudan tanımlanması ya da güncellenmesi görevini kolaylaştırır.

Vektör grafik verileriyle çizim yapma

IGraphicsData örneklerinin bir koleksiyonuna sahip olduktan sonra, grafik oluşturmak için Graphics sınıfının `drawGraphicsData()` yöntemini kullanın. `drawGraphicsData()` yöntemi, IGraphicsData örneği vektöründeki bir çizim talimatı kümesini ardışık sırada uygular:

```
// 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);
```

Daha karmaşık bir görüntü için, örnekteki çizim tarafından kullanılan yolda tek bir değer değiştirilerek şekil birden çok defa yeniden çizilebilir:

```
// draw the drawing multiple times
// change one value to modify each variation
graphics.drawGraphicsData(drawing);
path.data[2] += 200;
graphics.drawGraphicsData(drawing);
path.data[2] -= 150;
graphics.drawGraphicsData(drawing);
path.data[2] += 100;
graphics.drawGraphicsData(drawing);
path.data[2] -= 50;graphicsS.drawGraphicsData(drawing);
```

IGraphicsData nesneleri dolgu ve kontur stillerini tanımlayabilse de, dolgu ve kontur stilleri bir gereklilik değildir. Başka bir deyişle, yolların kaydedilmiş bir koleksiyonunu çizmek için IGraphicsData nesneleri kullanılabilirken, stilleri ayarlamak için Graphics sınıfı yöntemleri kullanılabilir veya tam tersi de geçerlidir.

Not: *Orijinal çizime ekleme yapmıyorsanız, yukarıdaki örnekte görüldüğü gibi, yenisine başlamadan önce bir önceki çizimi temizlemek için `Graphics.clear()` yöntemini kullanın. Bir yolun tek bir kısmını veya IGraphicsData nesnelerinin koleksiyonunu değiştirdiğinizde, değişiklikleri görmek için çizimin tamamını yeniden çizin.*

Grafik veri sınıfları kullanılırken, şekil doğası gereği o noktada kapandığından, üç veya daha fazla nokta çizildiğinde dolgu oluşturulur. Dolgu kapansa da kontur kapanmaz ve bu davranış birden çok `Graphics.lineTo()` veya `Graphics.moveTo()` komutu kullanılırken gerçekleşen davranıştan farklıdır.

Vektör grafik verilerini okuma

Flash Player 11.6 ve üstü, Adobe AIR 3.6 ve üstü

Flash Player 11.6'da ve Adobe AIR 3.6 ile üst sürümlerinde, bir görüntüleme nesnesine yönelik çizim vektör içeriğinin yanı sıra, bir görüntüleme nesnesinin vektör grafik içeriğine ilişkin veri temsili elde etmek üzere Graphics sınıfının `readGraphicsData()` yöntemini kullanabilirsiniz. Bu, çalışma zamanında hareketli grafik sayfası kaydetmek, kopyalamak, oluşturmak ve daha fazlasını yapmak üzere bir grafik anlık görüntüsü oluşturmak için kullanılabilir.

`readGraphicsData()` yöntemi çağrıldığında, IGraphicsData nesneleri içeren bir Vektör örneği döndürülür. Bunlar, `drawGraphicsData()` yöntemiyle vektör grafikleri çizmek için kullanılan aynı nesnelerdir.

Vektör grafiklerinin `readGraphicsData()` yöntemiyle okunmasına yönelik bazı sınırlamalar bulunur. Daha fazla bilgi için bkz. ActionScript Dil Başvurusu'nda readGraphicsData() girişi.

drawTriangles() öğesini kullanma hakkında

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player 10 ve Adobe AIR 1.5 `Graphics.drawTriangles()`, `Graphics.drawPath()` yöntemine benzer. `Graphics.drawTriangles()` yöntemi de bir Vektör kullanır.<Number> nesnesini de kullanır.

Ancak, `Graphics.drawTriangles()` yönteminin gerçek amacı, ActionScript üzerinden üç boyutlu efektlerin kolaylaştırılmasıdır. Üç boyutlu efektler üretmek için `Graphics.drawTriangles()` öğesini kullanma hakkında bilgi almak için, bkz. “3B efektler için üçgenleri kullanma” sayfa 365.

Bölüm 13: Bitmap'lerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, vektör çizme yeteneklerine ek olarak, bitmap görüntüler oluşturma veya bir SWF dosyasına yüklenen harici bitmap görüntülerin piksel verilerini işleme yeteneğini de içerir. Piksel değerlerine erişip onları değişebilmeniz sayesinde kendi filtre benzeri görüntü efektlerinizi oluşturabilir, doku ve rast gele gürültü oluşturmak için dahili gürültü işlevlerini kullanabilirsiniz.

- Renaun Erickson: Rendering game assets in ActionScript using blitting techniques (Doldurarak oluşturma tekniklerini kullanarak ActionScript'te oyun varlıkları oluşturma)
- Bitmap programming: Chapter 26 of Essential ActionScript 3 by Colin Moock (O'Reilly Media, 2007) (Essential ActionScript 3, Bölüm 26, Colin Moock)
- Mike Jones: Working with Sprites in Pushbutton Engine (Basma Düğmesi Motorunda Hareketli Grafiklerle Çalışma)
- Flash & Math: Pixel Particles Made Simple (Flash ve Matematik: Basitleştirilmiş Piksel Parçacıkları)
- Flixel

Bitmap'lerle çalışmanın temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dijital görüntülerle çalışırken, karşınıza iki ana grafik türü çıkar: bitmap ve vektör. Raster grafikler olarak da bilinen bitmap grafikleri dikdörtgen bir ızgara biçiminde düzenlenmiş çok küçük karelerden (piksellerden) oluşur. Vektör grafikler çizgiler, eğriler ve çokgenler gibi matematiksel geometrik şekillerden oluşur.

Bitmap görüntüler, görüntünün piksel cinsinden genişliği ve yüksekliği ile her bir pikselin içinde yer alan ve o pikselin içerebileceği renk sayısını temsil eden bit sayısı ile tanımlanır. RGB renk modelini kullanan bir bitmap görüntüsünde pikseller kırmızı, yeşil ve mavi olmak üzere üç bayttan oluşur. Bu baytların her biri 0 ile 255 arasında bir değere sahiptir. Piksel içindeki baytlar birleştiğinde, bir ressamın boyaları karıştırmasına benzer şekilde ortaya bir renk çıkarırlar. Örneğin, kırmızı-255, yeşil-102 ve mavi-0 bayt değerlerini içeren bir piksel canlı bir turuncu renk ortaya çıkaracaktır.

Bir bitmap görüntünün kalitesi, görüntünün çözünürlüğü ve renk derinliğinin bit değeri ile tanımlanır. *Çözünürlük* bir görüntü içerisinde bulunan piksel sayısı ile ilgilidir. Piksel sayısı ne kadar yüksek olursa, çözünürlük o kadar yüksek olur ve görüntü o kadar iyi görünür. *Renk derinliği* bir pikselin içerebileceği bilgi miktarı ile ilgilidir. Örneğin, 16 bit/piksel değerinde renk derinliğine sahip bir görüntü 48 bit renk derinliğine sahip bir görüntü ile aynı sayıda rengi gösteremez. Sonuç olarak, 48-bit görüntünün gölgeleme derecesi 16-bit görüntüden daha düzgün olacaktır.

Bitmap grafikler çözünürlüğe bağlı olduğundan, iyi şekilde ölçeklenemezler. Bu durum özellikle bitmap görüntülerin boyutları artırıldığında fark edilir. Bir bitmap'in boyutunu artırmak genelde detay ve kalite kaybına neden olacaktır.

Bitmap dosya biçimleri

Bitmap görüntüler birçok yaygın dosya biçimine ayrılmıştır. Bu biçimler dosya boyutunu düşürmek ve görüntünün son kullanım amacına uygun olarak görüntü kalitesini optimize etmek için farklı sıkıştırma algoritmaları kullanır. Adobe çalışma zamanları tarafından desteklenen bitmap görüntü biçimleri BMP, GIF, JPG, PNG, ve TIFF'dir.

BMP

BMP (bit eşlemli) biçim, Microsoft Windows işletim sistemi tarafından kullanılan varsayılan görüntü biçimidir. Herhangi bir sıkıştırma algoritması şeklini kullanmaz ve bu nedenle de genellikle büyük dosya boyutlarına neden olur.

GIF

GIF (Graphics Interchange Format) biçimi 1987 yılında CompuServe tarafından 256 renkli (8-bit renk) görüntüleri iletmek için geliştirilmiştir. Bu biçim düşük dosya boyutları sunar ve web-tabanlı görüntüler için idealdir. GIF biçimi, renk paletinin sınırlı olmasından dolayı genelde yüksek gölgeleme derecesi ve renk degradeleri gerektiren fotoğraflar için uygun değildir. GIF görüntüler renklerin şeffaf (veya saydam) olarak eşlenmesine imkan tanıyan tek-bit saydamlığı destekler. Bunun sonucunda resmin saydamlığın eşlendiği kısımlarından web sayfasının arka plan rengi görünür.

JPEG

Joint Photographic Experts Group (JPEG) tarafından geliştirilmiş olan JPEG (genelde JPG şeklinde yazılır) görüntü biçimi küçük dosya boyutunda 24-bit renk derinliği kullanılabilmesi için kayıplı sıkıştırma algoritması kullanır. Kayıplı sıkıştırma uygulanan bir görüntü her kaydedildiğinde kalite ve veri kaybına uğrar ancak bunun sonucunda dosya boyutu düşer. JPEG biçimi fotoğraflar için idealdir çünkü milyonlarca renk görüntüleyebilir. Bir görüntüye uygulanan sıkıştırmanın derecesini kontrol ederek görüntü kalitesini ve dosya boyutunu değiştirebilirsiniz.

PNG

PNG (Portable Network Graphics) biçimi patentli GIF biçimine açık kaynaklı bir alternatif olarak geliştirilmiştir. PNG biçimi en fazla 64-bit renk derinliğini destekleyerek en fazla 16 milyon rengin kullanılmasına izin verir. PNG görece olarak daha yeni bir biçim olduğundan, eski tarayıcıların bazıları PNG dosyalarını desteklemez. JPG biçiminden farklı olarak PNG biçimi kayıpsız sıkıştırma kullanır, yani görüntü kaydedildiğinde görüntü verileri kaybolmaz. PNG dosyaları ayrıca 256 düzeye kadar saydamlığa izin veren alfa saydamlığını destekler.

TIFF

PNG'den önce tercih edilen platformlar arası biçim, Etiketli Görüntü Dosyası Biçimi (TIFF) idi. TIFF biçimiyle ilgili en önemli sorun çok sayıda farklı TIFF sürümü olması nedeniyle her sürümü işleyebilecek tek bir okuyucu olmamasıdır. Ayrıca, web tarayıcıları geçerli olarak bu biçimi desteklememektedir. TIFF, kayıplı veya kayıpsız sıkıştırma kullanabilir ve aygıta özgü renk alanlarını (örn. CMYK) işleyebilir.

Saydam bitmap'ler ve opak bitmap'ler

GIF veya PNG biçimlerinden birini kullanan bitmap görüntülerin piksellerinin her birinde ekstra bir bit (alfa kanalı) olabilir. Bu ekstra piksel baytı, pikselin saydamlık değerini temsil eder.

GIF görüntüleri tek-bit saydamlığa izin verir yani 256 renkli paletten istediğiniz bir rengi saydam olarak belirleyebilirsiniz. Diğer taraftan PNG görüntüleri 256 düzeye kadar saydamlığa sahip olabilir. Bu işlev özellikle görüntünün veya metnin arka planla harmanlanması gerektiğinde faydalıdır.

ActionScript 3.0 BitmapData sınıfındaki bu ekstra saydamlık piksel baytını çoğaltır. PNG saydamlık modeliyle benzer şekilde ActionScript 256 düzeye kadar saydamlık sunar.

Önemli kavramlar ve terimler

Aşağıdaki liste bitmap grafiklerini öğrenirken karşılaşacağınız önemli terimleri içerir

Alfa Bir rengin veya görüntünün saydamlık (daha doğrusu opaklık) düzeyi. Alfa miktarı genelde *alfa kanalı* değeri olarak belirtilir.

ARGB renk Her bir pikselin renginin kırmızı, yeşil ve mavi renk değerlerinin karışımından oluştuğu ve saydamlığının bir alfa değeri olarak belirtildiği renk düzeni.

Renk kanalı Genellikle, renkler birkaç temel rengin karışımından oluşur - bunlar genelde (bilgisayar grafikleri için) kırmızı, yeşil ve mavi renklerdir. Temel renklerin her biri bir renk kanalı olarak kabul edilir ve renk kanallarının her birindeki renk miktarı bir araya gelerek son rengi belirler.

Renk derinliği *Bit derinliği* olarak da bilinen renk derinliği her bir piksele adanmış olan ve böylece görüntüde gösterilebilecek renk sayısını belirleyen bilgisayar belleği miktarı ile ilgilidir.

Piksel Bir bitmap görüntüdeki en küçük bilgi birimi, bir renk noktasıdır.

Çözünürlük Görüntünün, görüntüdeki ince ayrıntı düzeyini belirleyen piksel cinsinden boyutlarıdır. Çözünürlük genellikle genişlik ve yüksekliğin piksel sayısı olarak belirtilir.

RGB Renk Her bir pikselin renginin kırmızı, yeşil ve mavi renk değerlerinin karışımı olarak belirtildiği bir renk düzenidir.

Bitmap ve BitmapData sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bitmap sınıflarıyla çalışmaya yarayan ana ActionScript 3.0 sınıfları ekranda bitmap görüntüleri görüntülemek için kullanılan Bitmap sınıfı ve bir bitmap'in ham görüntü verisine erişmek ve değiştirmek için kullanılan BitmapData sınıfıdır.

Daha fazla Yardım konusu

flash.display.Bitmap

flash.display.BitmapData

Bitmap sınıfını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

DisplayObject sınıfının bir alt sınıfı olan Bitmap sınıfı bitmap görüntüleri görüntülemek için kullanılan ana ActionScript 3.0 sınıfıdır. Görüntüler flash.display.Loader sınıfı tarafından yüklenmiş veya `Bitmap()` yapıcısını kullanarak dinamik olarak oluşturulmuş olabilir. Harici bir kaynaktan görüntü yüklerken, bir Bitmap nesnesi sadece GIF, JPEG veya PNG biçimindeki görüntüleri kullanabilir. Bitmap örneği oluşturulduktan sonra, Sahne alanında oluşturulması gereken bir BitmapData nesnesi için bir sarıcı olarak düşünülebilir Bir Bitmap örneği aynı zamanda bir görüntüleme nesnesi olduğundan, görüntüleme nesnelerinin özelliklerinin ve işlevselliklerinin tümü bir Bitmap örneğini düzenlemek için de kullanılabilir. Görüntüleme nesneleri ile çalışma hakkında daha fazla bilgi için "Görüntü programlama" sayfa 153 bölümünü inceleyin.

Piksel yapışması ve düzgünleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tüm görüntüleme nesnelerinde ortak olan işlevselliğe ek olarak, Bitmap sınıfı, bitmap görüntülere has bazı ek özellikler sunar.

Bitmap sınıfının `pixelSnapping` özelliği bir Bitmap nesnesinin ona en yakın piksele yapışıp yapışmadığını belirler. Bu özellik PixelSnapping sınıfında tanımlanmış olan `ALWAYS`, `AUTO` veya `NEVER` sabitini kabul eder.

Piksel yapışması uygulama için sözdizimi şu şekildedir:

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

Genelde, bitmap görüntüler ölçeklendiğinde bulanıklaşır ve deforme olur. Bu deformasyonu azaltmak için BitmapData sınıfının `smoothing` özelliğini kullanın. Bu Boolean özelliğine `true` değeri verildiği zaman bir görüntü ölçeklendiğinde pikselleri düzgünleştirir veya kenarlarını yumuşatır. Bu da görüntünün daha net ve daha doğal görünmesini sağlar.

BitmapData sınıfını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

flash.display paketi içindeki BitmapData sınıfı yüklenen veya dinamik olarak oluşturulan bir bitmap görüntüsü içerisindeki piksellerin fotografik anlık görüntüsüne benzetilebilir. Bu anlık görüntü, nesne içindeki piksel verilerinin dizisi ile ifade edilir. BitmapData sınıfı ayrıca piksel verisi oluşturma ve düzenleme için yardımcı olan bir dizi dahili yöntem içerir.

Bir BitmapData nesnesi oluşturmak için aşağıdaki kodu kullanın:

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

`Width`ve`height`parametreleri bitmap'in boyutunu belirler. AIR 3 ve Flash Player 11'den başlamak üzere, BitmapData nesnesine yönelik boyut sınırları kaldırıldı. Maksimum bitmap boyutu işletim sistemine bağlıdır.

AIR 1.5 ve Flash Player 10'da, bir BitmapData nesnesi için maksimum boyut genişlikte ve yükseklikte 8.191 pikseldir ve toplam piksel sayısı 16.777.215'i geçemez. (Bu nedenle, bir BitmapData nesnesi 8.191 piksel genişliğindeyse en fazla 2.048 piksel yüksekliğinde olabilir.) Flash Player 9 ve önceki sürümleri ile AIR 1.1 ve önceki sürümlerinde, sınırlama yükseklikte 2.880 piksel ve genişlikte 2.880 pikseldir.

`transparent` parametresi, bitmap verisinin bir alfa kanalı içerip (`true`) içermeyeceğini (`false`) belirler. `fillColor` parametresi arka plan rengini ve saydamlık değerini (`true` değeri verilmişse) belirleyen 32-bit renk değeridir. Aşağıdaki örnek yüzde 50 oranında saydam turuncu arka plana sahip bir BitmapData nesnesi oluşturur:

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

Yeni oluşturulan bir BitmapData nesnesini ekranda işlemek için, bir Bitmap örneğine atayın veya sarın. Bunu yapmak için, BitmapData nesnesini Bitmap nesnesinin yapıcısının bir parametresi olarak iletebilir veya varolan bir Bitmap örneğinin `bitmapData` özelliğine atayabilirsiniz. Ayrıca görüntüleme nesnesi kabının Bitmap örneği içeren `addChild()` veya `addChildAt()` yöntemlerini çağırarak görüntüleme listesine Bitmap örneğini de eklemeniz gerekir. Görüntüleme listesi ile çalışma hakkında daha fazla bilgi için, "Görüntüleme listesine görüntüleme nesneleri ekleme" sayfa 161 bölümünü inceleyin.

Aşağıdaki örnek kırmızı dolgulu bir BitmapData nesnesi oluşturur ve bir Bitmap örneği içerisinde görüntüler.

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

Piksellerde değişiklik yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

BitmapData sınıfı, piksel veri değerlerini değiştirmenize imkan tanıyan bir dizi yöntem içerir.

Piksellerde tek tek değişiklik yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir bitmap görüntünün görünümünü piksel düzeyinde değiştirirken, öncelikle değişiklik yapmak istediğiniz alandaki piksellerin renk değerlerini bilmeniz gerekir. Bu piksel değerleri `getPixel()` yöntemi ile okunur.

`getPixel()` yöntemi parametre olarak iletilen x, y (piksel) koordinatları kümesinden bir RGB değeri alır. Değişiklik yapmak istediğiniz piksellerden herhangi biri saydamlık (alfa kanalı) bilgisi içeriyorsa, `getPixel32()` yöntemini kullanmanız gerekir. Bu yöntem de bir RGB değeri alır ancak `getPixel()` yönteminden farklı olarak `getPixel32()` sonucunda dönen değer seçilen pikselin alfa kanalını (saydamlığını) temsil eden ek veri içerir.

Ayrıca, bir bitmap içerisinde bulunan piksellerden herhangi birinin rengini veya saydamlık değerini değiştirmek isterseniz, `setPixel()` veya `setPixel32()` yöntemini kullanabilirsiniz. Bir pikselin rengini ayarlamak için, bu yöntemlerden birine x, y koordinatlarını ve renk değerini iletin.

Aşağıdaki örnek yeşil bir BitmapData arka planı üzerine bir artı çizmek için `setPixel()` yöntemini kullanır. Ardından 50, 50 koordinatlarında bulunan pikselden renk değerini almak için `getPixel()` yöntemini kullanır ve dönen değeri izler.

```
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));
```

Tek bir piksel yerine bir piksel grubunun değerini okumak isterseniz, `getPixels()` yöntemini kullanın. Bu yöntem bir parametre olarak iletilen piksel verilerinin dikdörtgen bölgesinden bir bayt dizisi oluşturur. Bayt dizisinin (başka bir deyişle piksel değerlerinin) öğelerinden her biri birer işaretsiz tam sayı - 32-bit, çarpılmamış piksel değeridir.

Bir piksel grubunun değerini değiştirmek (veya belirlemek) için ise `setPixels()` yöntemini kullanın. Bu yöntem dikdörtgen şeklinde (`rect`) bir piksel verisi (`inputByteArray`) çıktısı oluşturmak için birleştirilen iki parametre (`rect` ve `inputByteArray`) bekler.

Veriler `inputByteArray` içerisinden okunduğu (ve içerisine yazıldığı) için `ByteArray.readUnsignedInt()` yöntemi dizi içindeki piksellerin her biri için çağrılır. Herhangi bir nedenden dolayı `inputByteArray` piksel değerine eşdeğer tam bir dikdörtgen içermezse, yöntem görüntü verisi işlemeyi o noktada durdurur.

Piksel verisini alırken ve ayarlarken bayt dizisinin 32-bit alfa, kırmızı, yeşil ve mavi (ARGB) piksel değerlerini beklediği unutulmamalıdır.

Aşağıdaki örnek, bir BitmapData nesnesindeki piksel grubunu başka bir BitmapData nesnesine kopyalamak için `getPixels()` ve `setPixels()` yöntemlerini kullanır:

```
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;
```

Piksel düzeyinde çakışma saptama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`BitmapData.hitTest()` yöntemi bitmap verileri ile başka bir nesne veya nokta arasında piksel düzeyinde çakışma saptaması gerçekleştirir.

`BitmapData.hitTest()` yöntemi beş parametre kabul eder:

- `firstPoint` (Point): Bu parametre ilk BitmapData'nın vuruş testinin gerçekleştirildiği sol üst köşesinin piksel konumudur.
- `firstAlphaThreshold` (uint): Bu parametre, bu vuruş testi için opak olarak kabul edilen en yüksek alfa kanalı değerini belirtir.
- `secondObject` (Object): Bu parametre etki alanını temsil eder. `secondObject` nesnesi Rectangle, Point, Bitmap veya BitmapData nesnesi olabilir. Bu nesne, gerçekleştirilen çakışma saptamanın vuruş alanını temsil eder.
- `secondBitmapDataPoint` (Point): İsteğe bağlı olan bu parametre ikinci BitmapData nesnesinde bir piksel konumunu tanımlamak için kullanılır. Bu parametre yalnızca `secondObject` değeri bir BitmapData nesnesi olduğunda kullanılır. Varsayılan değer `null` şeklindedir.
- `secondAlphaThreshold` (uint): İsteğe bağlı bu parametre ikinci BitmapData nesnesinde opak olarak kabul edilen en yüksek alfa kanalı değerini temsil eder. Varsayılan değer 1'dir. Bu parametre yalnızca `secondObject` değeri bir BitmapData nesnesi olduğunda ve her iki BitmapData nesnesi de saydam olduğunda kullanılır.

Opak görüntülerde çakışma saptama gerçekleştirirken, ActionScript'in görüntüyü tamamen opak bir dikdörtgen (veya sınırlama kutusu) olarak değerlendirdiğini unutmayın. Ayrıca, saydam görüntülerde piksel düzeyinde vuruş testi yaparken her iki görüntünün de saydam olması gerekir. Buna ek olarak, ActionScript piksellerin hangi noktada saydamlıktan opaklığa geçiş yaptığını belirlemek için alfa eşiği parametrelerini kullanır.

Aşağıdaki örnek üç bitmap görüntüsü oluşturur ve iki farklı çakışma noktası kullanarak piksel çarpışmalarını kontrol eder. (Biri false diğeri true değeri verir.)

```
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
```

Bitmap verilerini kopyalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir görüntüden diğerine bitmap verilerini kopyalamak için birkaç yöntem kullanabilirsiniz: `clone()`, `copyPixels()`, `copyChannel()`, `draw()` ve `drawWithQuality()` (`drawWithQuality` yöntemi Flash Player 11.3 ve üstünde; AIR 3.3 ve üstünde mevcuttur).

Adından da anlaşılacağı üzere, `clone()` yöntemi bir BitmapData nesnesinden bir diğerine bitmap verilerini klonlamanızı veya örneklemenizi sağlar. Yöntem çağrıldığında orijinal örneğin birebir kopyası olan yeni bir BitmapData nesnesi döndürür.

Aşağıdaki örnek turuncu (üst) kareyi klonlar ve bu kopyayı orijinal üst karenin arkasında yerleştirir:

```
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;
```

`copyPixels()` yöntemi bir BitmapData nesnesinden bir diğerine piksel kopyalamanın hızlı ve kolay bir yoludur. Bu yöntem kaynak görüntünün dikdörtgen şeklinde bir anlık görüntüsünü alır (`sourceRect` parametresi ile tanımlanır) ve dikdörtgen şeklindeki başka bir alana (aynı boyutta) kopyalar. Yeni "yapıştırılan" dikdörtgenin konumu `destPoint` parametresi ile tanımlanır.

`copyChannel()` yöntemi kaynak BitmapData nesnesinden önceden tanımlı bir renk kanalı değeri (alfa, kırmızı, yeşil veya mavi) örnekler ve hedef BitmapData nesnesinin kanalına kopyalar. Bu yöntemi çağırmak hedef BitmapData nesnesinin diğer kanallarını etkilemez.

`draw()` ve `drawWithQuality()` yöntemi; bir kaynak hareketli grafik, film klibi veya başka bir görüntüleme nesnesindeki grafik içeriğini yeni bir bitmap üzerinde çizer veya oluşturur. `matrix`, `colorTransform`, `blendMode` ve `clipRect` parametrelerini kullanarak yeni bitmap'in oluşturulma şeklini değiştirebilirsiniz. Bu yöntem verileri oluşturmak için Flash Player ve AIR içerisindeki vektör oluşturucuyu kullanır.

`draw()` veya `drawWithQuality()` öğesini çağırdığınızda kaynak nesneyi (hareketli grafik, film klibi veya başka bir görüntüleme nesnesi) burada gösterildiği şekilde birinci parametre olarak iletirsiniz:

```
myBitmap.draw(movieClip);
```

Kaynak nesne ilk yüklendikten sonra herhangi bir dönüştürme (renk, matrix, vs.) uygulandıysa, bu dönüştürmeler yeni nesneye kopyalanmaz. Dönüştürmeleri yeni bitmap'e kopyalamak isterseniz, orijinal nesnenin `transform` özelliği değerini yeni BitmapData nesnesini kullanan Bitmap nesnesinin `transform` özelliğine kopyalamanız gerekir.

Bitmap verilerini sıkıştırma

Flash Player 11.3 ve üstü, AIR 3.3 ve üstü

`flash.display.BitmapData.encode()` yöntemi, bitmap verilerini aşağıdaki görüntü sıkıştırma biçimlerinden birine yerel olarak sıkıştırmanıza olanak tanır:

- **PNG** - Dosya boyutu yerine sıkıştırma hızını vurgulayan PNG sıkıştırmasını kullanır (isteğe bağlı olarak hızlı sıkıştırmayı kullanarak). PNG sıkıştırmasını kullanmak için yeni bir `flash.display.PNGEncoderOptions` nesnesini `BitmapData.encode()` yönteminin ikinci parametresi olarak iletin.
- **JPEG** - İsteğe bağlı olarak görüntü kalitesini belirterek JPEG sıkıştırmasını kullanır. JPEG sıkıştırmasını kullanmak için yeni bir `flash.display.JPEGEncoderOptions` nesnesini `BitmapData.encode()` yönteminin ikinci parametresi olarak iletin.

- **JPEGXR** - İsteğe bağlı olarak renk kanalını, kaybı olanları ve entropi ayarlarını belirterek JPEG Genişletilmiş Aralık (XR) sıkıştırmasını kullanır. JPEGXR sıkıştırmasını kullanmak için yeni bir `flash.display.JPEGXREncoderOptions` nesnesini `BitmapData.encode()` yönteminin ikinci parametresi olarak iletin.

Bu özelliği, sunucu yükleme veya indirme iş akışının bir parçası olarak görüntü işleme amacıyla kullanabilirsiniz.

Aşağıdaki örnek parçacık `JPEGEncoderOptions` öğesini kullanarak bir BitmapData nesnesi sıkıştırır:

```
// 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);
```

Gürültü işlevleri ile doku oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir bitmap'e, görünümünü değiştirmek için `noise()` veya `perlinNoise()` yöntemini kullanarak gürültü efekti uygulayabilirsiniz. Gürültü efekti, karıncalı gösteren bir televizyona benzetilebilir.

Bir bitmap'e gürültü efekti uygulamak için `noise()` yöntemini kullanın. Bu yöntem bir bitmap görüntünün belirli bir alanındaki piksellere rastgele bir renk değeri uygular.

Bu yöntem beş parametre kabul eder:

- `randomSeed` (int): Deseni belirleyen rastgele tohum sayısı. Adının içinde rastgele ifadesi geçmesine rağmen, aynı sayı iletildiğinde bu sayı aynı sonuçları oluşturur. Gerçekten rastgele bir sonuç elde etmek için `Math.random()` yöntemini kullanarak bu parametreye rastgele bir sayı iletin.
- `low` (uint): Bu parametre her bir piksel için üretilecek en düşük değeri belirtir (0-255). Varsayılan değer 0'dır. Bu değerin düşük olması daha koyu bir parazit deseni oluştururken daha yüksek bir değer verilmesi halinde daha parlak bir desen oluşacaktır.
- `high` (uint): Bu parametre her bir piksel için üretilecek en yüksek değeri belirtir (0-255). Varsayılan değer 255'tir. Bu değerin düşük olması daha koyu bir gürültü deseni oluştururken daha yüksek bir değer verilmesi halinde daha parlak bir desen oluşacaktır.
- `channelOptions` (uint): Bu parametre gürültünün bitmap nesnenin hangi kanalına uygulanacağını belirler. Bu sayı dört kanal ARGB değerlerinin her türlü kombinasyonu olabilir. Varsayılan değer 7'dir.
- `grayScale` (Boolean): `true` değeri verildiğinde, bu parametre bitmap piksellerine `randomSeed` değerini uygulayarak görüntünün tüm renklerini siler. Alfa kanalı bu parametreden etkilenmez. Varsayılan değer `false`'dır.

Aşağıdaki örnek bir bitmap görüntüsü oluşturur ve bu görüntüye mavi gürültü deseni uygular:

```
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);
        }
    }
}
```

Daha canlı görünen bir doku oluşturmak isterseniz `perlinNoise()` yöntemini kullanın. `perlinNoise()` yöntemi duman, bulut, su, ateş ve hatta patlamalar için ideal olan gerçekçi ve canlı doku oluşturur.

Bir algoritma ile üretildiğinden `perlinNoise()` yöntemi bitmap tabanlı dokulardan daha az bellek kullanır. Ancak bu yine de, özellikle de eski bilgisayarlarda içeriğinizi yavaşlatarak ve ekranın kare hızından daha yavaş şekilde yeniden çizilmesine sebep olarak işlemci kullanımını etkileyebilir. Bunun ana nedeni perlin gürültü algoritmalarını işlemek için gereken kayan nokta hesaplamalarıdır.

Bu yöntem dokuz parametre kabul eder (ilk altısı gereklidir):

- `baseX` (Number): Oluşturulan desenlerin x (boyut) değerini belirler.
- `baseY` (Number): Oluşturulan desenlerin y (boyut) değerini belirler.
- `numOctaves` (uint): Bu paraziti oluşturmak üzere birleştirilecek gürültü veya ayrı gürültü işlevleri sayısı. Oktav sayısının yüksek olması daha detaylı görüntülerin oluşturulmasını sağlasa da aynı zamanda daha fazla işleme süresi gerektirir.
- `randomSeed` (int): Rastgele tohum sayısı aynen `noise()` işlevinde olduğu şekilde çalışır. Gerçekten rastgele bir sonuç elde etmek için `Math.random()` yöntemini kullanarak bu parametreye rastgele bir sayı iletin.
- `stitch` (Boolean): Bu yöntem `true` değeri verildiğinde bitmap dolgusu olarak döşemek üzere eksiz dokular oluşturmak için görüntünün geçiş kenarlarını birleştirmeyi (veya düzgünleştirmeyi) dener.
- `fractalNoise` (Boolean): Bu parametre, yöntemin oluşturduğu degradelerin kenarları ile ilgilidir. `true` değeri verildiğinde, yöntem efektin kenarlarını düzgünleştiren parçalı gürültü oluşturur. `false` değeri verildiğinde, türbülans oluşturur. Türbülanslı bir görüntünün degradesinde görünebilir kesintiler vardır, bu da alev veya deniz dalgası gibi keskin görsel efektleri daha gerçekçi hale getirir.
- `channelOptions` (uint): `channelOptions` parametresi aynen `noise()` yönteminde olduğu şekilde çalışır. Gürültü deseninin uygulanacağı (bitmap) renk kanalını belirler. Bu sayı dört kanal ARGB değerlerinin her türlü kombinasyonu olabilir. Varsayılan değer 7'dir.
- `grayScale` (Boolean): `grayScale` parametresi aynen `noise()` yönteminde olduğu şekilde çalışır. `true` değeri verildiğinde bu parametre bitmap piksellerine `randomSeed` değerini uygulayarak görüntünün tüm renklerini siler. Varsayılan değer `false` şeklindedir.
- `offsets` (Dizi): Her bir oktavın x ve y uzaklıklarına karşılık gelen noktalar dizisi. Uzaklık değerlerini işleyerek, bir görüntünün katmanlarını sorunsuzca kaydırabilirsiniz. Uzaklık dizisindeki her nokta belirli bir oktav gürültü işlevini etkiler. Varsayılan değer `null` şeklindedir.

Aşağıdaki örnek, yeşil ve mavi bulut efekti oluşturmak için `perlinNoise()` yöntemini çağıran bir 150 x 150 piksel BitmapData nesnesi oluşturur.

```
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);
        }
    }
}
```

Bitmap'leri kaydırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcının haritayı her hareket ettirişinde (birkaç piksel hareket etse dahi) görünümü güncellemeniz gereken bir sokak haritası uygulaması oluşturduğunuzu düşünün.

Bu işlevi oluşturmanın bir yolu kullanıcının haritayı her hareket ettirişinde güncellenmiş haritayı içeren yeni bir görüntü oluşturmaktır. Ayrıca, tek bir büyük görüntü oluşturabilir ve `scroll()` yöntemini kullanabilirsiniz.

`scroll()` yöntemi ekrandaki bitmap'i kopyalar ve parametreler (`x`, `y`) ile belirlenen yeni bir uzak konuma yapıştırır. Bitmap'in bir kısmı sahne alanı dışında kalırsa görüntüye değişme efekti verir. Bir zamanlama işlevi (veya `enterFrame` olayı) ile birlikte kullanıldığında görüntünün hareket eder veya kayar şekilde görünmesini sağlayabilirsiniz.

Aşağıdaki örnek önceki perlin gürültüsü örneğini alır ve daha büyük bir bitmap görüntüsü oluşturur. (Bu görüntünün dörtte üçü sahne alanı dışında oluşturulur.) Ardından `scroll()` yöntemi görüntüyü çapraz şekilde aşağıya doğru uzaklaştıran bir `enterFrame` olay dinleyici ile birlikte uygulanır. Bu yöntem her kare girildiğinde çağrılır ve sonuç olarak görüntü aşağı doğru kaydıkça görüntünün ekran dışında kalan kısımları Sahne Alanında oluşturulur.

```
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);
}
```

Mipmap oluşturma avantajından yararlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

MIP eşlemler(*mipmap'ler* olarak da bilinir), çalışma zamanı oluşturma kalitesini ve performansını artırmak için gruplanmış ve bir doku ile ilişkilendirilmiş bitmap'lerdir. MIP eşlemindeki her bir bitmap görüntüsü, ana bitmap görüntüsünün bir sürümüdür ancak ana görüntüden küçük bir ayrıntı düzeyindedir.

Örneğin, en yüksek kalitede 64 × 64 piksel bir ana görüntü içeren MIP eşlemine sahip olabilirsiniz. MIP eşlemindeki daha düşük kaliteye sahip görüntüler şu şekilde olur: 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2 ve 1 × 1 piksel.

Doku akışı, öncelikle en düşük kalitedeki bitmap'i yükleme ve ardından bitmap'ler yüklenirken daha yüksek kalitedeki bitmap'leri aşamalı olarak görüntüleme yeteneğidir. Daha düşük kalitedeki bitmap'ler küçük olduğundan ana görüntüden daha hızlı yüklenirler. Bu nedenle, uygulama kullanıcıları ana, yüksek kalitedeki bitmap yüklenmeden önce bir uygulamada görüntü görüntüleyebilirler.

Flash Player 9.115.0 ve sonraki sürümleri ve AIR her bir bitmap'in farklı ölçeklerinin en iyileştirilmiş sürümlerini oluşturarak (%50'den başlayarak) bu teknolojiyi uygular. (Bu işleme *mipmap oluşturma* adı verilir.)

Flash Player 11.3 ve AIR 3.3, `Context3D.createCubeTexture()` ve `Context3D.createTexture()` yöntemlerinin `streamingLevels` parametresi aracılığıyla doku akışını destekler.

Doku sıkıştırma, doku görüntülerini doğrudan GPU üzerinde sıkıştırılmış biçimde saklamanıza izin verir. Bu, GPU belleğinden ve bellek bant genişliğinden tasarruf edilmesini sağlar. Genellikle, sıkıştırılmış dokular çevrimdışı olarak sıkıştırılır ve GPU'ya sıkıştırılmış biçimde yüklenir. Ancak, Flash Player 11.4 ve AIR 3.4 çalışma zamanı doku sıkıştırmasını destekler. Bu, vektör resminden dinamik dokular oluşturulması gibi durumlarda faydalıdır. Çalışma zamanı doku sıkıştırmasını kullanmak için aşağıdaki adımları uygulayın:

- `Context3D.createTexture()` yöntemini çağırıp üçüncü parametrede `flash.display3D.Context3DTextureFormat.COMPRESSED` veya `flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA` öğesini ileterek doku nesnesini oluşturun.

- `createTexture()` tarafından döndürülen `flash.display3D.textures.Texture` örneğini kullanarak `flash.display3D.textures.Texture.uploadFromBitmapData()` veya `flash.display3D.textures.Texture.uploadFromByteArray()` öğesini çağırın. Bu yöntemler dokuyu tek bir adımda yükler ve sıkıştırır.

MIP eşlemleri aşağıdaki bitmap türleri için oluşturulur:

- ActionScript 3.0 Loader sınıfını kullanarak görüntülenen bir bitmap (JPEG, GIF, veya PNG dosyaları)
- bir Flash Professional belgesinin kütüphanesindeki bir bitmap
- bir BitmapData nesnesi
- ActionScript 2.0 `loadMovie()` işlevini kullanarak görüntülenen bir bitmap

MIP eşlemleri filtrelenmiş nesnelere veya bitmap önbellekli film kliplerine uygulanmaz. Ancak, bitmap maskelenmiş içerik içinde olsa dahi filtrelenmiş bir görüntüleme nesnesi içinde bitmap dönüştürmeleriniz varsa MIP mapler uygulanır.

Mipmap oluşturma otomatik olarak gerçekleşir ancak görüntülerinizin bu en iyileştirmeden faydalandığından emin olmak için birkaç noktaya dikkat edebilirsiniz:

- Video oynatma için, Video nesnesinin `smoothing` özelliğine `true` değerini verin (bkz. Video sınıfı).
- Bitmap'ler için `smoothing` özelliğine `true` değerinin verilmesi gerekli değildir ancak kalite iyileştirmeleri bitmap'lerde düzgünleştirme kullandığında daha belirgindir.
- İki boyutlu görüntüler için 4'e veya 8'e bölünebilir bitmap boyutları (örn. 640 x 128 boyutlarındaki bir görüntü şu şekilde küçültülebilir: 320 x 64 > 160 x 32 > 80 x 16 > 40 x 8 > 20 x 4 > 10 x 2 > 5 x 1) kullanın.

  Üç boyutlu dokular için, her bir görüntünün 2'nin kuvveti (yani 2^n) olan bir çözünürlükte olduğu MIP eşlemleri kullanın. Örneğin, ana görüntü 1024 x 1024 piksel çözünürlüğündedir. MIP eşlemindeki daha düşük kaliteye sahip görüntüler bu durumda şu şekilde olur: MIP eşlemindeki toplam 11 görüntü için 512 x 512, 256 x 256, 128 x 128'den 1 x 1 piksele düşüş.

  Mipmap oluşturmanın tek sayılı genişlik veya yüksekliği olan bir bitmap içeriğinde meydana gelmediğini unutmayın.

Bitmap örneği: Animasyonlu dönen ay

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Animasyonlu dönen ay örneği Bitmap nesneler ve bitmap görüntü verileri (BitmapData nesneleri) ile çalışma tekniklerini gösterir. Bu örnek düz ay yüzeyi görüntüsünü ham görüntü verisi olarak kullanarak dönen küresel bir ay animasyonu oluşturur. Bu örnekte aşağıdaki teknikler gösterilmiştir:

- Harici bir görüntü yükleme ve ham görüntü verilerine erişme
- Kaynak görüntünün farklı kısımlarından sürekli pikseller kopyalayarak animasyon oluşturma
- Piksel değerlerini belirleyerek bir bitmap görüntü oluşturma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Animasyonlu dönen ay uygulama dosyaları Samples/SpinningMoon klasörü içinde bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
SpinningMoon.mxml veya SpinningMoon.fla	Flex (MXML) veya Flash (FLA) uygulamasındaki ana uygulama dosyası.
com/example/programmingas3/moon/MoonSphere.as	Ay öğesini yükleme, görüntüleme ve animasyon uygulama işlevlerini gerçekleştiren sınıf.
moonMap.png	Animasyonlu, dönen ayı oluşturmak için yüklenmiş ve kullanılmış olan ay yüzeyi fotoğrafı içeren görüntü dosyası.

Harici bir görüntüyü bitmap verisi olarak yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örneğin gerçekleştirdiği ilk ana görev harici bir görüntü dosyası olan ay yüzeyi fotoğrafını yüklemektir. Yükleme işlemi MoonSphere sınıfındaki iki yöntem tarafından işlenir: yükleme işleminin başlatıldığı `MoonSphere()` yapıcısı ve harici görüntü tamamen yüklendiğinde çağrılan `imageLoadComplete()` yöntemi.

Harici bir görüntü yüklemek harici bir SWF öğesi yüklemek gibidir; her ikisi de yükleme işlemini gerçekleştirmek için flash.display.Loader sınıfının birer örneğini kullanır. `MoonSphere()` yöntemindeki gerçek kod, görüntüyü aşağıdaki şekilde yüklemeye başlar:

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

İlk satır `imageLoader` adlı Loader örneğini bildirir. Üçüncü satır, Loader nesnesinin `load()` yöntemini çağırır ve yüklenecek görüntünün URL'sini temsil eden bir URLRequest örneği ileterek yükleme işlemini başlatır. İkinci satır, görüntü tamamen yüklendiğinde tetiklenecek olay dinleyiciyi ayarlar. `addEventListener()` yönteminin Loader örneğinin kendisinde değil Loader nesnesinin `contentLoaderInfo` özelliğinde çağrıldığına dikkat edin. Loader örneğinin kendisi yüklenen içerik ile ilgili olayları göndermez. Ancak `contentLoaderInfo` özelliği Loader nesnesine (bu örnekte harici görüntü) yüklenen içerik ile ilişkili olan LoaderInfo nesnesine bir başvuru içerir. LoaderInfo nesnesi görüntü tamamen yüklendiğinde `imageLoadComplete()` çağrısı tetikleyecek `complete`olayı (`Event.COMPLETE`) dahil harici içerik yükleme süreci ve tamamlanması ile ilgili birden çok olay sağlar.

Her ne kadar harici bir görüntü yüklemeye başlama işlemin önemli bir kısmını oluştursa da, yükleme bittiğinde ne yapılacağını bilmek de aynı derecede önemlidir. Yukarıdaki kodda gösterildiği üzere, görüntü yüklendiğinde `imageLoadComplete()` fonksiyonu çağrılır. Bu fonksiyon yüklenen görüntü verilerinde takip eden bölümlerde anlatılan birçok şeyi gerçekleştirir. Ancak, görüntü verilerini kullanması için bu verilere erişmesi gerekir. Bir Loader nesnesi harici bir görüntüyü yükleme için kullanıldığında, yüklenen görüntü Loader nesnesine alt görüntüleme nesnesi olarak eklenen bir Bitmap örneği haline gelir. Bu durumda, Loader örneği olay dinleyici yönteminde yönteme bir parametre olarak iletilen olay nesnesinin bir parçası olarak kullanılabilir. `imageLoadComplete()` yönteminin ilk satırları şu şekildedir:

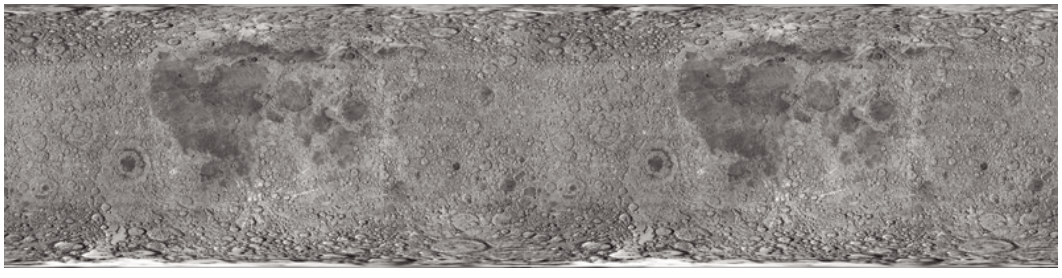
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

Olay nesnesi parametresinin adının `event` olduğuna ve Event sınıfının bir örneği olduğuna dikkat edin. Event sınıfının her örneğinin olayı tetikleyen nesneye başvuran bir `target` özelliği vardır (bu durumda, daha önce de belirtildiği gibi, `addEventListener()` yönteminin çağrıldığı LoaderInfo örneği). LoaderInfo nesnesinin ise yüklenen bitmap görüntü ile Bitmap örneğini içeren (yükleme işlemi tamamlandığında) bir `content` özelliği vardır. Görüntüyü doğrudan ekranda görüntülemek isterseniz, bu Bitmap örneğini (`event.target.content`) bir görüntüleme nesnesi kabına ekleyebilirsiniz. (Loader nesnesini bir görüntüleme nesnesi kabına da ekleyebilirdiniz.) Ancak, bu örnekte, yüklenen içerik ekranda görüntülenmek yerine bir ham görüntü kaynağı olarak kullanılmaktadır. Sonuç olarak, `imageLoadComplete()` yönteminin ilk satırı yüklenen Bitmap örneğinin (`event.target.content.bitmapData`) `bitmapData` özelliğini okur ve dönen ay animasyonunu oluşturmak için görüntü verilerinin bir kaynağı olarak kullanılan `textureMap` isimli örnek değişkeni içinde saklar. Sırada bu açıklanacak.

Piksel kopyalayarak animasyon oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Animasyon temel anlamında bir görüntünün belirli zaman aralıklarıyla değiştirilmesiyle oluşturulan hareket veya değişiklik yanılsamasını ifade eder. Bu örnekte amaç dikey ekseni etrafında dönen bir ay yanılsaması oluşturmaktır. Ancak, animasyonun amaçları doğrultusunda örneğin küresel deforme olma yönünü yok sayabilirsiniz. Yüklenen ve ay görüntüsü verilerinin kaynağı olarak kullanılan gerçek görüntüyü ele alın:



Gördüğünüz üzere, görüntü bir veya birkaç küreden değil dikdörtgen bir ay yüzeyi fotoğrafından oluşur. Fotoğraf tam olarak ayın ekvatorunda çekildiğinden, görüntünün alt ve üst kısımlarına yakın olan bölümleri uzamış ve bozulmuştur. Görüntüdeki deformasyonu düzeltmek ve yuvarlak şekilde görünmesini sağlamak için daha sonra anlatılacak bir yer değiştirme eşleme filtresi kullanacağız. Ancak, bu kaynak görüntünün dikdörtgen şeklinde olmasından dolayı, kürenin döndüğü izlenimini yaratmak için kodun ay yüzeyi fotoğrafını yatay olarak kaydırması gerekir.

Görüntünün aslında yan yana konulmuş iki ay yüzeyi fotoğrafından oluştuğuna dikkat edin. Görüntü, hareket görünümünü oluşturmak için sürekli görüntü verilerinin kopyalandığı kaynak görüntüdür. Görüntünün iki kopyası yan yana konularak sürekli, kesintisiz bir kaydırma efekti daha kolay elde edilebilir. Gelin bunun nasıl çalıştığını görmek için animasyon işlemini adım adım inceleyelim.

İşlem iki ayrı ActionScript nesnesi içerir. İlk olarak, kod içinde `textureMap` adlı BitmapData örneği ile temsil edilen yüklenmiş kaynak görüntü bulunur. Daha önceden anlatıldığı üzere, `textureMap` harici görüntü yüklenir yüklenmez şu kod kullanılarak görüntü verileri ile doldurulur:

```
textureMap = event.target.content.bitmapData;
```

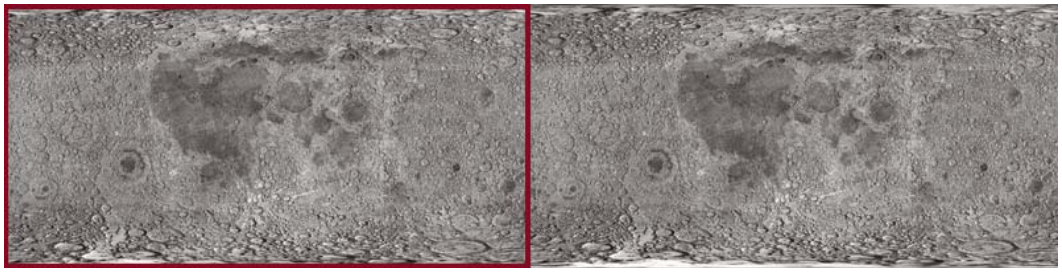
`textureMap`'in içeriği dikdörtgen şeklindeki ay görüntüsüdür. Buna ek olarak, dönüş animasyonunu oluşturmak için kod esasında ay görüntüsünü ekranda gösteren gerçek görüntüleme nesnesi olan `sphere` adlı bir Bitmap örneği kullanır. `textureMap` gibi `sphere` nesnesi de şu kod kullanılarak `imageLoadComplete()` içerisindeki ilk görüntü verileri ile oluşturulmuş ve doldurulmuştur:

```
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));
```

Kodun belirttiği üzere `sphere` oluşturulmuştur. `bitmapData` özelliği (`sphere` tarafından görüntülenen ham görüntü verileri) `textureMap` görüntüsüyle aynı yükseklikte ve yarısı kadar genişlikte oluşturulmuştur. Başka bir deyişle, `sphere` içeriği ay fotoğrafının boyutunda olacaktır (`textureMap` görüntüsü yan yana iki ay fotoğrafı içerdiği için). Ardından `bitmapData` özelliği `copyPixels()` yöntemi kullanılarak görüntü verileri ile doldurulmuştur. `copyPixels()` yöntemi çağrısındaki parametreler birkaç şeyi gösterir:

- İlk parametre görüntü verilerinin `textureMap` içinden kopyalandığını gösterir.
- Yeni bir Rectangle örneği olan ikinci parametre, anlık görüntünün `textureMap` görüntüsünün hangi kısmından alınacağını belirtir; bu örnekte anlık görüntü `textureMap` görüntüsünün sol üst köşesinden başlayan bir dikdörtgendir (ilk iki `Rectangle()` parametresi ile gösterilir: `0, 0`) ve dikdörtgen şeklindeki bu anlık görüntünün genişliği ve yüksekliği `sphere` öğesinin `width` ve `height` özellikleri ile aynıdır.
- `0` x ve y değerine sahip yeni bir Point örneği olan üçüncü parametre piksel verilerinin hedefini tanımlar - bu örnekte `sphere.bitmapData` örneğinin sol üst köşesidir (0,0).

Görsel olarak temsil edilen kod aşağıdaki görüntüde anahatları verilen `textureMap` görüntüsünden pikselleri kopyalayıp `sphere` öğesi üzerine yapıştırır. Başka bir deyişle, `sphere` öğesinin BitmapData içeriği burada vurgulanan `textureMap` görüntüsünün bir bölümüdür.



Ancak bunun sadece `sphere` öğesinin ilk durumu - `sphere` üzerine kopyalanan ilk görüntü içeriği olduğunu unutmayın.

Kaynak görüntü yüklendikten ve `sphere` oluşturulduktan sonra `imageLoadComplete()` yöntemi tarafından gerçekleştirilen son görev animasyon ayarlarını yapmaktır. Animasyon şu kod ile oluşturulan ve başlatılan `rotationTimer` adlı bir Timer örneği tarafından çalıştırılır.

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

Kod ilk olarak `rotationTimer` adlı Timer örneğini oluşturur; `Timer()` yapıcısına iletilen parametre `rotationTimer` öğesinin `timer` öğesini her 15 milisaniyede bir tetiklemesi gerektiğini belirtir. Ardından `timer` olayı (`TimerEvent.TIMER`) oluştuğunda `rotateMoon()` yönteminin çağrılacağını belirten `addEventListener()` yöntemi çağrılır. Son olarak `start()` yöntemi çağrılarak zamanlayıcı gerçek anlamda başlatılır.

`rotationTimer` öğesinin tanımlanma şeklinden dolayı Flash Player yaklaşık her 15 milisaniyede bir ay animasyonunun gerçekleştiği MoonSphere sınıfı içindeki `rotateMoon()` yöntemini çağırır. `rotateMoon()` yönteminin kaynak kodu şu şekildedir:

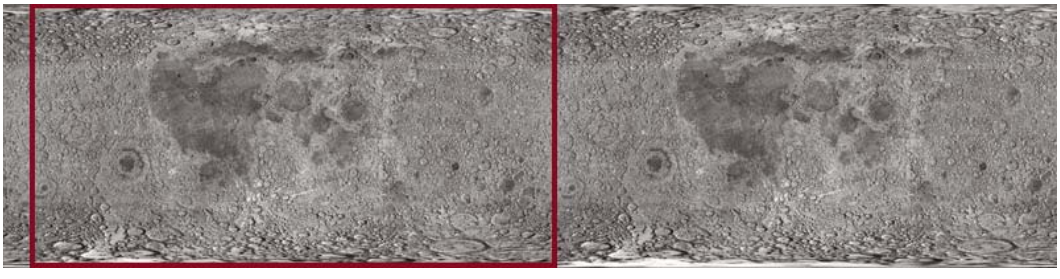
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                    new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                    new Point(0, 0));

    event.updateAfterEvent();
}
```

Bu kod üç şey gerçekleştirir:

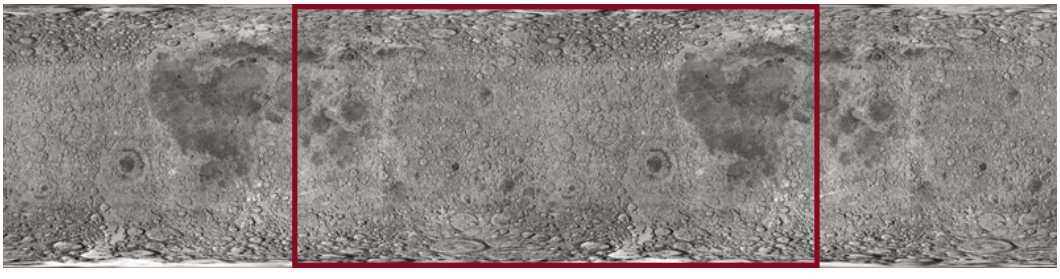
1 `sourceX` değişkeninin (başlangıçta 0 olan) değeri 1 artar.

```
sourceX += 1;
```

Göreceğiniz üzere, `sourceX`, `sphere` üzerine kopyalanacak piksellerin `textureMap` içerisindeki konumunu belirlemek için kullanılır, bu nedenle bu kod dikdörtgeni `textureMap` üzerinde bir piksel sağa taşıma efektine sahiptir. Görsel açıklamaya geri dönecek olursak, birkaç animasyon döngüsünden sonra kaynak dikdörtgen şu şekilde birkaç piksel sağa kaymış olacaktır:

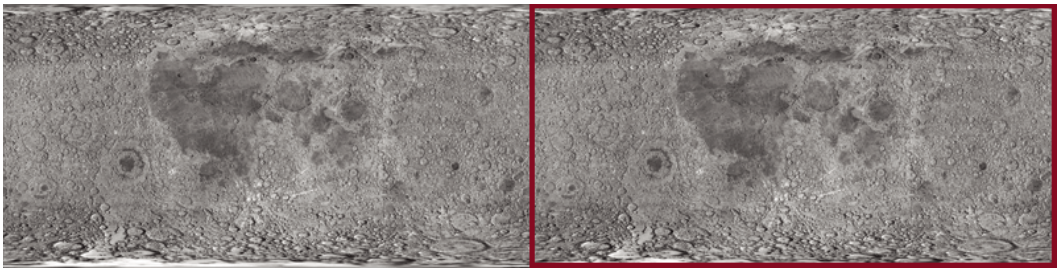


Birkaç döngünün ardından dikdörtgen daha da uzağa taşınacaktır:



Piksellerin kopyalandığı konumun yavaşça ve aynı doğrultuda değişmesi animasyonun temelini oluşturur. Kaynak konumu yavaşça ve sürekli olarak sağa kaydırıldığında ekrandaki `sphere` öğesi içinde görüntülenen görüntü sürekli olarak sola kayıyormuş gibi görünür. İşte bu nedenle kaynak görüntüde (`textureMap`) ay yüzeyi fotoğrafından iki tane olması gerekir. Dikdörtgen sürekli olarak sağa taşındığından genelde tek bir ay fotoğrafı üzerinde değil aynı anda iki ay fotoğrafı üzerindedir.

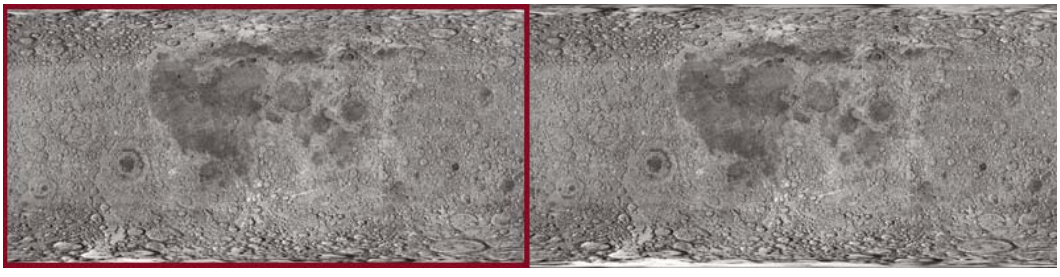
2 Kaynak dikdörtgenin yavaşça sağa doğru hareket etmesi konusunda bir sorun vardır. Dikdörtgen sonunda `textureMap` görüntüsünün sağ kenarına ulaşacak ve `sphere` öğesine kopyalanacak ay fotoğrafı pikseli kalmayacaktır:



Kodun bir sonraki satırı bu konu ile ilgilidir:

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

Kod `sourceX` öğesinin (dikdörtgenin sol kenarı) `textureMap` görüntüsünün ortasına gelip gelmediğini denetler. Geldiğinde `sourceX` değerini sıfırlayarak `textureMap` görüntüsünün sol kenarına geri götürür ve döngüyü baştan başlatır:



3 Uygun `sourceX` değerinin hesaplanmasının ardından animasyon oluşturmanın son adımı yeni kaynak dikdörtgen piksellerini `sphere` öğesi üzerine kopyalamaktır. Bu işlemi gerçekleştiren kod `sphere` öğesini başlangıçta dolduran koda çok benzer (önceki bölümlerde anlatılan); aralarındaki tek fark `new Rectangle()` yapıcı çağrısında dikdörtgenin sol kenarının `sourceX` konumunda olmasıdır:

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

Bu kodun her 15 milisaniyede bir çağrıldığını unutmayın. Kaynak dikdörtgenin konumu ve`sphere` öğesi üzerine kopyalanan pikseller sürekli olarak değiştiğinden ekranda `sphere` öğesinde görüntülenen fotoğraf görüntüsü sürekli olarak kayar. Başka bir deyişle, ay sürekli dönüyormuş gibi görünür.

Küresel görünümü oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bildiğiniz gibi ay bir dikdörtgen değil bir küredir. Bu nedenle, örneğin sürekli hareket eden dikdörtgen ay yüzeyi fotoğrafını alıp onu bir küreye çevirmesi gerekir. Bu işlem iki adımda gerçekleştirilir: ay yüzeyi fotoğrafının dairesel bir alanı haricindeki tüm içeriği gizlemek için bir maske kullanılır ve ay fotoğrafını üç boyutlu görünecek şekilde deforme etmek için bir yer değiştirme eşleme filtresi kullanılır.

Öncelikle, MoonSphere nesnesinin filtrenin oluşturduğu küre haricindeki içeriğinin tümünü gizlemek için daire şeklinde bir maske kullanılır. Aşağıdaki kod maskeyi bir Shape örneği olarak oluşturur ve MoonSphere örneğinin maskesi olarak uygular:

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

MoonSphere öğesi bir görüntüleme nesnesi olduğundan (Sprite sınıfına dayalıdır), maske miras `mask` özelliği kullanılarak doğrudan MoonSphere örneğine uygulanabilir.

Daire şeklinde bir maske kullanarak fotoğrafın bazı kısımlarını gizlemek gerçekçi bir dönen küre efekti oluşturmak için yeterli değildir. Ay yüzeyi fotoğrafının çekilme biçiminden dolayı boyutları orantılı değildir; görüntünün üst veya alt kısmına yakın olan kısımları ekvator kısmına göre daha fazla deforme olmuş ve uzamıştır. Ay fotoğrafının üç boyutlu görünmesini sağlamak amacıyla görünümünü deforme etmek için bir yer değiştirme eşleme filtresi kullanacağız.

Yer değiştirme eşleme filtresi bir görüntüyü deforme etmek için kullanılan bir filtredir. Bu durumda, ay fotoğrafı görüntünün üst ve alt kısmı yatay olarak sıkıştırılırken orta kısmı değişmeyecek şekilde "deforme edilecek" ve bu şekilde daha gerçekçi görülecektir. Filtrenin fotoğrafın kare şeklinde bir kısmında çalıştığını varsayarsanız, üstten ve alttan sıkıştırıp ortasını olduğu gibi bıraktığınızda kare, daireye dönüşecektir. Deforme edilmiş bu görüntüden animasyon oluşturmanın bir diğer etkisi de görüntünün ortasının üste ve alta yakın alanlardan daha fazla gerçek piksel mesafesi boyunca hareket ederek dairenin aslında üç boyutlu bir nesne (küre) olduğu izlenimini yaratmaktır.

Aşağıdaki kod `displaceFilter` adlı yer değiştirme eşleme filtresini oluşturmak için kullanılır.

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

İlk parametre olan `fisheyeLens` eşleme görüntüsü olarak bilinir; bu örnekte programlı şekilde oluşturulan bir BitmapData nesnesidir. Görüntünün oluşturulması "Piksel değerlerini belirleyerek bir bitmap görüntü oluşturma" sayfa 264 bölümünde anlatılmıştır. Diğer parametreler filtrenin, filtrelenmiş görüntü içinde uygulanacağı konumu, yer değiştirme efektini kontrol etmek için kullanılacak renk kanallarını ve bunların yer değiştirmeyi etkileme düzeyini belirtmektedir. Yer değiştirme eşleme filtresi oluşturulduktan sonra, halen `imageLoadComplete()` yöntemi içindeki `sphere` öğesine uygulanır.

```
sphere.filters = [displaceFilter];
```

Görüntünün maske ve yer değiştirme eşleme filtresi uygulandıktan sonraki hali şu şekildedir:

Dönen ay animasyonunun her döngüsünde kürenin BitmapData içeriği üzerine kaynak görüntü verilerinden yeni bir anlık görüntü yazılır. Ancak, filtrenin her seferinde yeniden uygulanması gerekli değildir. Bunun nedeni filtrenin bitmap verilerine (ham piksel bilgisi) değil Bitmap örneğine (görüntüleme nesnesi) uygulanmış olmasıdır. Bitmap örneğinin gerçek bitmap verileri olmadığını, bitmap verilerini ekranda görüntüleyen bir görüntüleme nesnesi olduğunu unutmayın. Benzetme yapmak gerekirse, bir Bitmap görüntüsü bir ekranda fotoğraf slaytları görüntülemek için kullanılan bir slayt projektörü gibidir, BitmapData nesnesi ise bir slayt projektörü ile sunulabilecek fotoğraf slaydının kendisidir. Görüntüyü değiştirmek amacıyla bir fotoğraf slaydının doğrudan üzerine çizim yapmak gibi bir BitmapData nesnesine doğrudan bir filtre uygulanabilir. Filtre ayrıca Bitmap örneği dahil herhangi bir görüntüleme nesnesine de uygulanabilir; bu durum ekranda görülen sonucu deforme etmek için slayt projektörün önüne bir filtre yerleştirmek gibi olacaktır. (Orijinal slayt değişmeyecektir.) Ham bitmap verilerine bir Bitmap örneğinin bitmapData özelliği ile erişilebildiğinden, filtre doğrudan ham bitmap verilerine de uygulanabilirdi. Ancak bu örnekte filtreyi bitmap verileri yerine Bitmap görüntüleme nesnesine uygulamak daha mantıklıdır.

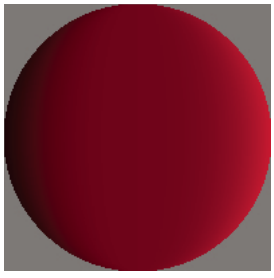
ActionScript ile yer değiştirme eşleme filtresini kullanma hakkında daha detaylı bilgi için "Görüntüleme nesnelerine filtre uygulama" sayfa 270 kısmını inceleyin.

Piksel değerlerini belirleyerek bir bitmap görüntü oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yer değiştirme eşleme filtresinin önemli yanlarından biri aslında iki görüntü içermesidir. Görüntülerden biri olan kaynak görüntü, filtre tarafından gerçekte değiştirilen görüntüdür. Bu örnekte kaynak görüntü `sphere` adlı Bitmap örneğidir. Filtre tarafından kullanılan diğer görüntü ise eşleme görüntüsü olarak bilinir. Eşleme görüntüsü gerçekte ekranda görüntülenmez. Bunun yerine piksellerinin renkleri yer değiştirme işlevi girdisi olarak kullanılır - eşleme görüntüsü içerisinde belirli bir x, y koordinatındaki pikselin rengi kaynak görüntüde aynı x, y koordinatındaki piksele uygulanacak yer değiştirme miktarını (fiziksel konum değişikliğini) belirler.

Bunun sonucunda, bir küre efekti oluşturmak amacıyla yer değiştirme eşleme filtresini kullanmak için, örnek uygun eşleme görüntüsüne ihtiyaç duyar - burada gösterildiği şekilde arka planı gri renk olan ve yatay olarak koyudan açığa doğru giden tek bir renk (kırmızı) degradesi ile dolu bir daire.



Bu örnekte sadece birer tane eşleme görüntüsü ve filtre kullanılmış olduğundan, eşleme görüntüsü `imageLoadComplete()` yönteminde sadece bir kere oluşturulur (başka bir deyişle, harici görüntünün yüklenmesi tamamlandıktan sonra). `fisheyeLens` adlı eşleme görüntüsü MoonSphere sınıfının `createFisheyeMap()` yöntemi çağrılarak oluşturulmuştur.

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

`createFisheyeMap()` yöntemi içerisinde, eşleme görüntüsü aslında BitmapData sınıfının `setPixel()` yöntemi kullanılarak piksel piksel çizilmiştir. `createFisheyeMap()` yönteminin tam kodu aşağıda verilmiş ve nasıl çalıştığı adım adım anlatılmıştır:

```
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;
}
```

İlk olarak, yöntem çağrıldığında oluşturulacak daire şeklindeki görüntünün yarıçapını gösteren `radius` parametresini alır. Ardından kod üzerine dairenin çizileceği BitmapData nesnesini oluşturur. `result` adlı bu nesne sonuçta yöntemin dönüş değeri olarak geri iletilir. Aşağıdaki kod parçacığında görüldüğü üzere, `result` BitmapData örneği dairenin çapı büyüklüğünde genişlik ve yükseklik değeri ile oluşturulur, saydamlığı yoktur (üçüncü parametre `false`) ve `0x808080` rengi (normal gri) ile doldurulmuştur:

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

Ardından kod, görüntünün her bir pikselini tekrarlamak için iki döngü kullanır. Dış döngü, görüntünün sütunlarını soldan sağa doğru tek tek işlerken (değiştirilmekte olan pikselin yatay konumunu temsil etmek için `i` değişkenini kullanır) iç döngü geçerli sütunun piksellerini yukarıdan aşağıya doğru tek tek işler. (`j` değişkeni geçerli pikselin dikey konumunu temsil eder.) Döngü kodları (iç döngünün içeriği hariç) aşağıda verilmiştir:

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

Döngüler pikselleri tek tek işlediğinden, her pikselde bir değer (o pikselin eşleme görüntüsündeki renk değeri) hesaplanır. İşlem dört adımdan oluşur:

1. Kod geçerli pikselin dairenin merkezine olan uzaklığını x ekseni (`i - radius`) üzerinden hesaplar. Bu değer mutlak mesafe yerine yarıçapı yüzdesi elde etmek için yarıçapa bölünür (`(i - radius) / radius`). Yüzde değeri `pctX` adlı değişken içinde saklanır ve y ekseni için eşdeğer değer hesaplanarak aşağıdaki kodda gösterildiği şekilde `pctY` değişkeni içinde saklanır.

   ```
   var pctX:Number = (i - radius) / radius;
   var pctY:Number = (j - radius) / radius;
   ```

2 Standart bir trigonometrik formül olan Pisagor teoremi kullanılarak dairenin merkezi ve geçerli nokta arasındaki doğrusal mesafe `pctX` ve `pctY` değerleri ile hesaplanır. Bu değer aşağıda gösterildiği şekilde `pctDistance` adlı bir değişkende saklanır:

```
var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
```

3 Ardından kod mesafe yüzdesinin 1'den daha düşük olup olmadığını (yani yarıçapın %100'ü olup olmadığını, bir başka deyişle işlenen pikselin dairenin merkezinde olup olmadığını) kontrol eder. Piksel dairenin içerisindeyse, hesaplanan bir renk değeri atanır (burada atlanmıştır ancak 4. adımda anlatılmıştır); değilse, bu piksele herhangi bir işlem yapılmaz ve varsayılan normal gri renkte bırakılır.

```
if (pctDistance < 1)
{
    ...
}
```

4 Daire içerisinde kalan piksellerin her biri için bir renk değeri hesaplanır. Son renk dairenin sol kenarındaki siyahtan (%0 kırmızı) dairenin sağ kenarındaki parlak (%100) kırmızıya doğru giden bir kırmızı tonu olacaktır. Bu renk değeri başlangıçta aşağıda gösterildiği şekilde üç kısımda hesaplanır (kırmızı, yeşil ve mavi):

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

Rengin sadece kırmızı kısmının (`red` değişkeni) bir değeri olduğuna dikkat edin. Yeşil ve mavi değerleri (`green` ve `blue` değişkenleri) burada görülmesi için kullanılmıştır ancak kullanılmayabilir. Bu yöntemin amacı kırmızı degrade dolgulu bir daire oluşturmak olduğundan yeşil veya mavi değerine gerek yoktur.

Üç renk değeri teker teker belirlendikten sonra standart bit değiştirme algoritması kullanılarak aşağıda gösterildiği şekilde tek bir tam sayı renk değeri olacak şekilde birleştirilir:

```
rgb = (red << 16 | green << 8 | blue);
```

Son olarak, renk değeri hesaplandıktan sonra aşağıdaki şekilde `result` BitmapData nesnesinin `setPixel()` yöntemi kullanılarak geçerli piksele atanır:

```
result.setPixel(i, j, rgb);
```

Bitmap görüntülerin senkronize olmayan kod çözme işlemi

Flash Player 11 ve üstü, Adobe AIR 2.6 ve üstü

Bitmap görüntüleriyle çalışırken uygulamanızın algılanan performansını arttırmak için bitmap görüntülerini senkronize olmayan kod çözme işleminden geçirip yükleyebilirsiniz. Çoğu kez bitmap görüntüsünün senkronize olmayan kod çözme işlemi, senkronize kod çözme işlemiyle aynı süreyi alabilir. Ancak bitmap görüntüsü, ilgili `Loader` nesnesi `COMPLETE` olayını göndermeden önce, ayrı iş parçacıkları içine kodu çözülmüş olarak girerler. Bu nedenle daha büyük görüntüleri yükledikten sonra senkronize olmayan bir şekilde kod çözme işleminden geçirebilirsiniz.

`flash.system` paketindeki `ImageDecodingPolicy` sınıfı, bitmap yükleme şemasını belirlemenizi sağlar. Varsayılan yükleme şeması senkronizedir.

Bitmap Kod Çözme İlkesi	Bitmap Yükleme Şeması	Açıklama
`ImageDecodingPolicy.ON_DEMAND`	Eşzamanlı	Görüntü verisine erişildiğinde yüklenen görüntülerin kodu çözülür. Daha küçük görüntülerin kod çözme işlemi için bu ilkeyi kullanın. Bu ilkeyi ayrıca uygulamanız karmaşık efektlere ve geçişlere dayanmadığı zaman kullanabilirsiniz.
`ImageDecodingPolicy.ON_LOAD`	Senkronize olmayan	Yüklenen görüntülerin kodu, `COMPLETE` olayı gönderilmeden önce yükleme sırasında çözülür. Daha büyük görüntüler için idealdir (10MP'den daha büyük). Sayfa geçişleri olan AIR temelli mobil uygulamalar geliştirirken, uygulamanızın algılanan performansını geliştirmek için bu bitmap yükleme ilkesini kullanın.

Not: *Yüklenen dosya bitmap görüntüsüyse ve kullanılan kod çözme ilkesi `ON_LOAD` ise görüntü, `COMPLETE` olayı gönderilmeden önce senkronize olmayan kod çözme işleminden geçirilir.*

Aşağıdaki kodlar `ImageDecodingPolicy` sınıfının kullanımını göstermektedir:

```
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);
```

Buna rağmen `Loader.load()` ve `Loader.loadBytes()` yöntemleriyle `ON_DEMAND` kod çözme işlemini kullanabilirsiniz. Ancak, argüman olarak `LoaderContext` nesnesini alan diğer tüm yöntemler, iletilen `ImageDecodingPolicy` değerini yok sayar.

Aşağıdaki örnek bir bitmap görüntüsünün senkronize ve senkronize olmayan kod çözme işlemleri arasındaki farkı göstermektedir:

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

Farklı kod çizme ilkelerinin etkilerinin gösterimi için bkz. Thibaud Imbert: Asynchronous bitmap decoding in the Adobe Flash runtimes (Adobe Flash Çalışma Zamanlarında Senkronize Olmayan Bitmap Kodu Çözme İşlemi)

Bölüm 14: Görüntüleme nesnelerine filtre uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Eskiden, bitmap görüntülerine filtre efektleri uygulanması, Adobe Photoshop® ve Adobe Fireworks® gibi özel görüntü düzenleme yazılımlarının uygulama alanına giriyordu. ActionScript 3.0 bir bitmap efekt filtresi sınıfları serisi içeren flash.filters paketini içerir. Bu efektler geliştiricilerin bitmap ve görüntüleme nesnelerine filtreler uygulamasına ve grafik değiştirme uygulamalarında kullanılabilir olan aynı efektlerin bir çoğunu başarmasına olanak verir.

Görüntüleme nesnelerine filtre uygulama temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir uygulamaya lehçe eklemenin yollarından biri ona basit grafik efektleri eklemektir. 3 boyut yanılsaması oluşturmak için bir fotoğrafın arkasına alt gölge, veya bir düğmenin aktif olduğunu göstermek için etrafına ışıma ekleyebilirsiniz. ActionScript 3.0, herhangi bir görüntüleme nesnesine veya BitmapData örneğine uygulayabileceğiniz on filtre içerir. Alt gölge ve ışıma filtreleri gibi basit filtrelerden, öteleme eşleme filtresi ve evrişim filtresi gibi karmaşık olanlara kadar bir çok yerleşik filtre vardır.

Not: *Yerleşik filtrelere ek olarak, Pixel Bender kullanarak özel filtreler de programlayabilirsiniz. Bkz. "Pixel Bender gölgelendiricileriyle çalışma" sayfa 302*

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi filtre oluştururken karşılaşabileceğiniz önemli terimleri içerir:

Eğim İki taraftaki pikselleri açıklaştırarak ve ters iki taraftaki pikselleri karartarak oluşturulan bir kenar. Bu efekt bir üç boyutlu kenarlık görünümünü oluşturur. Bu efekt basılmış veya basılmamış düğmeler ve benzer grafikler için sıkça kullanılır.

Evrişim Çeşitli oranlar kullanarak her pikselin değerini yakınındaki piksellerin bir kısmının veya tamamının değerleriyle birleştirip pikselleri deforme etme.

Öteleme Bir görüntüdeki pikselleri yeni bir konuma kaydırma veya taşıma.

Matris Izgaradaki sayıları çeşitli değerlere uygulayıp sonra da sonuçları birleştirerek belirli matematiksel hesapları gerçekleştirmek için kullanılan bir sayı ızgarası.

Daha fazla Yardım konusu

flash.filters package

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

Filtreler oluşturma ve uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Filtreler, bitmap ve görüntüleme nesnelerine, gölge, eğim ve bulanıklaştırma gibi çeşitli efektler uygulamanıza olanak sağlar. Her filtre bir sınıf olarak tanımlanır, bu nedenle filtre uygulanması kapsamında filtre nesnelerinin örnekleri oluşturulur; bu işlem herhangi bir nesne oluşturulmasından farklı değildir. Filtre nesnesinin bir örneği oluşturulduktan sonra, nesnenin `filters` özelliği kullanılarak veya BitmapData nesnesi olması durumunda `applyFilter()` yöntemi kullanılarak bu kolayca bir görüntüleme nesnesine uygulanabilir.

Filtre oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir filtre nesnesi oluşturmak için, seçtiğiniz filtre sınıfınızın yapıcı yöntemini çağırmanız yeterlidir. Örneğin, bir DropShadowFilter nesnesi oluşturmak için şu kodu kullanın:

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

Burada gösterilmese de, `DropShadowFilter()` yapıcısı (filtre sınıfının diğer tüm yapıcıları gibi), filtre efektinin görünümünü özelleştirmek için kullanılabilen birçok isteğe bağlı parametreyi kabul eder.

Filtre uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir filtre nesnesi oluşturduktan sonra, bunu bir görüntüleme nesnesine veya BitmapData nesnesine uygulayabilirsiniz; filtreyi uygulama şekliniz, filtreyi uyguladığınız nesneye bağlıdır.

Görüntüleme nesnesine filtre uygulama

Bir görüntüleme nesnesine filtre efektleri uyguladığınızda, bunları `filters` özelliği üzerinden uygularsınız. Görüntüleme nesnesinin `filters` özelliği, özellikleri, görüntüleme nesnesine uygulanan filtre nesneleri olan bir Array örneğidir. Bir görüntüleme nesnesine tek bir filtre uygulamak için, filtre örneğini oluşturun, bir Array örneğine ekleyin ve bu Array nesnesini görüntüleme nesnesinin `filters` özelliğine atayın:

```
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;
```

Nesneye birden çok filtre atamak istiyorsanız, Array örneğini `filters` özelliğine atamadan önce filtrelerin tümünü Array örneğine ekleyin. Nesneleri Array öğesinin yapıcısına parametreler olarak ileterek birden çok nesneyi Array öğesine ekleyebilirsiniz. Örneğin, bu kod, bir önceki oluşturulan görüntüleme nesnesine bir eğim filtresi ve ışıma filtresi uygular:

```
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;
```

Filtreleri içeren dizi oluştururken, `new Array()` yapıcısını kullanarak (önceki örneklerde gösterildiği gibi) diziyi oluşturabilir veya Array değişmez sözdizimini kullanarak filtreleri köşeli ayraçlar (`[]`) içine sarabilirsiniz. Örneğin, bu kod satırı:

```
var filters:Array = new Array(dropShadow, blur);
```

bu kod satırıyla aynı şeyi yapar:

```
var filters:Array = [dropShadow, blur];
```

Görüntüleme nesnelerine birden çok filtre uygularsanız, bunlar birikimli olarak art arda uygulanır. Örneğin, bir filtreler dizisi iki öğe içeriyorsa ve ilk olarak bir eğim filtresi eklenip ikinci olarak bir gölge filtresi eklenirse, gölge filtresi hem eğim filtresine hem de görüntüleme nesnesine uygulanır. Bu, gölge filtresinin filtreler dizisindeki ikinci konumundan kaynaklanır. Filtreleri birikimli şekilde uygulamak istemiyorsanız, her filtreyi, görüntüleme nesnesinin yeni bir kopyasına uygulayın.

Bir görüntüleme nesnesine yalnızca bir filtre veya birkaç filtre atıyorsanız, tek bir deyimde filtre örneğini oluşturabilir ve nesneye atayabilirsiniz. Örneğin, şu kod satırı, `myDisplayObject` adındaki bir görüntüleme nesnesine bulanıklaştırma filtresi uygular:

```
myDisplayObject.filters = [new BlurFilter()];
```

Önceki kod, Array değişmez sözdizimini (köşeli ayraçlar) kullanarak bir Array örneği oluşturur, Array öğesinde bir öğe olarak yeni bir BlurFilter örneği oluşturur ve bu Array öğesini `myDisplayObject` adındaki görüntüleme nesnesinin `filters` özelliğine atar.

Görüntüleme nesnesinden filtreleri kaldırma

Bir görüntüleme nesnesindeki tüm filtrelerin kaldırılması, `filters` özelliğine null değeri atanması kadar kolaydır:

```
myDisplayObject.filters = null;
```

Bir nesneye birden çok filtre uyguladıysanız ve filtrelerden yalnızca birini kaldırmak istiyorsanız, `filters` özelliği dizisini değiştirmek için birçok adım uygulamanız gerekir. Daha fazla bilgi için, bkz. "Filtrelerle çalışırken karşılaşılabilecek sorunlar" sayfa 273.

BitmapData nesnesine filtre uygulama

Bir BitmapData nesnesine filtre uygulanması için, BitmapData nesnesinin `applyFilter()` yönteminin kullanılması gerekir:

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

`applyFilter()` yöntemi, kaynak BitmapData nesnesine bir filtre uygulayarak yeni ve filtre uygulanmış bir görüntü oluşturur. Bu yöntem, orijinal kaynak görüntüyü değiştirmez; kaynak görüntüye uygulanan filtrenin sonucu, `applyFilter()` yönteminin çağrıldığı BitmapData örneğinde saklanır.

Filtreler nasıl çalışır

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme nesnesine filtre uygulama, orijinal nesnenin bir kopyasının saydam bitmap olarak önbelleğe alınmasıyla gerçekleşir.

Görüntüleme nesnesine filtre uygulandıktan sonra, nesne geçerli bir filtre listesine sahip olduğu sürece, çalışma zamanı nesneyi bitmap olarak önbelleğe alır. Ardından kaynak bitmap, sonradan uygulanan filtre efektlerinin tümü için orijinal görüntü olarak kullanılır.

Her görüntüleme nesnesi genellikle iki bitmap içerir: biri orijinal filtre uygulanmamış kaynak görüntüleme nesnesi ile ve diğeri filtre uygulamanın ardından elde edilen son görüntü için. Son görüntü, oluşturma sırasında kullanılır. Görüntüleme nesnesi değişmediği sürece, son görüntünün güncellenmesi gerekmez.

Filtrelerle çalışırken karşılaşılabilecek sorunlar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Filtrelerle çalışırken dikkate almanız gereken birkaç kafa karıştırıcı veya sorun çıkarıcı nokta vardır.

Filtreler ve bitmap önbelleğe alma

Görüntüleme nesnesine bir filtre uygulamak üzere, o nesne için bitmap önbelleğe almanın etkinleştirilmesi gerekir. `cacheAsBitmap` özelliği `false` değerine ayarlanmış olan bir görüntüleme nesnesine bir filtre uyguladığınızda, nesnenin `cacheAsBitmap` özelliği otomatik olarak `true` olur. Daha sonra görüntüleme nesnesinden tüm filtreleri kaldırırsanız, `cacheAsBitmap` özelliği daha önce ayarlandığı değere döndürülür.

Çalışma zamanında filtreleri değiştirme

Bir görüntüleme nesnesi önceden kendisine uygulanmış bir veya birkaç filtreye sahipse, `filters` özellik dizisine ek filtreler ekleyerek veya bu özellik dizisinden filtreleri kaldırarak filtre kümesini değiştiremezsiniz. Bunun yerine, uygulanan filtre kümesine ekleme yapmak veya bu filtre kümesini değiştirmek için, ayrı bir dizi üzerinde kendi değişikliklerinizi yapmanız ve daha sonra bu diziyi, nesneye uygulanacak filtrelerin görüntüleme nesnesinin filters özelliğine atamanız gerekir. Bunu yapmanın en kolay yolu, `filters` özellik dizisini bir Array değişkenine okumak ve bu geçici dizi üzerinde değişikliklerinizi yapmaktır. Bu diziyi daha sonra görüntüleme nesnesinin `filters` özelliğine yeniden atayabilirsiniz. Daha karmaşık durumlarda, ayrı bir ana filtre dizisi tutmanız gerekebilir. Bu ana filtre dizisi üzerinde değişikliklerinizi yapar ve her değişiklikten sonra ana diziyi görüntüleme nesnesinin `filters` özelliğine yeniden atarsınız.

Ek filtre ekleme

Aşağıdaki kod, önceden kendisine bir veya birkaç filtre uygulanmış bir görüntüleme nesnesine ek bir filtre ekleme işlemini göstermektedir. İlk olarak, `myDisplayObject` adındaki görüntüleme nesnesine bir ışıma filtresi uygulanır; daha sonra görüntüleme nesnesi tıklatıldığında, `addFilters()` işlevi çağrılır. Bu işlevde, `myDisplayObject` öğesine iki ek filtre uygulanır:

```
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);
```

Filtreler kümesinden tek bir filtreyi kaldırma

Bir görüntüleme nesnesine birden çok filtre uygulanmışsa ve tek bir filtreyi kaldırırken diğer filtrelerin nesneye uygulanmaya devam etmesini istiyorsanız, filtreleri geçici bir diziye kopyalarsınız, o diziden istenmeyen filtreyi kaldırırsınız ve geçici diziyi görüntüleme nesnesinin `filters` özelliğine yeniden atarsınız. Herhangi bir diziden bir veya birkaç öğenin kaldırılmasına yönelik birkaç yöntem "Değerleri alma ve dizi öğelerini kaldırma" sayfa 31 bölümünde açıklanmaktadır.

En basit yol, nesnede en üstte bulunan filtrenin (nesneye en son uygulanan filtrenin) kaldırılmasıdır. Diziden filtreyi kaldırmak için Array sınıfının `pop()` yöntemini kullanırsınız:

```
// 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;
```

Aynı şekilde, en alttaki filtreyi (nesneye ilk uygulanan filtreyi) kaldırmak için de aynı kodu kullanırsınız, böylece `pop()` yönteminin yerine Array sınıfının `shift()` yöntemini getirirsiniz.

Filtreler dizisinin ortasından bir filtreyi kaldırmak için (dizide ikiden fazla filtre olduğu varsayılarak), `splice()` yöntemini kullanmanız gerekir. Kaldırmak istediğiniz filtrenin dizinini (dizideki konumunu) bilmeniz gerekir. Örneğin, aşağıdaki kod, görüntüleme nesnesinden ikinci filtreyi (dizin 1'deki filtreyi) kaldırır:

```
// 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;
```

Filtrenin dizinini belirleme

Filtrenin dizinini bilmeniz için, diziden hangi filtreyi kaldırmak istediğinizi bilmeniz gerekir. Kaldırılacak filtrenin dizinini bilmeniz (uygulamanın tasarlanma yolundan) veya hesaplamanız gerekir.

En iyisi, kaldırmak istediğiniz filtre her zaman filtre kümesinde aynı konumda olacak şekilde uygulamanızı tasarlamaktır. Örneğin, evrişim filtresi ve gölge filtresi uygulanmış (bu sırayla) tek bir görüntüleme nesneniz varsa ve gölge filtresini kaldırmak ancak evrişim filtresini tutmak istiyorsanız, filtre bilinen bir konumda (filtrenin en üstünde) olduğundan, hangi Array yönteminin kullanılacağını (bu durumda gölge filtresini kaldırmak için `Array.pop()`) önceden bilebilirsiniz.

Kaldırmak istediğiniz filtre her zaman belirli bir türde olup mutlaka filtre kümesinde aynı konumda değilse, hangisinin kaldırılacağını belirlemek için dizide her filtrenin veri türünü kontrol edebilirsiniz. Örneğin, aşağıdaki kod hangi filtre kümesinin ışıma filtresi olduğunu belirler ve bu filtreyi kümeden kaldırır.

```
// 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;
```

Daha karmaşık bir durumda (örneğin, kaldırılacak filtre çalışma zamanında seçilirse), en iyisi, ana filtre listesi görevi görecek ayrı ve kalıcı bir filtre dizisi kopyasının saklanmasıdır. Filtre kümesi üzerinde herhangi bir değişiklik yaptığınızda, ana listeyi değiştirin, sonra bu filtre dizisini, görüntüleme nesnesinin `filters` özelliği olarak uygulayın.

Örneğin, aşağıdaki kod listesinde, farklı görsel efektler oluşturmak için görüntü nesnesine birden çok evrişim filtresi uygulanır ve uygulamanın ilerleyen bir noktasında bu filtrelerden biri kaldırılırken diğerleri saklanır. Bu durumda kod, filtreler dizisinin ana kopyasını ve kaldırılacak filtreye başvuruyu saklar. Belirli bir filtrenin bulunup kaldırılması, önceki yaklaşıma benzer, tek farkı filtreler dizisinin geçici bir kopyasının yapılması yerine ana kopyanın işlenmesi ve sonra görüntüleme nesnesine uygulanmasıdır.

```
// 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;
}
```

Bu yaklaşımda (hangi filtrenin kaldırılacağını belirlemek için, saklanan filtre başvurusunu, filtreler dizisindeki öğelerle karşılaştırdığınızda), filtreler dizisinin ayrı bir kopyasını tutmanız *gerekir*—saklanan filtre başvurusunu, görüntüleme nesnesinin `filters` özelliğinden kopyalanmış geçici bir dizideki öğelerle karşılaştırırsanız, kod çalışmaz. Bunun nedeni, `filters` özelliğine bir dizi atadığınızda dahili olarak çalışma zamanının dizideki her filtre nesnesinin bir kopyasını oluşturmasıdır. Görüntüleme nesnesine bu kopyalar (orijinal nesneler değil) uygulanır ve `filters` özelliğini geçici bir diziye okuduğunuzda, geçici dizi, orijinal filtre nesnelerine başvuruları değil, kopyalanan filtre nesnelerine başvuruları içerir. Sonuç olarak, önceki örnekte, `filterToRemove` öğesinin dizinini geçici filtreler dizisindeki filtrelerle karşılaştırarak belirlemeyi denerseniz herhangi bir eşleşme bulunmaz.

Filtreler ve nesne dönüştürmeleri

Vuruşun algılanması (bir örneğin başka bir örnekle örtüşmesinin veya kesişmesinin belirlenmesi) amacıyla, görüntüleme nesnesinin sınırlama kutusu dikdörtgeninin dışında kalan filtre uygulanmış hiçbir bölge (örneğin, gölge), yüzeyin bir bölümü olarak değerlendirilmez. DisplayObject sınıfının vuruş algılama yöntemleri vektör tabanlı olduğundan, bitmap sonucu üzerinde vuruş algılaması gerçekleştiremezsiniz. Örneğin, bir düğme örneğine eğim filtresi uygularsanız, örneğin eğim uygulanmış kısmında vuruş algılama kullanılamaz.

Ölçekleme, döndürme ve eğme, filtreler tarafından desteklenmez; filtre uygulanmış görüntüleme nesnesi ölçeklenirse (`scaleX` ve `scaleY` %100 olmazsa), filtre efekti, örnekle ölçeklenmez. Başka bir deyişle, örneğin orijinal şekli döner, ölçeklenir veya eğrilir ancak filtre örnekle birlikte dönmez, ölçeklenmez veya eğrilmez.

Gerçekçi efektler oluşturmak için, filtre içeren bir örneğe animasyon uygulayabilirsiniz veya örnekleri yuvalayabilir ve bu efekti elde etmek üzere filtrelere animasyon uygulamak için BitmapData sınıfını kullanabilirsiniz.

Filtreler ve Bitmap nesneleri

Bir BitmapData nesnesine herhangi bir filtre uyguladığınızda, `cacheAsBitmap` özelliği otomatik olarak `true` değerine ayarlanır. Bu şekilde, orijinal nesneye değil, nesnenin kopyasına filtre gerçekten uygulanır.

Daha sonra bu kopya, olabildiğince en yakın pikselle ana görüntüye (orijinal nesnenin üzerine) yerleştirilir. Orijinal bitmap'in sınırları değişirse, filtre uygulanan kopya bitmap, uzatılmak veya deforme edilmek yerine, yeniden orijinalden oluşturulur.

Görüntüleme nesnesinin tüm filtrelerini temizlerseniz, `cacheAsBitmap` özelliği, filtre uygulanmadan önce bulunduğu değere sıfırlanır.

Kullanılabilir görüntü filtreleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, görüntüleme nesnelerine ve BitmapData nesnelerine uygulayabileceğiniz on filtre sınıfını içerir:

- Eğim filtresi (BevelFilter sınıfı)
- Bulanıklaştırma filtresi (BlurFilter sınıfı)
- Gölge filtresi (DropShadowFilter sınıfı)
- Işıma filtresi (GlowFilter sınıfı)
- Degrade eğim filtresi (GradientBevelFilter sınıfı)
- Degrade ışıma filtresi (GradientGlowFilter sınıfı)
- Renk matrisi filtresi (ColorMatrixFilter sınıfı)
- Evrişim filtresi (ConvolutionFilter sınıfı)
- Öteleme eşlemesi filtresi (DisplacementMapFilter sınıfı)
- Gölgelendirici filtresi (ShaderFilter sınıfı)

İlk altı filtre, belirli bir efekt oluşturmak için kullanılabilen ve kullanılabilir efektin birkaç özelleştirmesini de içeren basit filtrelerdir. Bu altı filtre, ActionScript kullanılarak uygulanabilir ve Filtreler paneli kullanılarak Flash Professional'da nesnelere de uygulanabilir. Sonuç olarak, ActionScript kullanarak filtreler uyguluyor olsanız da, Flash Professional'ınız varsa, istediğiniz efektin nasıl oluşturulacağını bulmak üzere farklı filtreleri ve ayarları hızlı şekilde denemek için görsel arabirimi kullanabilirsiniz.

Son dört filtre yalnızca ActionScript'te kullanılabilir. Renk matrisi filtresi, evrişim filtresi, öteleme eşlemesi filtresi ve gölgelendirici filtresi olarak adlandırılan bu filtreler, oluşturdukları efektlerin türlerinde çok daha esnektir. Tek bir efekt için en iyileştirilmek yerine güç ve esneklik sağlarlar. Örneğin, evrişim filtresi, farklı matris değerleri seçilerek, bulanıklaştırma, kabartma, keskinleştirme, renkli kenarları bulma, dönüştürme vb. efektler oluşturmak için kullanılabilir.

İster basit ister karmaşık olsun, filtrelerin her biri özellikleri kullanılarak özelleştirilebilir. Genellikle filtre özelliklerini ayarlamak için iki seçeneğiniz vardır. Tüm filtreler, parametre değerlerini filtre nesnesinin yapıcısına ileterek özellikler ayarlamanıza olanak sağlar. Alternatif olarak, parametreler ileterek filtre özelliklerini ayarlasanız da ayarlamasanız da, daha sonra filtre nesnesinin özelliklerinin değerlerini ayarlayarak filtreleri ayarlayabilirsiniz. Örnek kod listelerinin çoğu, örneğin takip edilmesini kolaylaştırmak için özellikleri doğrudan ayarlar. Ancak, genellikle filtre nesnesinin yapıcısında değerleri parametreler olarak ileterek daha az sayıda kod satırında aynı sonucu elde edebilirsiniz. Her filtrenin belirli özellikleri ve yapıcı parametreleriyle ilgili ayrıntılı bilgi için, Adobe Flash Platformu için ActionScript 3.0 Başvurusu bölümündeki flash.filters paketine yönelik listelere bakın.

Eğim filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

BevelFilter sınıfı filtrelenen nesneye bir 3D eğimli kenar eklemenize olanak verir. Bu filtre, nesnenizin sert köşe ve kenarlarının yontulmuş veya eğimli görünmesini sağlar.

BevelFilter sınıfının özellikleri, eğimin görünümünü özelleştirmenize olanak sağlar. Vurgu ve gölge renkleri, eğimli kenar bulanıklaştırma, eğim açıları ve eğimli kenar yerleştirme ayarlayabilir, ayrıca boşaltma efekti de oluşturabilirsiniz.

Aşağıdaki örnek, harici bir görüntü yükler ve buna bir eğim filtresi uygular.

```
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];
```

Bulanıklaştırma filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

BlurFilter sınıfı bir görüntüleme nesnesini ve içeriğini lekelendirir veya bulanıklaştırır. Bulanıklaştırma efektleri, bir nesne için odak dışında kalmış izlenimi vermek veya hareket bulanıklaştırmasında olduğu gibi hızlı hareketi simüle etmek için kullanışlıdır. Bulanıklaştırma efektinin `quality` özelliğini düşük bir değere ayarlayarak yumuşak şekilde odak dışında kalmış efekti verebilirsiniz. `quality` özelliğinin yüksek bir değere ayarlanması, Gauss bulanıklaştırmasına benzer şekilde yumuşak bir bulanıklaştırma efekti sağlar.

Aşağıdaki örnek, Graphics sınıfının `drawCircle()` yöntemini kullanarak bir daire nesnesi oluşturur ve bu daire nesnesine bir bulanıklaştırma filtresi uygular:

```
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];
```

Gölge filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Gölgeler, hedef nesnenin yukarısında ayrı bir ışık kaynağı varmış izlenimini verir. Çeşitli gölge efektleri oluşturmak için bu ışık kaynağının konumu ve yoğunluğu değiştirilebilir.

DropShadowFilter sınıfı bulanıklaştırma filtresinin algoritmasına benzeyen bir algoritma kullanır. En büyük fark, gölge filtresinin, farklı ışık kaynağı niteliklerini simüle etmek için değiştirebileceğiniz biraz daha az özellik (alfa, renk, uzaklık ve parlaklık) içermesidir.

Gölge filtresi ayrıca gölge stilinde iç ve dış gölge ve boşaltma (kesme olarak da bilinir) modu gibi özel dönüştürme seçenekleri uygulamanıza da olanak sağlar.

Aşağıdaki kod, kare bir kutu hareketli grafiği oluşturur ve buna bir gölge filtresi uygular:

```
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];
```

Işıma filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

GlowFilter sınıfı görüntüleme nesnelerine yumuşak bir ışıma oluşturmak için onların nesnenin altından bir ışık parlıyormuş gibi görünmesini sağlayan bir ışıklandırma efekti uygular.

Gölge filtresine benzer şekilde, ışıma filtresi de değişik efektler oluşturmak için ışık kaynağının mesafesinin, açısının ve renginin değiştirilmesine yönelik özellikler içerir. GlowFilter ayrıca iç veya dış ışıma ve boşaltma modu da dahil olmak üzere, ışıma stilinin değiştirilmesine yönelik birçok seçenek içerir.

Aşağıdaki kod, Sprite sınıfını kullanarak bir çarpı oluşturur ve buna bir ışıma filtresi ekler:

```
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];
```

Degrade eğim filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

GradientBevelFilter sınıfı görüntüleme nesnelerine veya BitmapData nesnelerine gelişmiş bir eğim efekti uygulamanıza olanak verir. Eğim üzerinde bir degrade rengin kullanılması, eğimin uzaysal derinliğini artırarak kenarlara daha gerçekçi bir 3B görünüm verir.

Aşağıdaki kod, Shape sınıfının `drawRect()` yöntemini kullanarak bir dikdörtgen nesnesi oluşturur ve bu nesneye bir degrade eğim filtresi uygular.

```
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);
```

Degrade ışıma filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

GradientBevelFilter sınıfı görüntüleme nesnelerine veya BitmapData nesnelerine gelişmiş bir ışıma efekti uygulamanıza olanak verir. Bu efekt, ışıma üzerinde daha fazla denetim elde etmenizi sağlar ve sonuç olarak daha gerçekçi bir ışıma efekti oluşturur. Ayrıca, degrade ışıma filtresi, bir nesnenin iç, dış veya üst kenarlarına bir degrade ışıma uygulamanıza olanak sağlar.

Aşağıdaki örnek, Sahne Alanı'na bir daire çizer ve buna bir degrade ışıma filtresi uygular. Fareyi sağa ve aşağı doğru hareket ettirdikçe, bulanıklaştırma miktarı sırayla yatay ve dikey yönlerde artar. Ayrıca, Sahne Alanı'nı her tıklattığınızda, bulanıklaştırmanın gücü de artar.

```
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);
```

Örnek: Temel filtreleri birleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki kod örneği, animasyon uygulanmış bir trafik ışığı simülasyonu oluşturmak üzere yinelenen eylemler oluşturmaya yönelik Timer öğesiyle birlikte birçok temel filtreyi kullanır.

```
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();
```

Renk matrisi filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ColorMatrixFilter sınıfı filtrelenen nesnenin renk ve alfa değerlerini değiştirmek için kullanılır. Bu, tek bir renk kanalındaki değerleri kullanıp bunları diğer kanallara uygulayarak, doygunluk değişiklikleri, ton dönüşü (paleti bir renk aralığından diğerine kaydırma), parlaklıktan alfaya değişiklikler ve başka renk işleme efektleri oluşturmanıza olanak sağlar.

Kavramsal olarak, filtre kaynak görüntüdeki piksellerde birer birer ilerler ve her pikseli kırmızı, yeşil, mavi ve alfa bileşenlerine ayırır. Daha sonra renk matrisinde sağlanan değerleri bu değerlerin her biriyle çarparak sonuçta elde edilen ve o piksel için ekranda görüntülenecek renk değerini belirlemek için sonuçları birbirine ekler. Filtrenin `matrix` özelliği, son rengin hesaplanmasında kullanılan ve 20 sayıdan oluşan bir dizidir. Renk değerlerini hesaplamada kullanılan özel algoritmayla ilgili ayrıntılar için ColorMatrixFilter sınıfının `matrix` özelliğini anlatan Flash Platformu için Adobe ActionScript 3.0 Başvurusu'ndaki girişe bakın.

Evrişim filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

BitmapData nesnelerine veya görüntüleme nesnelerine bulanıklaştırma, kenar algılama, keskinleştirme, kabartma ve eğim verme gibi çeşitli görüntüleme dönüştürmeleri uygulamak için ConvolutionFilter sınıfı kullanılabilir.

Evrişim filtresi kavramsal olarak kaynak görüntüdeki her pikselde birer birer ilerler ve pikselin ve çevreleyen piksellerin değerini kullanarak son rengi belirler. Sayısal değerler dizisi olarak belirtilen bir matris, yakındaki piksellerin her birinin değerinin sonuçta elde edilen değeri ne derece etkilediğini belirtir.

3 x 3 matris olarak bilinen ve yaygın olarak kullanılan matris türünü göz önünde bulundurun. Matris, dokuz değer içerir:

```
N   N   N
N   P   N
N   N   N
```

Belirli bir piksele evrişim filtresi uygulandığında, pikselin kendi renk değerinin yanı sıra (örnekte “P”), çevreleyen piksellerin değerlerine de (örnekte “N” etiketli) bakar. Ancak, matristeki değerleri ayarlayarak belirli piksellerin sonuçta elde edilen görüntüyü etkileme konusunda ne kadar önceliğe sahip olduğunu belirtirsiniz.

Örneğin, evrişim filtresi kullanılarak uygulanan matris, görüntüyü değiştirmeden olduğu gibi bırakır:

```
0   0   0
0   1   0
0   0   0
```

Görüntünün değişmemesinin nedeni, orijinal piksel değeri, son piksel renginin belirlenmesinde 1 göreceli kuvvetine sahipken, çevreleyen piksellerin değerinin 0 göreceli kuvvetine sahip olmasıdır, başka bir deyişle, çevreleyen pikseller son görüntüyü etkilemez.

Aynı şekilde, bu matris de görüntünün piksellerinin bir piksel sola kaydırılmasını sağlar:

```
0   0   0
0   0   1
0   0   0
```

Bu durumda, pikselin, son görüntü üzerinde o konumda görüntülenen son piksel değeri üzerinde herhangi bir etkisi olmadığına, pikselin son değerini belirlemek için yalnızca sağdaki pikselin değerinin kullanıldığına dikkat edin.

ActionScript'te, değerleri içeren bir Array örneği ile matristeki satır ve sütunların sayısını belirten iki özelliğin bir birleşimi olarak matrisi oluşturursunuz. Aşağıdaki örnek, bir görüntüyü yükler ve görüntünün yüklemesi sona erdiğinde önceki listede bulunan matrisi kullanarak görüntüye bir evrişim filtresi uygular:

```
// 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);
```

Bu kodda mutlak olmayan şey, matriste 1 veya 0 dışındaki değerlerin kullanılmasının oluşturduğu efekttir. Örneğin, sağ konumda 1 yerine 8 sayısının yer aldığı aynı matris de aynı eylemi gerçekleştirir. (Pikselleri sola kaydırır.) Ayrıca bu, görüntülerin renklerini etkileyerek bunları 8 kat büyütür. Bunun nedeni, matris değerleri ile orijinal piksel renklerinin çarpılıp değerler birlikte toplandıktan sonra filtrenin `divisor` özelliğinin değerine sonucun bölünmesiyle son piksel rengi değerlerinin hesaplanmasıdır. Örnek kodda `divisor` özelliğinin 1 değerine ayarlandığına dikkat edin. Genel bir kural olarak, renklerin parlaklığının orijinal görüntüdeki gibi aynı şekilde kalmasını istiyorsanız, böleni, matris değerlerinin toplamına eşit tutmanız gerekir. Böylece, değerlerin 8'e kadar toplandığı ve bölenin 1 olduğu bir matrisle, sonuçta elde edilen görüntü, orijinal görüntüden hemen hemen 8 kat daha parlak olur.

Bu matrisin efekti çok fazla fark edilmese de, çeşitli efektler oluşturmak için başka matris değerleri kullanılabilir. Aşağıda, 3 x 3 matrisi kullanan farklı efektler için birçok standart matris değeri kümesi verilmiştir:

- Temel bulanıklaştırma (bölen 5):

```
 0 1 0
 1 1 1
 0 1 0
```

- Keskinleştirme (bölen 1):

```
 0, -1, 0
-1, 5, -1
 0, -1, 0
```

- Kenar algılama (bölen 1):

```
 0, -1, 0
-1, 4, -1
 0, -1, 0
```

- Kabartma efekti (bölen 1):

```
-2, -1, 0
-1, 1, 1
 0, 1, 2
```

Bu efektlerin çoğunda bölenin 1 olmasına dikkat edin. Bunun nedeni, pozitif matris değerlerine eklenen negatif matris değerlerinin 1 sonucunu vermesidir (veya kenar algılama durumunda 0 sonucu verilir ancak `divisor` özelliğinin değeri 0 olamaz).

Öteleme eşlemesi filtresi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

DisplacementMapFilter sınıfı, yeni bir nesnede öteleme efekti oluşturmak için BitmapData nesnesindeki (öteleme eşlemesi görüntüsü olarak bilinir) piksel değerlerini kullanır. Öteleme eşlemesi görüntüsü genellikle gerçek görüntüleme nesnesinden veya filtrenin uygulandığı BitmapData örneğinden farklıdır. Öteleme efekti, filtre uygulanan görüntüdeki piksellerin ötelenmesi, başka bir deyişle, orijinal konumundan farklı bir yere kaydırılmasıyla oluşturulur. Bu filtre, kaydırılmış, çarpıtılmış veya lekelenmiş efekti oluşturmak için kullanılabilir.

Belirli bir piksele uygulanan ötelemenin konumu ve miktarı, öteleme eşlemesi görüntüsünün renk değeriyle belirlenir. Filtreyle çalışırken, eşleme görüntüsünü belirtmenin yanı sıra, eşleme görüntüsünden ötelemenin nasıl hesaplandığını denetlemek için şu değerleri de belirtirsiniz:

- Eşleme noktası: Filtre uygulanan görüntüdeki, öteleme filtresinin sol üst köşesinin uygulanacağı konum. Filtreyi yalnızca görüntünün bir bölümüne uygulamak istiyorsanız bunu kullanabilirsiniz.
- X bileşeni: Eşleme görüntüsünün hangi renk kanalı piksellerin x konumunu etkiler.
- Y bileşeni: Eşleme görüntüsünün hangi renk kanalı piksellerin y konumunu etkiler.
- X ölçeği: x ekseni ötelemesinin ne kadar güçlü olduğunu belirten bir çarpan değeri.
- Y ölçeği: y ekseni ötelemesinin ne kadar güçlü olduğunu belirten bir çarpan değeri.
- Filtre modu: Kaydırılan piksellerin oluşturduğu boşluklarda ne yapılacağını belirler. DisplacementMapFilterMode sınıfında sabitler olarak tanımlanan seçenekler şunlardır: orijinal pikselleri görüntüleme (`IGNORE` filtre modu), pikselleri görüntünün diğer tarafından doğru sarma (`WRAP` filtre modu, varsayılandır), en yakındaki kaydırılmış pikseli kullanma (`CLAMP` filtre modu) veya boşlukları bir renkle doldurma (`COLOR` filtre modu).

Öteleme eşlemesi filtresinin nasıl çalıştığını anlamak için, temel bir örneği dikkate alın. Aşağıdaki kodda, bir görüntü yüklenir ve yükleme sona erdiğinde görüntü Sahne Alanı'nda ortalanır, görüntüye bir öteleme eşlemesi filtresi uygulanarak görüntünün tamamındaki piksellerin yatay olarak sağa kayması sağlanır.

```
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);
```

Ötelemeyi tanımlamak için kullanılan özellikler şunlardır:

- Eşleme bitmap'i: Öteleme bitmap'i, kod tarafından oluşturulan yeni bir BitmapData örneğidir. Boyutları, yüklenen görüntünün boyutlarıyla eşleşir. (Böylece görüntünün tamamına öteleme uygulanır.) Düz kırmızı piksellerle doldurulur.
- Eşleme noktası: Bu değer 0, 0 noktasına ayarlanarak tekrar ötelemenin görüntünün tamamına uygulanması sağlanır.
- X bileşeni: Bu değer `BitmapDataChannel.RED` sabitine ayarlanır, başka bir deyişle, eşleme bitmap'inin kırmızı değeri, x ekseni boyunca piksellerin ne kadar öteleneceğini (ne kadar hareket edeceğini) belirler.
- X ölçeği: Bu değer 250 değerine ayarlanır. Tam öteleme miktarı (tamamen kırmızı olan eşleme görüntüsünden), görüntüyü yalnızca küçük bir miktar öteler (yaklaşık yarım piksel), bu nedenle, bu değer 1 değerine ayarlansaydı, görüntü yatay olarak yalnızca 0,5 piksel kaydırılırdı. Değer 250 olarak ayarlandığında görüntü yaklaşık 125 piksel kaydırılır.

Bu ayarlar, filtre uygulanan görüntünün piksellerinin 250 piksel sola kaydırılmasını sağlar. Kaydırma yönü (sol veya sağ) ve miktarı, eşleme görüntüsündeki piksellerin renk değerini esas alır. Kavramsal olarak, filtre uygulanmış görüntünün piksellerinde birer birer ilerler (en azından filtrenin uygulandığı bölgedeki piksellerde, bu durumda tüm piksellerde) ve her piksel için şunu yapar:

1. Eşleme görüntüsünde karşılık gelen pikseli bulur. Örneğin, filtre uygulanan görüntünün sol üst köşesindeki piksel için öteleme miktarını hesaplarken, eşleme görüntüsünün sol üst köşesindeki piksele bakar.
2. Eşleme pikselinde, belirtilen renk kanalının değerini belirler. Bu durumda, x bileşeni renk kanalı kırmızı kanalıdır, bu nedenle filtre söz konusu pikselde eşleme görüntüsünün kırmızı kanalının değerinin ne olduğuna bakar. Eşleme görüntüsü düz kırmızı olduğundan, pikselin kırmızı kanalı 0xFF veya 255 olur. Bu, öteleme değeri olarak kullanılır.
3. Öteleme değerini, "orta" değer (0 ile 255 aralığının ortasında yer alan 127) ile karşılaştırır. Öteleme değeri orta değerden düşükse, piksel pozitif yönde (x ötelemesi için sağa; y ötelemesi için aşağı) kaydırılır. Diğer yandan, öteleme değeri orta değerden yüksekse (bu örnekte olduğu gibi), piksel negatif yönde (x ötelemesi için sola; y ötelemesi için yukarı) kaydırılır. Daha net olmak gerekirse, filtre öteleme değerini 127'den çıkarır ve sonuç (pozitif veya negatif), uygulanan göreceli öteleme miktarıdır.
4. Son olarak, göreceli öteleme değerinin tam ötelemenin yüzde kaçını temsil ettiğini belirleyerek gerçek öteleme miktarını belirler. Bu durumda, tam kırmızı, %100 öteleme anlamına gelir. Daha sonra uygulanacak ötelemenin piksel sayısını belirlemek için bu yüzde, x ölçeği veya y ölçeği değeriyle çarpılır. Bu örnekte, 250 çarpanının %100'ü, öteleme miktarını belirler ve bu yaklaşık 125 piksel sola anlamına gelir.

y bileşeni ve y ölçeği için herhangi bir değer belirtilmediğinden, varsayılanlar (ötelemeye neden olmayan) kullanılmıştır—görüntünün dikey yönde kaydırılmamasının nedeni budur.

Örnekte, varsayılan filtre modu ayarı olan `WRAP` kullanılmıştır, böylece pikseller sola kaydırıldıkça, sağdaki boşluk, görüntünün sol kenarından kaydırılan piksellerle doldurulur. Farklı efektleri görmek için bu değerle denemeler yapabilirsiniz. Örneğin, öteleme özelliklerinin ayarlandığı kod kısmına (`loader.filters = [displacementMap]` satırından önce) aşağıdaki satırı eklerseniz, görüntünün Sahne Alanı boyunca lekelenmiş gibi görünmesi sağlanır:

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

Daha karmaşık bir örnek için, aşağıdaki liste, görüntüde büyüteç efekti oluşturmak üzere bir öteleme eşlemesi filtresi kullanır:

```
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);
```

Kod ilk önce iki degrade daire oluşturur ve daha sonra öteleme eşlemesi görüntüsünü oluşturmak için bu iki daire birleştirilir. Kırmızı daire, x ekseni ötelemesini (`xyFilter.componentX = BitmapDataChannel.RED`) ve mavi daire de y ekseni ötelemesini (`xyFilter.componentY = BitmapDataChannel.BLUE`) oluşturur. Öteleme eşlemesi görüntüsünün nasıl göründüğünü anlamanıza yardımcı olmak için, kod, ekranın en altına, eşleme görüntüsü görevi gören birleştirilmiş dairenin yanı sıra orijinal daireleri de ekler.

Daha sonra kod bir görüntü yükler ve fare hareket ettikçe, farenin altındaki görüntü kısmına öteleme filtresini uygular. Öteleme eşlemesi görüntüsü olarak kullanılan degrade daireler, ötelenen bölgenin işaretçiden uzağa yayılmasını sağlar. Öteleme eşlemesi görüntüsünün gri bölgelerinin herhangi bir ötelemeye neden olmadığına dikkat edin. `0x7F7F7F`, gri renktir. Bu gri gölgesinin mavi ve kırmızı kanalları, bu renk kanallarının orta gölgesiyle tam olarak eşleşir, bu nedenle eşleme görüntüsünün gri alanında herhangi bir öteleme olmaz. Aynı şekilde, dairenin de ortasında öteleme olmaz. Buradaki renk gri olmasa da, bu rengin mavi kanalı ve kırmızı kanalı, orta grinin mavi kanalı ve kırmızı kanalıyla aynıdır ve ötelemeye neden olan renkler mavi ve kırmızı olduğundan, burada herhangi bir öteleme gerçekleşmez.

Gölgelendirici filtresi

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

ShaderFilter sınıfı Pixel Bender gölgelendiricisi olarak tanımlanan bir özel filtre efekti kullanmanıza olanak verir. Filtre efekti, Pixel Bender gölgelendiricisi olarak yazıldığından, bu efekt tamamen özelleştirilebilir. Filtre uygulanan içerik, görüntü girdisi olarak gölgelendiriciye iletilir ve gölgelendirici işleminin sonucu filtre sonucu olur.

Not: *Gölgelendirici filtresi, Flash Player 10 ve Adobe AIR 1.5'ten itibaren ActionScript'te kullanılabilmektedir.*

Bir nesneye gölgelendirici filtresi uygulamak için, ilk önce kullandığınız Pixel Bender'ı temsil eden bir Shader örneği oluşturursunuz. Shader örneği oluşturma prosedürü hakkında ve girdi görüntüsünün ve parametre değerlerinin nasıl belirtildiği hakkında daha fazla bilgi için, bkz. "Pixel Bender gölgelendiricileriyle çalışma" sayfa 302.

Filtre olarak bir gölgelendirici kullanıldığında, dikkate alınması gereken üç önemli şey vardır:

- En az bir girdi görüntüsünü kabul etmek için gölgelendiricinin tanımlanması gerekir.
- Filtre uygulanan nesne (filtre uygulanan görüntüleme nesnesi veya BitmapData nesnesi), birinci girdi görüntüsü değeri olarak gölgelendiriciye iletilir. Bundan dolayı, birinci görüntü girdisi için elle bir değer belirtmeyin.
- Gölgelendirici birden çok girdi görüntüsünü tanımlarsa, ek girdilerin manuel olarak (başka bir deyişle, Shader örneğine ait olan herhangi bir ShaderInput örneğinin `input` özelliği ayarlanarak) belirtilmesi gerekir.

Gölgelendiriciniz için Shader nesnesine sahip olduktan sonra, bir ShaderFilter örneği oluşturursunuz. Bu, herhangi bir filtre gibi kullandığınız gerçek filtre nesnesidir. Shader nesnesini kullanan bir ShaderFilter oluşturmak için, bu listede gösterildiği gibi, `ShaderFilter()` yapıcısını çağırın ve bir argüman olarak Shader nesnesini iletin:

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

Gölgelendirici filtresi kullanmanın tam örneği için, bkz. "Gölgelendiriciyi filtre olarak kullanma" sayfa 320.

Görüntüleme nesnelerine filtre uygulama örneği: Filter Workbench

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Filter Workbench, görüntülere ve başka görsel içeriklere farklı filtreler uygulamak ve ActionScript'te aynı efekti oluşturmak için kullanılabilecek sonuç kodunu görmek için bir kullanıcı arabirimi sağlar. Filtrelerle deneme yapılmasına yönelik bir araç sağlamanın yanı sıra, bu uygulama şu teknikleri de gösterir:

- Çeşitli filtrelerin örneklerini oluşturma
- Bir görüntüleme nesnesine birden çok filtre uygulama

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Filter Workbench uygulama dosyaları, Samples/FilterWorkbench klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	Filtrelerin uygulandığı içeriği değiştirme ve içeriğe filtreler uygulama gibi uygulamanın ana işlevlerini sağlayan sınıf.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	Filtre fabrikası sınıflarının her biri tarafından uygulanan ortak yöntemleri tanımlayan arabirim. Bu arabirim, FilterWorkbenchController sınıfının, ayrı ayrı filtre fabrikası sınıflarıyla etkileşim kurmak için kullandığı ortak işlevleri tanımlar.
com/example/programmingas3/filterWorkbench/ klasöründe: BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	Her biri IFilterFactory arabirimini uygulayan sınıflar kümesi. Bu sınıfların her biri, tek bir filtre türü için değerler oluşturma ve ayarlama işlevlerini sağlar. Uygulamadaki filtre özelliği panelleri, FilterWorkbenchController sınıfının alıp görüntü içeriğine uyguladığı kendi belirli filtrelerinin örneklerini oluşturmak için bu fabrika sınıflarını kullanır.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	Uygulamadaki filtre değerlerini işlemek için kullanılan kullanıcı arabirimi panellerini tanımlayan sınıfların uyguladığı ortak yöntemleri tanımlayan arabirim.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	Sayısal renk değerini onaltılık bir String değerine dönüştürme yöntemini içeren yardımcı program sınıfı
com/example/programmingas3/filterWorkbench/GradientColor.as	GradientBevelFilter ve GradientGlowFilter öğesinde her bir renkle ilişkilendirilmiş üç değeri (renk, alfa ve oran) tek bir nesnede birleştirerek bir değer nesnesi görevi gören sınıf
Kullanıcı arabirimi (Flex)	
FilterWorkbench.mxml	Uygulamanın kullanıcı arabirimini tanımlayan ana dosya.
flexapp/FilterWorkbench.as	Ana uygulamanın kullanıcı arabirimi için işlevler sağlayan sınıf; bu sınıf, uygulama MXML dosyası için kod arkası sınıfı olarak kullanılır.
flexapp/filterPanels klasöründe: BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	Tek bir filtreye yönelik seçenekleri ayarlamak için kullanılan panellerin her biri için işlevler sağlayan MXML bileşenleri kümesi.

File	Açıklama
flexapp/ImageContainer.as	Ekranda yüklenen görüntü için kap görevi gören bir görüntüleme nesnesi
flexapp/controls/BGColorCellRenderer.as	DataGrid bileşeninde bir hücrenin arka plan rengini değiştirmek için kullanılan özel hücre oluşturucu
flexapp/controls/QualityComboBox.as	Birçok filtre panelinde Kalite ayarı için kullanılabilen birleşik kutuyu tanımlayan özel denetim.
flexapp/controls/TypeComboBox.as	Birçok filtre panelinde Tür ayarı için kullanılabilen birleşik kutuyu tanımlayan özel denetim.
Kullanıcı arabirimi (Flash)	
FilterWorkbench.fla	Uygulamanın kullanıcı arabirimini tanımlayan ana dosya.
flashapp/FilterWorkbench.as	Ana uygulamanın kullanıcı arabirimi için işlevler sağlayan sınıf; bu sınıf, uygulama FLA dosyası için belge sınıfı olarak kullanılır.
flashapp/filterPanels klasöründe: BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	Tek bir filtreye yönelik seçenekleri ayarlamak için kullanılan panellerin her biri için işlevler sağlayan sınıflar kümesi. Her bir sınıf için ayrıca ana uygulama FLA dosyasının kütüphanesinde, adı sınıfın adıyla eşleşen (örneğin, “BlurPanel” sembolü, BlurPanel.as dosyasında tanımlı sınıfa bağlıdır), ilişkilendirilmiş bir MovieClip sembolü vardır. Kullanıcı arabirimini oluşturan bileşenler, bu semboller içinde konumlandırılıp adlandırılır.
flashapp/ImageContainer.as	Ekranda yüklenen görüntü için kap görevi gören bir görüntüleme nesnesi
flashapp/BGColorCellRenderer.as	DataGrid bileşeninde bir hücrenin arka plan rengini değiştirmek için kullanılan özel hücre oluşturucu
flashapp/ButtonCellRenderer.as	DataGrid bileşeninde bir hücreye düğme bileşeni eklemek için kullanılan özel hücre oluşturucu
Filtre uygulanan görüntü içeriği	
com/example/programmingas3/filterWorkbench/ImageType.as	Bu sınıf, uygulamanın yüklenebileceği ve filtrelerin uygulanacağı tek bir görüntü dosyasının türünü ve URL'sini içeren bir değer nesnesi görevini görür. Bu sınıf ayrıca kullanılabilir gerçek görüntü dosyalarını temsil eden bir sabitler kümesini de içerir.
images/sampleAnimation.swf, images/sampleImage1.jpg, images/sampleImage2.jpg	Uygulamada filtrelerin uygulandığı görüntüler ve diğer görsel içerikler.

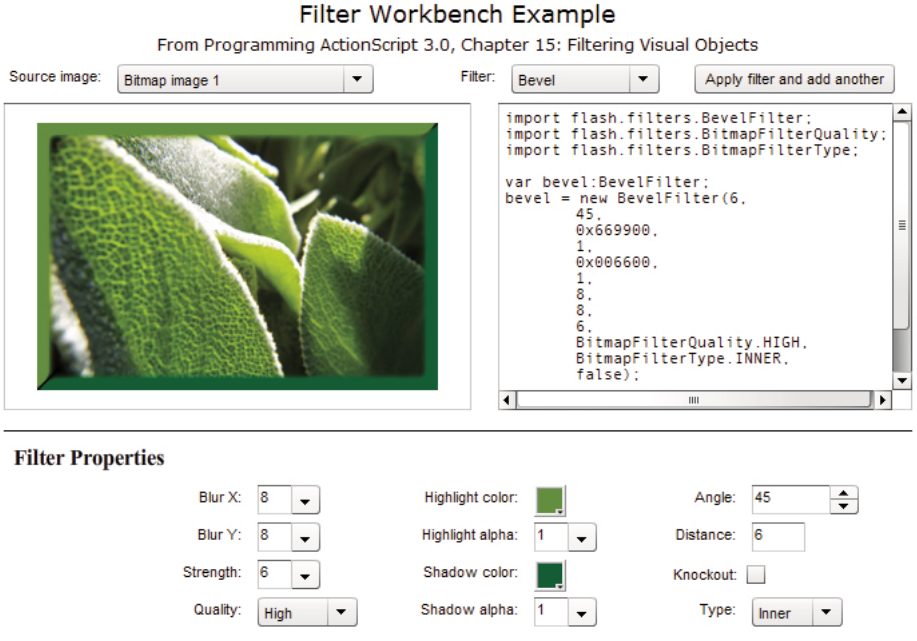
ActionScript filtreleriyle denemeler yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Filter Workbench uygulaması, çeşitli filtre efektleriyle deneme yapmanıza ve o efekt için uygun ActionScript kodunu oluşturmanıza yardımcı olmak üzere tasarlanmıştır. Bu uygulama, bitmap görüntüleri ve Flash ile oluşturulan animasyon gibi görsel içerikler barındıran üç farklı dosya arasından seçim yapmanıza ve seçilen görüntüye tek başına veya diğer filtrelerle birilikte sekiz farklı ActionScript filtresi uygulamanıza olanak sağlar. Uygulama şu filtreleri içerir:

- Eğim (flash.filters.BevelFilter)
- Bulanıklaştırma (flash.filters.BlurFilter)
- Renk matrisi (flash.filters.ColorMatrixFilter)
- Evrişim (flash.filters.ConvolutionFilter)
- Gölge (flash.filters.DropShadowFilter)
- Işıma (flash.filters.GlowFilter)
- Degrade eğim (flash.filters.GradientBevelFilter)
- Degrade ışıma (flash.filters.GradientGlowFilter)

Kullanıcı bir görüntüyü ve o görüntüye uygulanacak bir filtreyi seçtikten sonra, uygulama, seçilen filtrenin belirli özelliklerinin ayarlanmasına yönelik denetimlerin bulunduğu bir paneli görüntüler. Örneğin, aşağıdaki görüntüde, Eğim filtresi seçilmiş uygulama gösterilmektedir:



Kullanıcı filtre özelliklerini ayarladıkça, önizleme de gerçek zamanlı olarak güncellenir. Ayrıca kullanıcı tek bir filtreyi özelleştirip Uygula düğmesini tıklattıktan sonra başka bir filtreyi özelleştirip Uygula düğmesini tıklatarak birden çok filtre uygulayabilir.

Uygulamanın filtre panellerinde birkaç yeni özellik ve sınırlama vardır:

- Renk matrisi filtresi, parlaklık, kontrast, doygunluk ve ton gibi yaygın görüntü özelliklerinin doğrudan işlenmesine yönelik denetimler kümesi içerir. Ayrıca özel renk matrisi değerleri de belirtilebilir.
- Yalnızca ActionScript ile kullanılabilen evrişim filtresi, yaygın olarak kullanılan evrişim matrisi değerleri kümesini içerir veya özel değerler de belirtilebilir. Ancak, ConvolutionFilter sınıfı herhangi boyutta bir matrisi kabul ederken, Filter Workbench uygulaması en yaygın kullanılan filtre boyutu olan sabit 3 x 3 matrisini kullanır.
- Yalnızca ActionScript'te kullanılabilen öteleme eşleşmesi filtresi ve gölgelendirici filtresi, Filter Workbench uygulamasında kullanılamaz.

Filtre örnekleri oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Filter Workbench uygulaması, kullanılabilir filtreler için birer tane olmak üzere, filtreler oluşturmak için ayrı ayrı paneller tarafından kullanılan bir sınıflar kümesini içerir. Kullanıcı bir filtreyi seçtiğinde, filtre paneliyle ilişkilendirilmiş ActionScript kodu, uygun filtre fabrikası sınıfının bir örneğini oluşturur. (Gerçek fabrikaların ayrı ayrı ürünler oluşturması gibi, bu sınıfların amacı da diğer nesnelerin örneklerini oluşturmak olduğundan, bu sınıflar *fabrika sınıfları* olarak bilinir.

Kullanıcı paneldeki bir özellik değerini her değiştirdiğinde, panelin kodu fabrika sınıfındaki uygun yöntemi çağırır. Her fabrika sınıfı, uygun filtre örneğini oluşturmak için panelin kullandığı belirli yöntemleri içerir. Örneğin, kullanıcı Bulanıklaştırma filtresini seçerse, uygulama bir BlurFactory örneği oluşturur. BlurFactory sınıfı, istenen BlurFilter örneğini oluşturmak için birlikte kullanılan şu üç parametreyi kabul eden bir `modifyFilter()` yöntemini içerir: `blurX`, `blurY` ve `quality`:

```
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));
}
```

Diğer yandan, kullanıcı Evrişim filtresini seçerse, bu filtre daha fazla esneklik sağlar ve sonuç olarak denetleme için daha büyük bir özellikler kümesine sahiptir. ConvolutionFactory sınıfında, kullanıcı filtre panelinde farklı bir değeri seçtiğinde şu kod çağrılır:

```
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));
}
```

Her iki durumda da, filtre değerleri değiştirildiğinde fabrika nesnesinin, filtrenin değerlerinin değiştiğini dinleyicilere bildirmek için bir `Event.CHANGE` olayını gönderdiğine dikkat edin. Filtre uygulanmış içeriğe gerçekten filtreleri uygulama işini yapan FilterWorkbenchController sınıfı, ne zaman filtrenin yeni bir kopyasını alması ve ne zaman bunu filtre uygulanmış içeriğe yeniden uygulaması gerektiğinden emin olmak için bu olayı dinler.

FilterWorkbenchController sınıfının, filtre fabrikası sınıfının her birinin ayrıntılarını bilmesi gerekmez—filtrenin değişmiş olduğunu ve filtrenin kopyasına erişebildiğini bilmesi yeterlidir. Bunu desteklemek için uygulama, filtre fabrikası sınıfının sağlaması gereken davranışı tanımlayan IFilterFactory adında bir arabirim içerir, böylece uygulamanın FilterWorkbenchController örneği işini yapabilir. IFilterFactory, FilterWorkbenchController sınıfında kullanılan `getFilter()` yöntemini tanımlar:

```
function getFilter():BitmapFilter;
```

`getFilter()` arabirim yöntemi tanımının, belirli bir filtre türünü değil bir BitmapFilter örneğini döndürdüğünü belirtmesine dikkat edin. BitmapFilter sınıfı, belirli bir filtre türünü tanımlamaz. Bunu yerine, BitmapFilter öğesi, tüm filtre sınıflarının dayandığı temel sınıftır. Her filtre fabrikası sınıfı, dayandığı filtre nesnesine bir başvuru döndüren `getFilter()` yönteminin belirli bir uygulamasını tanımlar. Örneğin, aşağıda ConvolutionFactory sınıfının kaynak kodunun kısaltılmış bir sürümü yer almaktadır:

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

`getFilter()` öğesini çağıran tüm nesnelerin mutlaka bunu bilmesi gerekmese de, ConvolutionFactory sınıfının `getFilter()` yöntemi uygulamasında bu, ConvolutionFilter örneğini döndürür—ConvolutionFactory öğesinin takip ettiği `getFilter()` yönteminin tanımına göre bunun herhangi bir ActionScript filtresi sınıfının örneği olabilen bir BitmapFilter örneğini döndürmesi gerekir.

Görüntüleme nesnelerine filtreler uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Daha önce de açıklandığı gibi, Filter Workbench uygulaması, FilterWorkbenchController sınıfının bir örneğini (bundan sonra "denetleyici örneği" olarak ifade edilecektir) kullanır, bu örnek, seçilen görsel nesneye gerçek filtre uygulama görevini gerçekleştirir. Denetleyici örneğinin bir filtreyi uygulayabilmesi için öncelikle filtrenin hangi görüntüye veya görsel içeriğe uygulanması gerektiğini bilmesi gerekir. Kullanıcı bir görüntü seçtiğinde, uygulama, FilterWorkbenchController sınıfında `setFilterTarget()` yöntemini çağırarak ImageType sınıfında tanımlı sabitlerden birinde iletilir.

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

Denetleyici örneği, bu bilgileri kullanarak belirlenen dosyayı yükler ve yükleme tamamlandıktan sonra dosyayı `_currentTarget` adında bir örnek değişkeninde saklar:

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

Kullanıcı bir filtre seçtiğinde, uygulama, denetleyici örneğinin `setFilter()` yöntemini çağırarak denetleyici için `_filterFactory` adında bir örnek değişkeninde sakladığı ilgili filtre fabrikası nesnesine başvuru sağlar.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

Önceden açıklandığı gibi, denetleyici örneğinin verilen filtre fabrikası örneğinin belirli veri türünü bilmediğine; yalnızca IFilterFactory örneğini uygulayan nesneyi bildiğine, başka bir deyişle, `getFilter()` yöntemine sahip olup filtre değiştiğinde `change` (`Event.CHANGE`) olayı gönderdiğine dikkat edin.

Kullanıcı, filtrenin panelinde bir filtrenin özelliklerini değiştirdiğinde, denetleyici örneği, denetleyici örneğinin `filterChange()` yöntemini çağıran filtre fabrikasının `change` olayıyla bu filtrenin değiştirildiğini fark eder. Daha sonra bu yöntem `applyTemporaryFilter()` yöntemini çağırır:

```
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();
}
```

Görüntüleme nesnesine filtre uygulama işi, `applyTemporaryFilter()` yönteminin içinde gerçekleşir. İlk olarak denetleyici, filtre fabrikasının `getFilter()` yöntemini çağırarak filtre nesnesine bir başvuruyu alır.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

Denetleyici örneği, görüntüleme nesnesine uygulanan tüm filtreleri sakladığı, `_currentFilters` adında bir Array örneği değişkenine sahiptir. Sonraki adım, yeni güncellenen filtrenin bu diziye eklenmesidir:

```
_currentFilters.push(currentFilter);
```

Daha sonra kod, filtreleri gerçekten görüntüye uygulayan görüntüleme nesnesinin `filters` özelliğine filtreler dizisini atar:

```
_currentTarget.filters = _currentFilters;
```

Son olarak, bu en son eklenen filtre halen "çalışan" filtre olduğundan, bu filtrenin görüntüleme nesnesine kalıcı olarak uygulanmaması gerekir, bu nedenle de söz konusu filtre `_currentFilters` dizisinden kaldırılır:

```
_currentFilters.pop();
```

Görüntüleme nesnesi, `filters` özelliğine atandığında filtreler dizisinin bir kopyasını oluşturduğundan ve orijinal dizi yerine bu dahili diziyi kullandığından, bu filtrenin diziden kaldırılması, filtre uygulanan görüntüleme nesnesini etkilemez. Bu nedenle, dizi, görüntüleme nesnesinin `filters` özelliğine tekrar atanıncaya kadar, filtreler dizisi üzerinde yapılan değişiklikler görüntüleme nesnesini etkilemez.

Bölüm 15: Pixel Bender gölgelendiricileriyle çalışma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Adobe Pixel Bender Toolkit, geliştiricilerin grafiksel efektler oluşturan gölgelendiriciler yazmasına ve diğer görüntü ve veri işlemelerini gerçekleştirmesine olanak sağlar. Görüntü verisine veya görsel içeriğe efekti uygulamak için ActionScript'te Pixel Bender bayt kodu çalıştırılabilir. ActionScript'te Pixel Bender gölgelendiricilerinin kullanılması, özel görsel efektler oluşturma ve ActionScript'teki yerleşik yeteneklerin ötesinde veri işlemeleri gerçekleştirme yeteneğini size sağlar.

Not: *Flash Player 10 ve Adobe AIR 1.5'ten itibaren Pixel Bender desteği kullanılabilir. Pixel Bender karışımları, filtreleri ve dolguları GPU görüntü oluşturmada desteklenmez.*

Daha fazla Yardım konusu

Adobe Pixel Bender Teknoloji Merkezi

Pixel Bender Geliştirici Kılavuzu

Pixel Bender Başvurusu

flash.display.Shader

flash.filters.ShaderFilter

Flash için Pixel Bender temelleri

Flex için Pixel Bender temelleri

Pixel Bender gölgelendiricilerinin temelleri

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Adobe Pixel Bender, görüntü içeriğini oluşturmak veya işlemek için kullanılan bir programlama dilidir. Pixel Bender'ı kullanarak gölgelendirici olarak da adlandırılan bir çekirdek oluşturursunuz. Gölgelendirici, görüntü piksellerinin her birinde ayrı ayrı çalıştırılan tek bir işlevi tanımlar. İşleve yapılan her çağrının sonucu, görüntüde o piksel koordinatındaki çıktı rengidir. İşlemi özelleştirmek için girdi görüntüleri ve parametre değerleri belirtilebilir. Gölgelendiricinin tek bir çalıştırmasında girdi ve parametre değerleri sabittir. Değişen tek şey, rengi işlev çağrısının sonucu olan pikselin koordinatıdır.

Mümkün olduğunca, paralel olarak birden çok çıktı piksel koordinatı için gölgelendirici işlevi çağrılır. Bu da gölgelendirici performansını artırır ve yüksek performanslı işleme sağlar.

ActionScript'te bir gölgelendirici kullanılarak üç efekt türü kolayca oluşturulabilir.

- çizim dolgusu
- harmanlama modu
- filtre

Gölgelendirici, bağımsız modda da çalıştırılabilir. Bağımsız mod kullanılarak, tasarlanan kullanımı önceden belirtilmeden gölgelendiricinin sonucuna doğrudan erişilir. Görüntü verisi veya ikili veri ya da sayı verisi olarak sonuca erişilebilir. Verinin görüntü verisi olması gerekmez. Bu şekilde, gölgelendiriciye girdi olarak bir veri kümesi verebilirsiniz. Gölgelendirici verileri işler ve siz de gölgelendiricinin döndürdüğü sonuç verilerine erişebilirsiniz.

Flash Player 10 ve Adobe AIR 1.5'ten itibaren Pixel Bender desteği mevcuttur. Pixel Bender karışımları, filtreleri ve dolguları GPU işlemesi altında desteklenmez. Mobil aygıtlarda, Pixel Bender gölgelendiricileri işlemcinin görüntü oluşturma işlemi altında çalışmaz. Ancak, performans masaüstü bilgisayardakiyle aynı düzeyde değildir. Birçok gölgelendirici programı yalnızca saniye başına birkaç kare şeklinde yürütülür.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi Pixel Bender gölgelendiricileri kullanırken ve oluştururken karşılaşacağınız önemli terimleri içerir:

Çekirdek Pixel Bender için çekirdek, gölgelendirici ile aynı şeydir. Pixel Bender'ı kullanarak kodunuz, bir görüntünün piksellerinin her birinde ayrı ayrı çalışan tek bir işlevi tanımlayan bir çekirdeği tanımlar.

Pixel Bender bayt kodu Pixel Bender çekirdeği derlendiğinde, Pixel Bender bayt koduna dönüştürülür. Bayt kodu çalışma zamanında erişilir ve çalıştırılır.

Pixel Bender dili Pixel Bender çekirdeği oluşturmak için kullanılan programlama dili.

Pixel Bender Araç Kiti Pixel Bender kaynak kodundan bir Pixel Bender bayt kodu oluşturmak için kullanılan uygulama. Araç kiti, Pixel Bender kaynak kodunu yazmanıza, test etmenize ve derlemenize olanak sağlar.

Shader Bu belgenin amaçları doğrultusunda gölgelendirici, Pixel Bender dilinde yazılmış bir işlevler kümesidir. Gölgelendiricinin kodu bir görsel efekt oluşturur veya bir hesaplama gerçekleştirir. Her iki durumda da, gölgelendirici bir veri kümesi döndürür (genellikle görüntünün pikselleri). Gölgelendirici her bir veri noktasında aynı işlemi gerçekleştirir, tek fark çıktı pikselinin koordinatlarıdır. Gölgelendirici ActionScript'te yazılı değildir. Pixel Bender dilinde yazılır ve Pixel Bender bayt koduna derlenir. Derleme zamanında bir SWF dosyasına gömülebilir veya çalışma zamanında harici dosya olarak yüklenebilir. Her iki durumda da, Shader nesnesinin oluşturulmasıyla ve söz konusu nesnenin gölgelendirici bayt koduna bağlanmasıyla buna erişilebilir.

Gölgelendirici girdisi Hesaplamalarında kullanılmak üzere bir gölgelendiriciye sağlanan ve genellikle bitmap görüntü verisi olan karmaşık bir girdi. Gölgelendiricide tanımlanan her girdi değişkeni için gölgelendiricinin çalıştırmasının tamamına yönelik tek bir değer (başka bir deyişle, tek bir görüntü veya ikili veri kümesi) kullanılır.

Gölgelendirici parametresi Hesaplamalarında kullanılmak üzere bir gölgelendiriciye sağlanan tek bir değer (veya sınırlı değerler kümesi). Her parametre değeri tek bir gölgelendirici çalıştırması için tanımlanır ve gölgelendirici çalıştırması boyunca aynı değer kullanılır.

Kod örnekleriyle çalışma

Sağlanan örnek kod listelerini test etmek isteyebilirsiniz. Kodu test etmek kodu çalıştırıp oluşan SWF'deki sonuçları incelemeyi içerir. Tüm örnekler, gölgelendirici efektini kullanan veya gölgelendirici efekti tarafından değiştirilen çizim API'sini kullanarak içerik oluşturur.

Örnek kod listelerinin çoğu iki bölümden oluşur. Birinci bölüm, örnekte kullanılan gölgelendiricinin Pixel Bender kaynak kodudur. Kaynak kodu Pixel Bender bayt koduna derlemek için öncelikle Pixel Bender Araç Kitini kullanmanız gerekir. Pixel Bender bayt kodu dosyasını oluşturmak için bu adımları izleyin:

1. Adobe Pixel Bender Araç Kiti'ni açın. Gerekirse, Oluştur menüsünden "Flash Player uyarı ve hatalarını açın" seçeneğini belirleyin.
2. Pixel Bender kod listesini kopyalayın ve Pixel Bender Araç Kiti'nin kod düzenleyici bölmesine yapıştırın.
3. Dosya menüsünden "Flash Player için çekirdek filtresini dışa aktar" seçeneğini belirleyin.

4 Pixel Bender bayt kodu dosyasını Flash belgesiyle aynı dizine kaydedin. Dosya adının, örnek açıklamasında belirtilen adla eşleşmesi gerekir.

Her örneğin ActionScript bölümü, sınıf dosyası olarak yazılır. Flash Professional'da örneği test etmek için:

1 Boş bir Flash belgesi oluşturun ve bu belgeyi bilgisayarınıza kaydedin.
2 Yeni bir ActionScript dosyası oluşturun ve bu dosyayı Flash belgesiyle aynı dizine kaydedin. Dosyanın adı ile, kod listesindeki sınıf adının eşleşmesi gerekir. Örneğin, kod listesi, MyApplication adında bir sınıfı tanımlıyorsa, ActionScript dosyasını kaydetmek için MyApplication.as adını kullanın.
3 Kod listesini ActionScript dosyasına kopyalayın ve dosyayı kaydedin.
4 Belgenin Özellik denetçisini etkinleştirmek için, Flash belgesinde Sahne Alanı'nın boş bir bölümünü veya çalışma alanını tıklatın.
5 Özellik denetçisinde Belge Sınıfı alanına, metinden kopyaladığınız ActionScript sınıfının adını girin.
6 Kontrol Et > Filmi Test Et komutunu kullanarak programı çalıştırın

   Önizleme penceresinde, örneğin sonuçlarını göreceksiniz.

Örnek kod listelemelerini sınamak için kullanılan bu teknikler "ActionScript Örnekleri Nasıl Kullanılır?" sayfa 1108 bölümünde ayrıntılı bir şekilde bulunabilir.

Gölgelendirici yükleme veya gömme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

ActionScript'te bir Pixel Bender gölgelendiricinin kullanılmasında ilk adım, ActionScript kodunuzda gölgelendiriciye erişilmesidir. Adobe Pixel Bender Toolkit kullanılarak oluşturulmuş ve Pixel Bender dilinde yazılmış olduğundan, gölgelendiriciye doğrudan ActionScript'ten erişilemez. Bunun yerine, ActionScript'e Pixel Bender gölgelendiricisini temsil eden bir Shader sınıfı örneği oluştururuz. Shader nesnesi, gölgelendiricinin parametre veya girdi görüntüsü değeri bekleyip beklemediği gibi, gölgelendiriciyle ilgili bilgileri bulmanıza olanak sağlar. Gölgelendiriciyi gerçekten kullanmak için Shader nesnesini diğer nesnelere iletirsiniz. Örneğin, gölgelendiriciyi filtre olarak kullanmak için Shader nesnesini bir ShaderFilter nesnesinin `shader` özelliğine atayın. Alternatif olarak, gölgelendiriciyi çizim dolgusu olarak kullanmak için Shader nesnesini `Graphics.beginShaderFill()` yöntemine bir argüman olarak iletirsiniz.

ActionScript kodunuz, Adobe Pixel Bender Toolkit (.pbj dosyası) tarafından oluşturulan bir gölgelendiriciye iki şekilde erişebilir:

- Çalışma zamanında yüklenen: URLLoader nesnesi kullanılarak bir gölgelendirici dosyası harici bir varlık olarak yüklenebilir. Bu teknik, metin dosyası gibi harici bir varlığın yüklenmesine benzer. Aşağıdaki örnek, çalışma zamanında gölgelendirici bayt kodunun yüklenmesini ve bir Shader örneğine bağlanmasını gösterir:

```
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
}
```

- SWF dosyasında gömülü: `[Embed]` meta veri etiketi kullanılarak derleme zamanında gölgelendirici dosyası SWF dosyasına gömülebilir. `[Embed]` meta veri etiketi yalnızca SWF dosyasını derlemek için Flex SDK kullanırsanız kullanılabilir. Bu örnekte olduğu gibi, `[Embed]` etiketinin `source` parametresi gölgelendirici dosyasına işaret eder ve bunun `mimeType` parametresi `"application/octet-stream"` şeklindedir:

```
[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
```

Her iki durumda da ham gölgelendirici bayt kodunu (`URLLoader.data` özelliği veya `[Embed]` veri sınıfının bir örneği) Shader örneğine bağlarsınız. Önceki örneklerde gösterildiği gibi, Shader örneğine bayt kodunu iki şekilde atayabilirsiniz. Gölgelendirici bayt kodunu argüman olarak `Shader()` yapıcısına iletebilirsiniz. Alternatif olarak bu kodu Shader örneğinin `byteCode` özelliği olarak da ayarlayabilirsiniz.

Pixel Bender gölgelendiricisi oluşturulup bir Shader nesnesine bağlandıktan sonra, efektler oluşturmak için birçok şekilde gölgelendiriciyi kullanabilirsiniz. Bunu bir filtre, karışım modu, bitmap dolgusu olarak veya bitmap ya da diğer verilerin bağımsız işlemesi için kullanabilirsiniz. Gölgelendiricinin meta verilerine erişmek, girdi görüntülerini belirtmek ve parametre değerlerini ayarlamak için Shader nesnesinin `data` özelliğini de kullanabilirsiniz.

Gölgelendirici meta verilerine erişme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Yazar, Pixel Bender gölgelendirici çekirdeği oluştururken, Pixel Bender kaynak kodunda gölgelendirici hakkında meta veriler belirtebilir. ActionScript'te bir gölgelendirici kullanırken, gölgelendiriciyi inceleyebilir ve meta verilerini ayıklayabilirsiniz.

Bir Shader örneği oluşturup bu örneği Pixel Bender gölgelendiricisine bağladığınızda, gölgelendirici hakkındaki verileri içeren bir ShaderData nesnesi oluşturulur ve Shader nesnesinin `data` özelliğinde saklanır. ShaderData sınıfı kendi özelliklerini tanımlamaz. Ancak, gölgelendirici kaynak kodunda tanımlanan her meta veri değeri için çalışma zamanında bir özellik dinamik olarak ShaderData nesnesine eklenir. Özelliklerin her birine verilen ad, meta veride belirtilen adla aynıdır. Örneğin, Pixel Bender gölgelendiricisinin kaynak kodunun şu meta veri tanımını içerdiğini varsayın:

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

Bu gölgelendirici için oluşturulan ShaderData nesnesi, şu özellik ve değerlerle oluşturulur:

- `namespace` (String): `"Adobe::Example"`
- `vendor` (String): `"Bob Jones"`
- `version` (String): `"1"`
- `description` (String): `"Creates a version of the specified image with the specified brightness"`

Meta veri özellikleri ShaderData nesnesine dinamik olarak eklendiğinden, ShaderData nesnesini incelemek için `for..in` döngüsünü kullanabilirsiniz. Bu tekniği kullanarak, gölgelendiricinin herhangi bir meta veri içerip içermediğini ve meta veri değerlerinin ne olduğunu öğrenebilirsiniz. ShaderData nesnesi, meta veri özelliklerine ek olarak, gölgelendiricide tanımlanmış girdileri ve parametreleri temsil eden özelliklere sahip olabilir. Bir ShaderData nesnesini incelemek için `for..in` döngüsünü kullandığınızda, özelliğin bir girdi mi (ShaderInput örneği), parametre mi (ShaderParameter örneği) yoksa meta veri değeri mi (String örneği) olduğunu belirlemek için her özelliğin veri türünü kontrol edin. Aşağıdaki örnek, `for..in` döngüsü kullanılarak bir gölgelendiricinin `data` özelliğinin dinamik özelliklerinin nasıl incelendiğini gösterir. Her meta veri değeri, `metadata` adındaki bir Vector örneğine eklenir. Bu örnekte, `myShader` adında bir Shader örneğinin önceden oluşturulduğunun varsayıldığını unutmayın:

```
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
```

Bu örneğin, gölgelendirici girdilerini ve parametrelerini de ayıklayan bir modeli için bkz. “Gölgelendirici girdilerini ve parametrelerini tanımlama” sayfa 307. Girdi ve parametre özellikleri hakkında daha fazla bilgi almak için, bkz. “Gölgelendirici girdisi ve parametre değerlerini belirtme” sayfa 307.

Gölgelendirici girdisi ve parametre değerlerini belirtme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Birçok Pixel Bender gölgelendiricisi, gölgelendirici işlemede kullanılan bir veya daha fazla girdi görüntüsünü kullanmak için tanımlanır. Örneğin, bir gölgelendiricinin belirli bir efekt uygulanmış bir kaynak görüntüsünü ve çıktısını kabul etmesi çok yaygın bir durumdur. Gölgelendiricinin nasıl kullanıldığına bağlı olarak, girdi değeri otomatik olarak belirtilebilir veya sizin açıkça bir değer sağlamanız gerekebilir. Aynı şekilde, birçok gölgelendirici, gölgelendirici çıktısını özelleştirmek için kullanılan parametreleri belirtir. Ayrıca gölgelendiriciyi kullanmadan önce her parametre için açıkça bir değer ayarlamanız gerekir.

Gölgelendirici girdilerini ve parametrelerini ayarlamak ve belirli bir gölgelendiricinin girdi mi yoksa parametre mi beklediğini belirlemek için Shader nesnesinin `data` özelliğini kullanırsınız. `data` özelliği bir ShaderData örneğidir.

Gölgelendirici girdilerini ve parametrelerini tanımlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Gölgelendirici girdisinin ve parametre değerlerinin belirtilmesinde birinci adım, kullandığınız belirli gölgelendiricinin herhangi bir girdi görüntüsü veya parametre bekleyip beklemediğini öğrenmektir. Her Shader örneği, ShaderData nesnesini içeren bir `data` özelliğine sahiptir. Gölgelendirici herhangi bir girdi veya parametreyi tanımlıyorsa, bu girdi ya da parametreye, ShaderData nesnesinin özellikleri olarak erişilir. Özelliklerin adları, gölgelendirici kaynak kodunda girdiler ve parametreler için belirtilen adlarla eşleşir. Örneğin, bir gölgelendirici `src` adındaki bir girdiyi tanımlıyorsa, ShaderData nesnesi o girdiyi temsil eden `src` adındaki bir özelliğe sahiptir. Bir girdiyi temsil eden her özellik ShaderInput örneğidir ve bir parametreyi temsil eden her özellik de ShaderParameter örneğidir.

İdeal olarak, gölgelendiricinin yazarı, gölgelendirici için gölgelendiricinin hangi girdi görüntüsü değerlerini ve parametrelerini beklediği, neyi temsil ettiği, uygun değerlerinin ne olduğu, vb. bilgileri sunan belgeler sağlar.

Ancak gölgelendirici bu tür bir belge içermiyorsa (ve kaynak koduna sahip değilseniz), girdileri ve parametreleri tanımlamak için gölgelendirici verilerini inceleyebilirsiniz. Girdileri ve parametreleri temsil eden özellikler, dinamik olarak ShaderData nesnesine eklenir. Sonuç olarak, `for..in` döngüsünü kullanarak, ShaderData nesnesinin ilişkilendirilmiş gölgelendiricisinin herhangi bir girdiyi mi yoksa parametreyi mi tanımladığını öğrenmek için ShaderData nesnesini inceleyebilirsiniz. "Gölgelendirici meta verilerine erişme" sayfa 305 bölümünde açıklandığı gibi, gölgelendirici için tanımlanan herhangi bir meta veri değerine `Shader.data` özelliğine eklenmiş bir dinamik özellik olarak da erişilebilir. Gölgelendirici girdilerini ve parametrelerini tanımlamak için bu tekniği kullandığınızda, dinamik özelliklerin veri türünü kontrol edin. Bir özellik ShaderInput örneğiyse, bir girdiyi temsil eder. Özellik ShaderParameter örneğiyse, bir parametreyi temsil eder. Aksi takdirde özellik bir meta veri değeridir. Aşağıdaki örnek, `for..in` döngüsü kullanılarak bir gölgelendiricinin `data` özelliğinin dinamik özelliklerinin nasıl incelendiğini gösterir. Her girdi (ShaderInput nesnesi), `inputs` adındaki bir Vector örneğine eklenir. Her parametre (ShaderParameter nesnesi), `parameters` adındaki bir Vector örneğine eklenir. Son olarak, her meta veri `metadata` adındaki bir Vector örneğine eklenir. Bu örnekte, `myShader` adında bir Shader örneğinin önceden oluşturulduğunun varsayıldığını unutmayın:

```
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
```

Gölgelendirici girdisi değerlerini belirtme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Birçok gölgelendirici, gölgelendirici işlemede kullanılan bir veya daha fazla girdi görüntüsü bekler. Ancak birçok durumda, Shader nesnesi kullanıldığında otomatik olarak bir girdi belirtilir. Örneğin, bir gölgelendiricinin tek bir girdi gerektirdiğini ve bu gölgelendiricinin de filtre olarak kullanıldığını varsayın. Bir görüntüleme nesnesine veya BitmapData nesnesine filtre uygulandığında, o nesne otomatik şekilde girdi olarak ayarlanır. Bu durumda açıkça bir girdi değeri belirtmeniz gerekmez.

Ancak bazı durumlarda, özellikle gölgelendirici birden çok girdi tanımlıyorsa, girdi için açıkça bir değer ayarlarsınız. Bir gölgelendiricide tanımlanan her girdi, bir ShaderInput nesnesi tarafından ActionScript'te temsil edilir. "Gölgelendirici girdilerini ve parametrelerini tanımlama" sayfa 307 bölümünde açıklandığı gibi, ShaderInput nesnesi, Shader nesnesinin `data` özelliğinde ShaderData örneğinin bir özelliğidir. Örneğin, bir gölgelendiricinin `src` adında bir girdiyi tanımladığını ve bu gölgelendiricinin `myShader` adındaki bir Shader nesnesine bağlandığını varsayın. Bu durumda, şu tanımlayıcıyı kullanarak `src` girdisine karşılık gelen ShaderInput nesnesine erişirsiniz:

```
myShader.data.src
```

Her ShaderInput nesnesi, girdinin değerini ayarlamak için kullanılan bir `input` özelliğine sahiptir. Görüntü verilerini belirtmek için `input` özelliğini bir BitmapData örneğine ayarlarsınız. Ayrıca `input` özelliğini bir BitmapData veya Vector öğesine ayarlayabilirsiniz.<Number> örneğine ayarlayabilirsiniz. BitmapData veya Vector.Girdi olarak <Number> örneği için, `http://help.adobe.com/tr_TR/FlashPlatform/reference/actionscript/3/flash/display/ShaderInput.html#input`'ndaki ShaderInput.input listesine bakın.

ShaderInput nesnesi, `input` özelliğine ek olarak girdinin ne tür bir görüntü beklediğini belirlemek için kullanılabilen özelliklere de sahiptir. Bu özellikler arasında `width`, `height` ve `channels` özellikleri yer alır. Her ShaderInput nesnesi ayrıca girdi için açıkça bir değerin sağlanması gerekip gerekmediğinin belirlenmesinde kullanışlı olan `index` özelliğine de sahiptir. Gölgelendirici otomatik olarak ayarlı sayıdan daha fazla girdi bekliyorsa, bu durumda söz konusu girdilerin değerlerini ayarlarsınız. Gölgelendiricinin farklı kullanım şekilleri ve girdi değerlerinin otomatik olarak ayarlanıp ayarlanmaması hakkında ayrıntılı bilgi almak için, bkz. "Gölgelendirici kullanma" sayfa 312.

Gölgelendirici parametre değerlerini belirtme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bazı gölgelendiriciler, gölgelendiricinin sonuçlarını oluşturmakta kullandığı parametre değerlerini tanımlar. Örneğin, bir görüntünün parlaklığını değiştiren bir gölgelendirici, işlemin parlaklığı ne kadar etkilediğini belirleyen bir parlaklık parametresini belirtebilir. Gölgelendiricide tanımlanan tek bir parametre, gölgelendiricideki parametre tanımına göre tek bir değer veya birden çok değer bekleyebilir. Bir gölgelendiricide tanımlanan her parametre, bir ShaderParameter nesnesi tarafından ActionScript'te temsil edilir. "Gölgelendirici girdilerini ve parametrelerini tanımlama" sayfa 307 bölümünde açıklandığı gibi, ShaderParameter nesnesi, Shader nesnesinin data özelliğinde ShaderData örneğinin bir özelliğidir. Örneğin, bir gölgelendiricinin `brightness` adında bir parametreyi tanımladığını ve bu gölgelendiricinin `myShader` adındaki bir Shader nesnesi tarafından temsil edildiğini varsayın. Bu durumda, şu tanımlayıcıyı kullanarak `brightness` parametresine karşılık gelen ShaderParameter nesnesine erişirsiniz:

```
myShader.data.brightness
```

Parametre için bir değer (veya değerler) ayarlamak üzere, değer ya da değerleri içeren bir ActionScript dizisi oluşurun ve ShaderParameter nesnesinin `value` özelliğine bu diziyi atayın. Tek bir gölgelendirici parametresi birden çok değer gerektirebildiğinden, `value` özelliği bir Array örneği olarak tanımlanır. Gölgelendirici parametresi yalnızca tek bir değer beklese de, değeri `ShaderParameter.value` özelliğine atamak için bir Array nesnesinde sarmanız gerekir. Aşağıdaki liste, `value` özelliği olarak tek bir değer ayarlanmasını gösterir:

```
myShader.data.brightness.value = [75];
```

Gölgelendiricinin Pixel Bender kaynak kodu, parametrenin varsayılan değerini tanımlıyorsa, Shader nesnesi oluşturulduğunda varsayılan değeri veya değerleri içeren bir dizi oluşturulur ve ShaderParameter nesnesinin `value` özelliğine atanır. `value` özelliğine bir dizi atandıktan sonra (varsayılan diziyse dahil edilir), dizi öğesinin değeri değiştirilerek parametre değeri değiştirilebilir. Yeni bir dizi oluşturup bunu `value` özelliğine atamanız gerekmez.

Aşağıdaki örnek, ActionScript'te bir gölgelendiricinin parametre değerinin ayarlanmasını gösterir. Bu örnekte, gölgelendirici `color` adında bir parametreyi tanımlar. `color` parametresi, Pixel Bender kaynak kodunda bir `float4` değişkeni olarak bildirilir, başka bir deyişle bu, dört kayan nokta numarası dizisidir. Örnekte, `color` parametre değeri sürekli olarak değiştirilir ve bu değer her değiştiğinde ekranda renkli bir dikdörtgen çizmek için gölgelendirici kullanılır. Sonuçta, animasyon uygulanmış bir renk değişimi elde edilir.

Not: *Bu örneğin kodunu Ryan Taylor yazmıştır. Bu örneği bizimle paylaştığı için Ryan'a teşekkür ediyoruz. Şu adreste Ryan'ın portföyünü görebilir ve yazılarını okuyabilirsiniz: www.boostworthy.com.*

ActionScript kodunun merkezinde üç yöntem bulunur:

- `init()`: `init()` yönteminde kod, gölgelendiriciyi içeren Pixel Bender bayt kodu dosyasını yükler. Dosya yüklendiğinde, `onLoadComplete()` yöntemi çağrılır.
- `onLoadComplete()`: `onLoadComplete()` yönteminde kod, `shader` adında bir Shader nesnesi oluşturur. Ayrıca `texture` adında bir Sprite örneği oluşturur. `renderShader()` yönteminde kod her kare için bir defa gölgelendirici sonucunu `texture` öğesine çizer.
- `onEnterFrame()`: Her kare için birer kere `onEnterFrame()` yöntemi çağrılarak animasyon efekti oluşturulur. Bu yöntemde kod, gölgelendirici parametresi değerini yeni renge ayarlar ve sonra bir dikdörtgen olarak gölgelendirici sonucunu çizmek için `renderShader()` yöntemini çağırır.
- `renderShader()`: `renderShader()` yönteminde kod, bir gölgelendirici dolgusu belirtmek için `Graphics.beginShaderFill()` yöntemini çağırır. Daha sonra gölgelendirici çıktısı (oluşturulan renk) tarafından dolgusu tanımlanan bir dikdörtgen çizer. Bir gölgelendiricinin bu şekilde kullanılması hakkında daha fazla bilgi almak için, bkz. "Gölgelendiriciyi çizim dolgusu olarak kullanma" sayfa 312.

Aşağıda, bu örneğin ActionScript kodu verilmiştir. Bu sınıfı Flash Builder'da yalnızca ActionScript projesi için ana uygulama sınıfı olarak veya Flash Professional'da FLA dosyasının belge sınıfı olarak kullanın:

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

Aşağıda, “ColorFilter.pbj” Pixel Bender bayt kodu dosyasını oluşturmak için kullanılan, ColorFilter gölgelendirici çekirdeğinin kaynak kodu verilmiştir:

```
<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;
    }
}
```

Parametreleri belgelenmemiş bir gölgelendirici kullanıyorsanız, ShaderParameter nesnesinin `type` özelliğini kontrol ederek, dizide hangi tür öğelerden kaç tane bulunması gerektiğini öğrenebilirsiniz. `type` özelliği, gölgelendiricide tanımlandığı şekilde parametrenin veri türünü belirtir. Her parametre türünün beklediği öğelerin sayısını ve türünü belirten bir liste için, ActionScript 3.0 Başvurusundaki `ShaderParameter.value` özelliğine bakın.

Her ShaderParameter nesnesi aynı zamanda parametrenin gölgelendiricinin parametrelerinde hangi sırada olduğunu belirten bir `index` özelliğine de sahiptir. ShaderParameter nesnesi, bu özelliklere ek olarak, gölgelendiricinin yazarı tarafından sağlanan meta veri değerlerini içeren ek özelliklere de sahip olabilir. Örneğin, yazar bir parametre için minimum, maksimum ve varsayılan değerler gibi meta veri değerleri belirtebilir. Yazarın belirttiği tüm meta veri değerleri, dinamik özellikler olarak ShaderParameter nesnesine eklenir. Bu özellikleri incelemek için, `for..in` döngüsünü kullanarak, ShaderParameter nesnesinin dinamik özellikleri üzerinde döngü uygulayarak nesnenin meta verilerini tanımlayın. Aşağıdaki örnek, `for..in` döngüsünün nasıl bir ShaderParameter nesnesinin meta verisini tanımlamak için kullanıldığını gösterir. Her meta veri değeri, `metadata` adındaki bir Vector örneğine eklenir. Bu örnekte, `myShader` adında bir Shader örneğinin önceden oluşturulduğu ve bunun `brightness` adında bir parametreye sahip olduğunun varsayıldığını unutmayın:

```
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
```

Gölgelendirici kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Pixel Bender gölgelendiricisi, ActionScript'te bir Shader nesnesi olarak kullanılabilir olduktan sonra, birçok şekilde kullanılabilir:

- Gölgelendirici çizim dolgusu: Gölgelendirici, çizim api'si kullanılarak çizilen bir şeklin dolgu kısmını tanımlar
- Karışım modu: Gölgelendirici, iki örtüşen görüntüleme nesnesi arasındaki karışımı tanımlar
- Filtre: Gölgelendirici, görsel içeriğin görünümünü değiştiren bir filtreyi tanımlar
- Bağımsız gölgelendirici işleme: Gölgelendirici işleme, çıktının tasarlanan kullanımı belirtilmeden çalıştırılır. Gölgelendirici isteğe bağlı olarak arka planda çalışabilir ve işlem tamamlandığında sonuç kullanılabilir olur. Bitmap verileri oluşturmak ve görsel olmayan verileri işlemek için bu teknik kullanılabilir.

Gölgelendiriciyi çizim dolgusu olarak kullanma

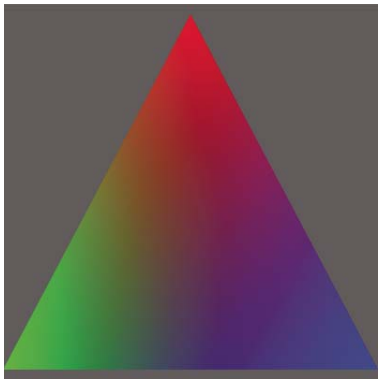
Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Çizim dolgusu oluşturmak için bir gölgelendirici kullandığınızda, vektör şekli oluşturmak için çizim api'si yöntemlerini kullanırsınız. Herhangi bir bitmap görüntüsünün, çizim api'si ile bitmap dolgusu olarak kullanılabilmesinde olduğu gibi, şekli doldurmak için gölgelendiricinin çıktısı kullanılır. Gölgelendirici dolgusu oluşturmak için, kodunuzda şekli çizmeye başlamak istediğiniz noktada, Graphics nesnesinin `beginShaderFill()` yöntemini çağırın. Bu listede gösterildiği gibi, `beginShaderFill()` yöntemine birinci argüman olarak Shader nesnesini iletin:

```
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
```

Bir gölgelendiriciyi çizim dolgusu olarak kullandığınızda, gölgelendiricinin gerektirdiği herhangi bir girdi görüntüsü değerini ve parametre değerini ayarlarsınız.

Şu örnek, gölgelendiricinin çizim dolgusu olarak kullanılmasını göstermektedir. Bu örnekte gölgelendirici üç noktalı degrade oluşturur. Bu degrade, renklerin her biri üçgenin bir ucunda ve aralarında degrade olacak şekilde üç renk içerir. Ayrıca, renkler animasyon uygulanmış dönen renk efekti oluşturmak için döner.



Not: *Bu örneğin kodu Petri Leskinen tarafından yazılmıştır. Bu örneği bizimle paylaştığı için Petri'ye teşekkür ediyoruz. Petri'nin örneklerinin ve öğreticilerinin daha fazlasını http://pixelero.wordpress.com adresinde görebilirsiniz.*

ActionScript kodu üç moddadır:

- `init()`: Uygulama yüklendiğinde `init()` yöntemi çağrılır. Bu yöntemde kod, üçgenin noktalarını temsil eden Point nesnelerinin başlangıç değerlerini ayarlar. Ayrıca, kod `canvas` adında bir Sprite örneği oluşturur. Daha sonra `updateShaderFill()` yönteminde kod her kare için bir defa gölgelendirici sonucunu `canvas` öğesine çizer. Son olarak kod, gölgelendirici bayt kodu dosyasını yükler.
- `onLoadComplete()`: `onLoadComplete()` yönteminde kod, `shader` adında bir Shader nesnesi oluşturur. Ayrıca başlangıç parametre değerlerini ayarlar. Son olarak kod, `enterFrame` olayının dinleyicisi olarak `updateShaderFill()` yöntemini ekler, bu animasyon efekti oluşturmak için her kare için bir defa bu yöntemi çağırdığı anlamına gelir.
- `updateShaderFill()`: Her kare için birer kere `updateShaderFill()` yöntemi çağrılarak animasyon efekti oluşturulur. Bu yöntemde kod, gölgelendirici parametrelerinin değerlerini hesaplar ve ayarlar. Daha sonra kod, gölgelendirici dolgusu oluşturmak için `beginShaderFill()` yöntemini çağırır ve gölgelendirici sonucunu bir üçgende çizmek için başka çizim api'si yöntemlerini çağırır.

Aşağıda, bu örneğin ActionScript kodu verilmiştir. Flex'te yalnızca ActionScript projesi için ana uygulama sınıfı olarak veya Flash geliştirme aracındaki bir FLA dosyasının belge sınıfı olarak bu sınıfı kullanın:

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

Aşağıda, “ThreePointGradient.pbj” Pixel Bender bayt kodu dosyasını oluşturmak için kullanılan, ThreePointGradient gölgelendirici çekirdeğinin kaynak kodu verilmiştir:

```
<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);
    }
}
```

Not: *Grafik işleme birimi (GPU) altında görüntü oluştururken bir gölgelendirici dolgusu kullanırsa doldurulan alanın rengi cam göbeği olur.*

Çizim api'sini kullanarak şekil çizme hakkında daha fazla bilgi için, bkz. "Çizim API'sini kullanma" sayfa 225.

Gölgelendiriciyi harmanlama modu olarak kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Gölgelendiricinin harmanlama modu olarak kullanılması, diğer harmanlama modlarının kullanılmasına benzer. Gölgelendirici, görsel olarak birbiriyle harmanlanmış iki görüntüleme nesnesinin oluşturduğu görünümü tanımlar. Gölgelendiriciyi harmanlama modu olarak kullanmak için, Shader nesnenizi ön plan görüntüleme nesnesinin `blendShader` özelliğine atayın. `blendShader` özelliğine `null` dışında bir değer atanması, görüntüleme nesnesinin `blendMode` özelliğini otomatik şekilde `BlendMode.SHADER` olarak ayarlar. Aşağıdaki listede bir gölgelendiricinin harmanlama modu olarak kullanılması gösterilmektedir. Bu örnekte, `foreground` öğesinin diğer içeriklerin üzerine geldiğinin ve görüntüleme listesinde diğer görüntüleme içerikleriyle aynı üst öğede bulunan `foreground` adında bir görüntüleme nesnesi olduğunun varsayıldığını unutmayın:

```
foreground.blendShader = myShader;
```

Gölgelendiriciyi harmanlama modu olarak kullandığınızda, gölgelendiricinin en az iki girdiyle tanımlanması gerekir. Örnekte gösterildiği gibi, girdi değerlerini kodunuzda ayarlamazsınız. Bunun yerine, iki harmanlanmış görüntü otomatik şekilde gölgelendirici girdisi olarak kullanılır. Ön plan görüntüsü ikinci görüntü olarak ayarlanır. (Bu, harmanlama modunun uygulandığı görüntüleme nesnesidir.) Ön plan görüntüsünün sınırlama kutusunun arkasındaki tüm piksellerin bileşiği alınarak bir arka plan görüntüsü oluşturulur. Arka plan görüntüsü, birinci girdi görüntüsü olarak ayarlanır. İkiden çok girdi bekleyen bir gölgelendirici kullanırsanız, bu iki girdi dışındaki tüm girdiler için bir değer sağlamanız gerekir.

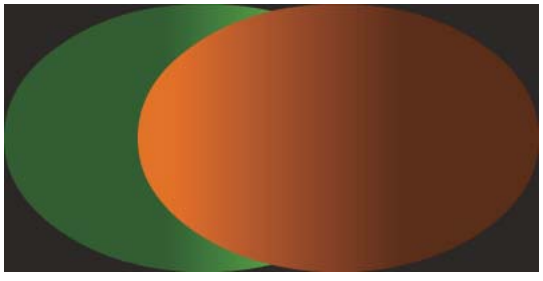
Aşağıdaki örnekte bir gölgelendiricinin harmanlama modu olarak kullanılması gösterilmektedir. Bu örnek, renk parlaklığını esas alarak açık bir harmanlama modu kullanır. Harmanlama sonucunda, harmanlanan nesnelerin herhangi birindeki en açık piksel değeri, görüntülenen piksel olur.

Not: *Bu örneğin kodu Mario Klingemann tarafından yazılmıştır. Bu örneği bizimle paylaştığı için Mario'ya teşekkür ediyoruz. Şu adreste Mario'nun çalışmalarını görebilir ve yazılarını okuyabilirsiniz www.quasimondo.com.*

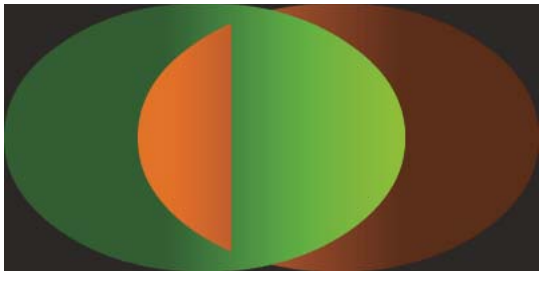
Bu iki örnekte önemli ActionScript kodu bulunur:

- `init()`: Uygulama yüklendiğinde `init()` yöntemi çağrılır. Bu yöntemde kod, gölgelendirici bayt kodu dosyasını yükler.
- `onLoadComplete()`: `onLoadComplete()` yönteminde kod, `shader` adında bir Shader nesnesi oluşturur. Daha sonra üç nesne çizer. Bunlardan ilki olan `backdrop`, harmanlanmış nesnelerin arkasında bulunan koyu gri bir arka plandır. İkincisi olan `backgroundShape`, yeşil degrade elipstir. Üçüncü nesne olan `foregroundShape` ise turuncu degrade elipstir.

  `foregroundShape` elips, harmanlamanın ön plan nesnesidir. Harmanlamanın arka plan görüntüsünün bir bölümünü `backdrop` öğesinin bir parçası ve bir bölümünü de üstüne `foregroundShape` nesnesinin sınırlama kutusu gelen `backgroundShape` öğesinin bir parçası oluşturur. `foregroundShape` nesnesi, görüntüleme listesinde en öndeki nesnedir. Bu kısmen `backgroundShape` öğesinin ve tamamen `backdrop` öğesinin üzerine gelir. Bu üst üste gelme nedeniyle, harmanlama modu uygulanmadan turuncu elips (`foregroundShape`) tamamen gösterilirken, yeşil elipsin (`backgroundShape`) bir kısmı gizlenir:



  Ancak harmanlama modu uygulandığında, yeşil elipsin açık renkli kısmı, üzerini örten `foregroundShape` bölümünden daha açık olduğundan "saydam olarak" görünür:



Aşağıda, bu örneğin ActionScript kodu verilmiştir. Bu sınıfı Flash Builder'da yalnızca ActionScript projesi için ana uygulama sınıfı olarak veya Flash Professional'da FLA dosyasının belge sınıfı olarak kullanın:

```
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;
        }
    }
}
```

Aşağıda, “LumaLighten.pbj” Pixel Bender bayt kodu dosyasını oluşturmak için kullanılan, LumaLighten gölgelendirici çekirdeğinin kaynak kodu verilmiştir:

```
<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;
    }
}
```

Harmanlama modlarını kullanma hakkında daha fazla bilgi için, bkz. “Karışım modları uygulama” sayfa 189.

Not: *Pixel Bender gölgelendirici programı Flash Player'da veya AIR'de bir karışım olarak çalıştırıldığında, örnekleme ve* `outCoord()` *işlevleri, diğer bağlamlarda olduğundan daha farklı davranır. Bir karışımda örnekleme işlevi, her zaman gölgelendirici tarafından değerlendirilmekte olan geçerli pikseli geri döndürür. Örneğin komşu bir pikseli örneklemek için* `outCoord()` *işlevine bir uzaklık eklenemez. Aynı şekilde,* `outCoord()` *işlevini örnekleme dışında bir işlev için kullanırsanız, koordinatları her zaman 0 olarak değerlendirilir. Örneğin karıştırılan görüntülerin birleştirilme şeklini etkilemek için bir pikselin konumunu kullanamazsınız.*

Gölgelendiriciyi filtre olarak kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Gölgelendiricinin filtre olarak kullanılması, ActionScript'teki herhangi bir filtrenin kullanılmasına benzer. Gölgelendiriciyi filtre olarak kullandığınızda, filtre uygulanan görüntü (görüntüleme nesnesi veya BitmapData nesnesi) gölgelendiriciye iletilir. Gölgelendirici, genellikle orijinal görüntünün değiştirilmiş bir sürümü olan filtre çıktısını oluşturmak için girdi görüntüsünü kullanır. Filtre uygulanan nesne bir görüntüleme nesnesiyse, ekranda filtre uygulanan görüntüleme nesnesi yerine gölgelendiricinin çıktısı görüntülenir. Filtre uygulanan nesne bir BitmapData nesnesiyse, gölgelendiricinin çıktısı, `applyFilter()` yöntemi çağrılan BitmapData nesnesinin içeriği olur.

Bir gölgelendiriciyi filtre olarak kullanmak için, ilk olarak "Gölgelendirici yükleme veya gömme" sayfa 304 bölümünde açıklandığı gibi Shader nesnesini oluştururusunuz. Daha sonra Shader nesnesine bağlanmış bir ShaderFilter nesnesi oluştururusunuz. ShaderFilter nesnesi, filtre uygulanmış nesneye uygulanan filtredir. Bunu herhangi bir filtreyi uyguladığınız şekilde bir nesneye uygulayabilirsiniz. Bunu görüntüleme nesnesinin `filters` özelliğine iletirsiniz veya bir BitmapData nesnesinde `applyFilter()` öğesini çağırırsınız. Örneğin, aşağıdaki kod bir ShaderFilter nesnesi oluşturur ve `homeButton` adındaki bir görüntüleme nesnesine filtreyi uygular.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

Gölgelendiriciyi filtre olarak kullandığınızda, gölgelendiricinin en az bir girdiyle tanımlanması gerekir. Örnekte gösterildiği gibi, girdi değerini kodunuzda ayarlamazsınız. Bunun yerine, filtre uygulanmış görüntüleme nesnesi veya BitmapData nesnesi, girdi görüntüsü olarak ayarlanır. Birden çok girdi bekleyen bir gölgelendirici kullanırsanız, bu bir girdi dışındaki tüm girdiler için bir değer sağlamanız gerekir.

Bazı durumlarda filtre, orijinal görüntünün boyutlarını değiştirir. Örneğin, tipik bir gölge efekti, görüntüye eklenen gölgeyi içeren fazladan pikseller ekler. Görüntü boyutlarını değiştiren bir gölgelendirici kullandığınızda, görüntü boyutunun ne kadar değişmesini istediğinizi belirtmek için `leftExtension`, `rightExtension`, `topExtension` ve `bottomExtension` özelliklerini ayarlayın.

Aşağıdaki örnekte bir gölgelendiricinin filtre olarak kullanılması gösterilmektedir. Bu örnekteki filtre, görüntünün kırmızı, yeşil ve mavi kanal değerlerini tersine çevirir. Sonuçta görüntünün "negatif" sürümü elde edilir.

Not: *Bu örneğin kullandığı gölgelendirici, Pixel Bender Araç Kiti'ne dahil edilen invertRGB.pbk Pixel Bender çekirdeğidir. Pixel Bender Araç Kiti yükleme dizininden çekirdek için kaynak kodu yükleyebilirsiniz. Kaynak kodu derleyin ve bayt kodu dosyasını kaynak koduyla aynı dizine kaydedin.*

Bu iki örnekte önemli ActionScript kodu bulunur:

- `init()`: Uygulama yüklendiğinde `init()` yöntemi çağrılır. Bu yöntemde kod, gölgelendirici bayt kodu dosyasını yükler.

- `onLoadComplete()`: `onLoadComplete()` yönteminde kod, `shader` adında bir Shader nesnesi oluşturur. Daha sonra `target` adında bir nesne oluşturur ve bu nesnenin içeriklerini çizer. `target` nesnesi, solda kırmızı, ortada sarı-yeşil ve sağda açık mavi olan doğrusal degrade renkle doldurulmuş bir dikdörtgendir. Filtre uygulanmamış nesne şöyle görünür:

Filtre uygulandığında renkler ters çevrilir ve böylece dikdörtgen şöyle görünür:

Bu örneğin kullandığı gölgelendirici, Pixel Bender Araç Kiti'ne dahil edilen “invertRGB.pbk” örnek Pixel Bender çekirdeğidir. Kaynak kod, Pixel Bender Araç Kiti yükleme dizininde “invertRGB.pbk” dosyasında mevcuttur. Kaynak kodu derleyin ve bayt kodu dosyasını ActionScript kaynak kodunuzla aynı dizine “invertRGB.pbj” adıyla kaydedin.

Aşağıda, bu örneğin ActionScript kodu verilmiştir. Bu sınıfı Flash Builder'da yalnızca ActionScript projesi için ana uygulama sınıfı olarak veya Flash Professional'da FLA dosyasının belge sınıfı olarak kullanın:

```
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];
        }
    }
}
```

Filtreleri uygulama hakkında daha fazla bilgi için, bkz. “Filtreler oluşturma ve uygulama” sayfa 271.

Gölgelendiriciyi bağımsız modda kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir gölgelendiriciyi bağımsız modda kullandığınızda, gölgelendirici işleme, çıktıyı nasıl kullanmayı planladığınızdan bağımsız olarak çalışır. Çalıştırılacak bir gölgelendirici belirtirsiniz, girdi ve parametre değerlerini ayarlarsınız ve sonuçta elde edilen verilerin yerleştirileceği bir nesne atarsınız. İki nedenden dolayı bağımsız modda bir gölgelendirici kullanabilirsiniz:

- Görüntü dışındaki verileri işleme: Bağımsız modda, gölgelendiriciye bitmap görüntü verisi yerine rastgele bir ikili veri veya sayı verisi iletmeyi seçebilirsiniz. Gölgelendirici sonucunun bitmap görüntü verisine ek olarak ikili veri veya sayı verisi olarak döndürülmesini seçebilirsiniz.
- Arka planda işleme: Bir gölgelendiriciyi bağımsız modda çalıştırdığınızda, varsayılan olarak gölgelendirici eşzamansız şekilde çalıştırılır. Başka bir deyişle, uygulamanız çalışmaya devam ederken gölgelendirici arka planda çalışır ve gölgelendirici işlemesi sona erince bu, kodunuza bildirilir. Çalıştırılması uzun süren ve gölgelendirici çalışırken uygulama kullanıcı arabirimini veya diğer işlemeleri dondurmayan bir gölgelendirici kullanabilirsiniz.

Gölgelendiriciyi bağımsız modda çalıştırmak için bir ShaderJob nesnesi kullanabilirsiniz. İlk olarak ShaderJob nesnesi oluşturur ve bunu çalıştırılacak gölgelendiriciyi temsil eden Shader nesnesine bağlarsınız:

```
var job:ShaderJob = new ShaderJob(myShader);
```

Daha sonra, gölgelendiricinin beklediği girdi veya parametre değerlerini ayarlarsınız. Gölgelendiriciyi arka planda çalıştırıyorsanız, ayrıca ShaderJob nesnesinin `complete` olayı için bir dinleyici kaydedersiniz. Gölgelendirici işini bitirdiğinde dinleyiciniz çağrılır:

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

Daha sonra, işlem sona erince gölgelendirici işlemi sonucunun yazıldığı bir nesne oluşturursunuz. Bu nesneyi ShaderJob nesnesinin `target` özelliğine atarsınız:

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

Görüntü işlemeyi gerçekleştirmek için ShaderJob kullanıyorsanız, `target` özelliğine bir BitmapData atayın. İkili veri veya sayı verisi işliyorsanız, `target` özelliğine bir ByteArray nesnesi veya Vector <Number> örneği atayın. Bu durumda, `target` nesnesine verilecek veri miktarını belirtmek için ShaderJob nesnesinin `width` ve `height` özelliklerini ayarlamanız gerekir.

Not: *`ShaderJob()` yapıcısına şu şekilde argümanlar ileterek tek bir adımda ShaderJob nesnesinin `shader`, `target`,`width` ve `height` özelliklerini ayarlayabilirsiniz:`var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);`*

Gölgelendiriciyi çalıştırmaya hazır olduğunuzda, ShaderJob nesnesinin `start()` yöntemini çağırırsınız:

```
job.start();
```

Varsayılan olarak, `start()` yönteminin çağrılması, ShaderJob öğesinin senkronize olmadan çalıştırılmasına neden olur. Bu durumda program çalıştırması, gölgelendiricinin sona ermesi beklenmeden hemen bir sonraki kod satırıyla birlikte devam eder. Gölgelendirici işlemi sona erdiğinde, ShaderJob nesnesi `complete` olay dinleyicilerini çağırarak işlemin bittiğini bildirir. Bu noktada (başka bir deyişle, `complete` olay dinleyicinizin gövdesinde), `target` nesnesi gölgelendirici işleminin sonucunu içerir.

Not: *`target` özellik nesnesini kullanmak yerine, gölgelendirici sonucunu doğrudan dinleyici yönteminize iletilen olay nesnesinden alabilirsiniz. Olay nesnesi bir ShaderEvent örneğidir. ShaderEvent nesnesi, `target` özelliği olarak ayarladığınız nesnenin veri türüne bağlı olarak sonuca erişmek için kullanılabilecek üç özelliğe sahiptir: `ShaderEvent.bitmapData`, `ShaderEvent.byteArray` ve `ShaderEvent.vector`.*

Alternatif olarak, `start()` yöntemine bir `true` argümanı iletebilirsiniz. Bu durumda gölgelendirici işlemi eşzamanlı olarak çalıştırılır. Tüm kod (kullanıcı arabirimiyle etkileşim ve diğer olaylar dahil), gölgelendirici çalıştırılırken duraklatılır. Gölgelendirici sona erdiğinde, `target` nesnesi, gölgelendirici sonucunu içerir ve program bir sonraki kod satırıyla devam eder.

```
job.start(true);
```

Bölüm 16: Film klipleriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

MovieClip sınıfı, Adobe® Flash® geliştirme ortamınızda oluşturduğunuz animasyon ve film klibi sembollerine ilişkin çekirdek sınıftır. Bu sınıf, görüntüleme nesnelerinin tüm davranış ve işlevselliğine sahiptir ancak bunun yanı sıra onun zaman çizelgesini denetlemeye yönelik ek özellikler ve yöntemler de içerir.

Film kliplerinin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Film klipleri, Flash geliştirme aracıyla animasyon uygulanmış içerik oluşturan ve bu içeriği ActionScript ile denetlemek isteyen kişiler için önemli bir öğedir. Flash uygulamasında her film klibi sembolü oluşturduğunuzda, Flash, o Flash belgesinin kütüphanesine sembolü ekler. Varsayılan olarak bu sembol, MovieClip sınıfı ve benzerleri MovieClip sınıfının özelliklerine ve yöntemlerine sahiptir.

Sahne Alanı'na bir film klibi sembolü örneği yerleştirildiğinde, ActionScript kullanılarak film klibinin oynatımı değiştirilmediği sürece film klibi otomatik olarak zaman çizelgesinde (birden çok kare içeriyorsa) ilerler. MovieClip sınıfını diğerlerinden ayıran bu zaman çizelgesi olup sizin hareket üzerinden animasyon oluşturmanıza veya Flash geliştirme aracı üzerinden ara şekiller oluşturmanıza olanak sağlar. Bunun tersine, Sprite sınıfının örneği olan bir görüntüleme nesnesiyle yalnızca nesnenin değerlerini değiştirerek programlama yoluyla animasyon oluşturabilirsiniz.

Önceki ActionScript sürümlerinde, MovieClip sınıfı, Sahne Alanı'ndaki tüm örneklerin temel sınıfıydı. ActionScript 3.0'da ise film klibi, ekranda görüntülenebilen çok sayıdaki görüntüleme nesnesinden yalnızca biridir. Bir görüntüleme nesnesinin işlevi için zaman çizelgesi gerekli değilse, MovieClip sınıfı yerine Shape sınıfının veya Sprite sınıfının kullanılması, oluşturma performansını artırabilir. Bir göreve yönelik uygun görüntüleme nesnesinin seçilmesiyle ilgili daha fazla bilgi için, bkz. “DisplayObject alt sınıfı seçme” sayfa 175.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi film kliplerine ilişkin önemli terimler içerir:

AVM1 SWF ActionScript 1.0 veya ActionScript 2.0 kullanılarak oluşturulmuş, genellikle Flash Player 8 ya da öncesini hedefleyen bir SWF dosyası.

AVM2 SWF Adobe Flash Player 9 veya sonrası ya da Adobe AIR için ActionScript 3.0 kullanılarak oluşturulmuş bir SWF dosyası.

Harici SWF Proje SWF dosyasından ayrı olarak oluşturulmuş ve proje SWF dosyasına yüklenmek ve o SWF dosyası içinde oynatılmak üzere tasarlanmış bir SWF dosyası.

Kare Zaman çizelgesindeki en küçük zaman bölümü. Hareketli resim film şeridinde olduğu gibi, her kare, animasyonun zaman içindeki bir anlık görüntüsüdür ve kareler sırayla hızlı şekilde oynatıldığında animasyon efekti oluşturulur.

Zaman Çizelgesi Film klibinin animasyon sırasını oluşturan kareler dizisinin mecazi temsilidir. Bir MovieClip nesnesinin zaman çizelgesi, Flash geliştirme aracındaki zaman çizelgesine eşdeğerdir.

Oynatma Kafası Zaman çizelgesinde belirli bir anda görüntülenen konumu (kareyi) tanımlayan bir işaretçi.

MovieClip nesneleriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir SWF dosyasını yayınladığınızda, Flash uygulaması Sahne Alanı'ndaki tüm film klibi sembolü örneklerini MovieClip nesnelerine dönüştürür. Özellik denetçisinin Örnek Adı alanında film klibi sembolüne bir örnek adı vererek o film klibi sembolünü ActionScript için kullanılabilir duruma getirebilirsiniz. SWF dosyası oluşturulduğunda, Flash uygulaması Sahne Alanı'nda MovieClip örneğini oluşturan ve örnek adını kullanarak bir değişken bildiren kodu oluşturur. Diğer adlandırılmış film kliplerinin içinde yuvalanmış film kliplerini adlandırdıysanız, bu alt film klipleri, üst film klibinin özellikleri gibi değerlendirilir—nokta sözdizimini kullanarak alt film klibine erişebilirsiniz. Örneğin, `childClip` örnek adına sahip bir film klibi, `parentClip` örnek adına sahip başka bir klibin içinde yuvalanırsa, bu kodu çağırarak alt klibin zaman çizelgesi animasyonunu oynatabilirsiniz:

```
parentClip.childClip.play();
```

Not: *: Flash geliştirme aracında Sahne Alanı'na yerleştirilen alt örnekler, kod çalıştırmasında üst örneğin yapıcısı içinde oluşturulmadıklarından, bu alt örneklere üst örneğin yapıcısı içindeki kodla erişilemez. Alt öğeye erişmeden önce üst öğenin kodla alt örneği oluşturması veya `Event.ADDED_TO_STAGE` olayını göndermek için alt öğeyi dinleyen geri çağrı işlevine erişimi geciktirmesi gerekir.*

ActionScript 2.0 MovieClip sınıfının bazı eski yöntem ve özellikleri aynı kalsa da diğerleri değişmiştir. Alt çizgi ile önek olarak eklenen tüm özellikler yeniden adlandırılmıştır. Örneğin, `_width` ve `_height` özelliklerine şimdi `width` ve `height` olarak erişilirken, `_xscale` ve `_yscale` öğelerine de şimdi `scaleX` ve `scaleY` olarak erişilir. MovieClip sınıfının özelliklerinin ve yöntemlerinin eksiksiz listesi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'na bakın.

Film klibi oynatımını denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash, animasyonu veya bir durum değişikliğini yürütmek için zaman çizelgesi benzetimini kullanır. Zaman çizelgesi kullanan herhangi bir görsel öğenin MovieClip nesnesi olması veya MovieClip sınıfından genişletilmesi gerekir. ActionScript herhangi bir film klibine durmasını, oynatmasını veya zaman çizelgesinde başka bir noktaya gitmesini bildirse de, dinamik olarak bir zaman çizelgesi oluşturmak ya da belirli karelere içerik eklemek için ActionScript kullanılamaz; bu yalnızca Flash geliştirme aracı kullanılarak mümkün olur.

Bir MovieClip oynatılırken, SWF dosyasının kare hızı tarafından dikte edilen bir hızda zaman çizelgesinde ilerler. Alternatif olarak, ActionScript'te `Stage.frameRate` özelliğini ayarlayarak bu ayarı geçersiz kılabilirsiniz.

Film kliplerini oynatma ve oynatmayı durdurma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`play()` ve `stop()` yöntemleri, zaman çizelgesi boyunca bir film klibinin temel denetimine olanak sağlar. Örneğin, Sahne Alanı'nda, örnek adı `bicycle` olarak ayarlanmış ve ekranda hareket eden bir bisiklet animasyonunu içeren bir film klibi sembolü olduğunu varsayın. Aşağıdaki kod, ana zaman çizelgesinde bir anahtar kareye eklenir,

```
bicycle.stop();
```

bisiklet hareket etmez. (Animasyonu oynatılmaz.) Bisikletin hareketi, bir kullanıcı etkileşimi yoluyla başlatılabilir. Örneğin, `startButton` adında bir düğmeniz varsa, ana zaman çizelgesi üzerinde anahtar karedeki şu kod, düğmenin tıklatılmasıyla animasyonun oynatılmasını sağlar:

```
// 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);
```

Hızlı ileri ve geri sarma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`play()` ve `stop()` yöntemleri, film klibinde oynatımı denetlemenin tek yolu değildir. `nextFrame()` ve `prevFrame()` yöntemlerini kullanarak da zaman çizelgesi boyunca oynatma kafasını elle ileri veya geri hareket ettirebilirsiniz. Bu yöntemlerden herhangi biri çağrıldığında oynatma durdurulur ve oynatma kafası sırayla bir kare ileri veya geri hareket ettirilir.

`play()` yönteminin kullanılması, film klibi nesnesinin `enterFrame` olayı her tetiklendiğinde `nextFrame()` öğesinin çağrılmasıyla aynıdır. Bu satırlar boyunca, aşağıdaki gibi, `enterFrame` olayı için bir olay dinleyicisi oluşturup `bicycle` öğesine dinleyici işlevinde bir önceki karesine gitmesini bildirerek `bicycle` film klibinin geriye doğru hareket etmesini sağlayabilirsiniz:

```
// 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);
```

Normal oynatımda, bir film klibi birden çok kare içeriyorsa, oynatma sırasında bu süresiz olarak döngü oluşturur; başka bir deyişle, son karesini geçerse Kare 1'e geri döner. `prevFrame()` veya `nextFrame()` öğesini kullandığınızda, bu davranış otomatik olarak oluşmaz. (Oynatma kafası Kare 1'deyken `prevFrame()` çağrılırsa, oynatma kafası son kareye taşınmaz.) Yukarıdaki örnekte `if` koşulu, oynatma kafasının birinci kareye doğru geri yönde ilerleyip ilerlemediğini kontrol eder ve oynatma kafasını son karesine ayarlayarak geriye doğru oynatılan film klibinin etkili şekilde sürekli bir döngüsünü oluşturur.

Farklı bir kareye atlama ve kare etiketleri kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yeni bir kareye film klibi gönderilmesi basit bir işlemdir. `gotoAndPlay()` veya `gotoAndStop()` öğesinin çağrılmasıyla, film klibi bir parametre tarafından belirtilen kare sayısına atlar. Alternatif olarak, bir kare etiketinin adıyla eşleşen bir dizeyi iletebilirsiniz. Zaman çizelgesindeki herhangi bir kare bir etikete atanabilir. Bunu yapmak için, zaman çizelgesinde bir kare seçin ve sonra Özellik denetçisinde Kare Etiketi alanına bir ad girin.

Sayı yerine kare etiketlerinin kullanılmasının avantajı, özellikle karmaşık bir olay klibi oluşturulurken daha belirgin olur. Animasyondaki karelerin, katmanların ve araların sayısı arttıkça, önemli kareleri, film klibinin davranışındaki kaydırmaları temsil eden bilgilendirici açıklamalarla (örneğin, "kapalı", "yürüyor", "çalışıyor") etiketleyin. Etiketlenmiş bir kareye giden ActionScript çağrıları, belirli bir kare sayısının değil, tek bir başvurunun (etiket) işaretçisi olduğundan bu, kodun okunabilirliğini artırır ve esneklik sağlar. Daha sonra animasyonun belirli bir parçasını farklı bir kareye taşımaya karar verirseniz, yeni konumda karelerin etiketlerini aynı tuttuğunuz sürece ActionScript kodunuzu değiştirmeniz gerekmez.

ActionScript 3.0, koddaki kare etiketlerini temsil etmek için FrameLabel sınıfını içerir. Bu sınıfın her örneği tek bir kare etiketini temsil eder ve Özellik denetçisinde belirtildiği gibi kare etiketinin adını temsil eden bir `name` özelliğine ve etiketin zaman çizelgesinde yerleştirildiği karenin kare sayısını temsil eden bir `frame` özelliğine sahiptir.

Bir film klibi örneğiyle ilişkilendirilmiş FrameLabel örneklerine erişmek için, MovieClip sınıfı doğrudan FrameLabel nesnelerini döndüren iki özellik içerir. `currentLabels` özelliği, bir film klibinin tüm zaman çizelgesi boyunca yer alan tüm FrameLabel nesnelerini içeren bir dizi döndürür. `currentLabel` özelliği, zaman çizelgesi boyunca en son karşılaşılan kare etiketinin adını içeren bir dize döndürür.

`robot` adında bir film klibi oluşturduğunuzu ve bunun animasyonunun çeşitli durumlarını etiketlediğinizi varsayın. Aşağıdaki kodda olduğu gibi, `robot` öğesinin geçerli durumuna erişmek için `currentLabel` özelliğini denetleyen bir koşul oluşturabilirsiniz:

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

Flash Player 11.3 ve AIR 3.3, `frameLabel` olayını FrameLabel sınıfına ekledi. Bir kare etiketini temsil eden FrameLabel örneğine bir olay işleyicisi atayabilirsiniz. Oynatma kafası kareye girdiğinde olay gönderilir.

Aşağıdaki örnek, MovieClip'e yönelik kare etiketi dizisindeki ikinci kare etiketi için bir FrameLabel örneği oluşturur. Ardından `frameLabel` olayı için bir olay işleyicisi kaydeder:

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

Sahnelerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash geliştirme ortamında, SWF dosyasının ilerleyeceği zaman çizelgeleri dizisini ayırmak için sahneleri kullanabilirsiniz. `gotoAndPlay()` veya `gotoAndStop()` yöntemlerinin ikinci parametresini kullanarak, oynatma kafasının gönderileceği bir sahneyi belirtebilirsiniz. Tüm FLA dosyaları yalnızca başlangıç sahnesinde başlar ancak siz yeni sahneler oluşturabilirsiniz.

Sahnelerin birçok dezavantajı olduğundan, sahnelerin kullanılması her zaman en iyi yaklaşım olmayabilir. Birden çok sahne içeren bir Flash belgesinin korunması, özellikle de birden çok yazarın bulunduğu ortamlarda zor olabilir. Yayınlama işlemi tüm sahneleri tek bir zaman çizelgesinde birleştirdiğinden, birden çok sahne, bant genişliğinde de verimsiz olabilir. Bu da asla oynatılmasa da, tüm sahnelerin aşamalı indirilmesine neden olur. Bu nedenlerden dolayı, birden çok uzun zaman çizelgeli animasyonların organize edilmesi dışında, birden çok sahne kullanılması genellikle önerilmez.

MovieClip sınıfının `scenes` özelliği, SWF dosyasındaki tüm sahneleri temsil eden bir Scene nesneleri dizisini döndürür. `currentScene` özelliği, geçerli olarak oynatılan sahne alanını temsil eden bir Scene nesnesini döndürür.

Scene sınıfı, bir sahneyle ilgili bilgi veren birçok özelliğe sahiptir. `labels` özelliği, o sahnedeki kare etiketlerini temsil eden bir FrameLabel nesneleri dizisini döndürür. `name` özelliği, dize olarak sahnenin adını döndürür. `numFrames` özelliği, sahnedeki toplam kare sayısını temsil eden bir int öğesini döndürür.

ActionScript ile MovieClip nesneleri oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash uygulamasında ekrana içerik eklemenin bir yolu, varlıkları kitaplıktan Sahne Alanı'na sürüklemektir ancak tek iş akışı bu değildir. Karmaşık projeler için, deneyimli geliştiriciler programlama yoluyla film klipleri oluşturmayı tercih eder. Bu yaklaşım birçok avantaj sağlar: kodun daha kolay kullanılması, derleme zamanı hızının daha yüksek olması ve yalnızca ActionScript için kullanılabilir olan daha gelişmiş değiştirmeler.

ActionScript 3.0'ın görüntüleme listesi API'si, dinamik olarak MovieClip nesneleri oluşturma işlemini kolaylaştırır. Görüntüleme listesine ekleme işleminden ayrı olarak, doğrudan bir MovieClip örneği başlatma yeteneği, denetimden taviz verilmeden esneklik ve basitlik sağlar.

ActionScript 3.0'da, programlama yoluyla bir film klibi (veya başka bir görüntüleme nesnesi) örneği oluşturduğunuzda, bu örnek, görüntüleme nesnesi konteynerinde `addChild()` veya `addChildAt()` yöntemi çağrılarak görüntüleme listesine ekleninceye kadar ekranda görünmez. Bu durum, bir film klibi oluşturmanıza, bunun özelliklerini ayarlamanıza ve hatta film klibi ekranda oluşturulmadan önce yöntemleri çağırmanıza olanak sağlar. Görüntüleme listesiyle çalışma hakkında daha fazla bilgi için, bkz. “Görüntüleme nesnesi kaplarıyla çalışma” sayfa 161.

ActionScript için kütüphane sembollerini dışa aktarma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varsayılan olarak, bir Flash belgesinin kütüphanesindeki film klibi sembollerinin örnekleri dinamik olarak oluşturulamaz (başka bir deyişle, yalnızca ActionScript kullanılarak oluşturulur) Bunun nedeni, ActionScript'te kullanılmak üzere dışa aktarılan her sembolün SWF dosyanızın boyutunu artırması ve bazı sembollerin sahne alanında kullanılmak üzere tasarlanmamış olabilmesidir. Bu nedenle, bir sembolü ActionScript'te kullanılabilir duruma getirmek için, sembolün ActionScript için dışa aktarılması gerektiğini belirtmeniz gerekir.

ActionScript için bir sembolü dışa aktarmak üzere:

1. Kütüphane panelinde sembolü seçin ve sembolün Sembol Özellikleri iletişim kutusunu açın.
2. Gerekirse, Gelişmiş ayarları etkinleştirin.
3. Bağlantı bölümünde, ActionScript için Dışa Aktar onay kutusunu etkinleştirin.

   Böylece Sınıf ve Temel Sınıf alanları etkinleştirilir.

Varsayılan olarak Sınıf alanında boşlukları kaldırılmış şekilde sembol adı bulunur (örneğin, "Ağaç Ev" adındaki bir sembol, "AğaçEv" olur). Sembolün davranışı için özel bir sınıf kullanması gerektiğini belirtmek üzere, paketi de dahil olmak üzere sınıfın tam adını bu alana girin. ActionScript'te sembolün örneklerini oluşturabilmeyi istiyorsanız ancak ek bir davranış eklemeniz gerekmiyorsa, sınıf adını olduğu gibi bırakabilirsiniz.

Temel Sınıf alanının değeri varsayılan olarak `flash.display.MovieClip` şeklindedir. Sembolünüzün başka bir müşteri sınıfının işlevselliğini genişletmesini istiyorsanız, bu sınıf Sprite (veya MovieClip) sınıfını genişlettiği sürece sınıfın adını belirtebilirsiniz.

4 Değişiklikleri kaydetmek için Tamam düğmesine basın.

Bu noktada Flash, bağlantılı bir SWC dosyasını veya belirtilen sınıfın bir tanımını içeren harici bir ActionScript dosyasını bulamazsa (örneğin, sembol için ek davranış eklemeniz gerekmediyse), bir uyarı görüntülenir:

Bu sınıfın tanımı sınıf yolunda bulunamadı, bu yüzden tanım dışa aktarma sırasında SWF dosyasında otomatik olarak üretilecek.

Kütüphane sembolünüz, MovieClip sınıfının işlevselliğinin ötesinde benzersiz bir işlevsellik gerektirmiyorsa, bu uyarıyı yoksayabilirsiniz.

Sembolünüz için bir sınıf sağlamazsanız, Flash uygulaması sembolünüz için buna eşdeğer bir sınıf oluşturur:

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

Sembolünüze fazladan bir ActionScript işlevselliği eklemek istiyorsanız, aşağıdaki kod yapısına uygun özellikleri ve yöntemleri ekleyin. Örneğin, 50 piksel genişliğinde ve 50 piksel yüksekliğinde daire içeren bir film klibi sembolünüz olduğunu ve sembolün Circle adındaki bir sınıfla ActionScript için dışa aktarılmak üzere belirtildiğini varsayın. Aşağıdaki kod, Circle.as dosyasına yerleştirildiğinde, MovieClip sınıfını genişletir ve sembole `getArea()` ve `getCircumference()` ek yöntemlerini sağlar:

```
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;
        }
    }
}
```

Flash belgesinin Kare 1'inde anahtar kareye yerleştirilen aşağıdaki kod, sembolün bir örneğini oluşturup ekranda görüntüler:

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

Bu kod, tek tek varlıkların Sahne Alanı'na sürüklenmesine alternatif olarak ActionScript tabanlı başlatmayı gösterir. Bu, bir film klibinin tüm özelliklerine sahip olan ve Circle sınıfında tanımlanan özel yöntemleri de içeren bir daire oluşturur. Bu çok basit bir örnektir—kütüphane sembolünüz, sınıfında herhangi bir sayıda özellik ve yöntemleri belirtebilir.

ActionScript tabanlı başlatma, elle düzenlenmesi zor olacak büyük miktarda örnekleri dinamik olarak oluşturmanıza olanak sağladığından güçlüdür. Ayrıca oluşturma sırasında her örneğin özelliklerini özelleştirebildiğiniz için esneklik de sağlar. Dinamik olarak birçok Circle örneği oluşturmak için bir döngü kullanarak bu iki avantajı daha iyi kavrayabilirsiniz. Flash belgenizin kütüphanesinde önceden açıklanan Circle sembolü ve sınıfıyla, aşağıdaki kodu Kare 1'de bir anahtar kareye yerleştirin:

```
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);
}
```

Bu, kod kullanarak bir sembolün birden çok örneğini nasıl hızlı şekilde oluşturup özelleştirebileceğinizi gösterir. Her örnek, döngü içinde geçerli sayısı esas alınarak konumlandırılır ve `transform` özelliği (Circle öğesinin MovieClip sınıfını genişleterek miras aldığı özellik) ayarlanarak her örneğe rastgele bir renk verilir.

Harici bir SWF dosyasını yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da, SWF dosyaları Loader sınıfı kullanılarak yüklenir. Harici bir SWF dosyasını yüklemek için, ActionScript uygulamanızın dört şey yapması gerekir:

1. Dosyanın url'si ile yeni bir URLRequest nesnesi oluşturma.
2. Yeni bir Loader nesnesi oluşturma.
3. Loader nesnesinin `load()` yöntemini çağırıp URLRequest örneğini parametre olarak iletme.
4. Loader örneğini görüntüleme listesine eklemek için, görüntüleme nesnesi konteynerinde (örn. bir Flash belgesinin ana zaman çizelgesinde) `addChild()` yöntemini çağırma.

Sonuç olarak kod şöyle görünür:

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

Bu aynı kod, SWF dosyasının url'si yerine görüntü dosyasının url'si belirtilerek JPEG, GIF veya PNG görüntüsü gibi harici bir görüntü dosyasını yüklemek için kullanılabilir. Görüntü dosyasından farklı olarak bir SWF dosyası ActionScript içerebilir. Bu nedenle, SWF dosyası yükleme işlemi görüntü yükleme işlemiyle aynı gibi görünse de, Flash Player veya AIR uygulaması SWF dosyasını oynatıyorsa ve harici SWF dosyasıyla bir şekilde iletişim kurmak için ActionScript kullanmayı planlıyorsanız, harici bir SWF dosyası yüklenirken hem yüklemeyi yapan SWF dosyası hem de yüklenen SWF dosyasının aynı güvenlik sanal alanında bulunması gerekir. Ayrıca, harici SWF dosyası, yükleyen SWF dosyasındaki sınıflarla aynı ad alanını paylaşan sınıflar içeriyorsa, ad alanı çakışmalarını önlemek için, yüklenen SWF dosyası için yeni bir uygulama etki alanı oluşturmanız gerekir. Güvenlik ve uygulama etki alanı konularıyla ilgili daha fazla bilgi için bkz. "Uygulama etki alanlarıyla çalışma" sayfa 149 ve "İçerik yükleme" sayfa 1071.

Harici bir SWF dosyası başarıyla yüklendiğinde, `Loader.content` özelliği üzerinden bu dosyaya erişilebilir. Harici SWF dosyası ActionScript 3.0 için yayınlanırsa, hangi sınıfı genişlettiğine bağlı olarak bu bir film klibi veya hareketli grafik olur.

Diğer platformlarla karşılaştırıldığında, Adobe AIR for iOS'ta SWF dosyasını yükleme ile ilgili birkaç farklılık vardır. Daha fazla bilgi için bkz. "AIR for iOS'ta SWF dosyalarını yükleme" sayfa 205.

Eski bir SWF dosyası yüklenirken dikkat edilmesi gereken noktalar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici bir SWF dosyası eski bir ActionScript sürümüyle yayınlanmışsa, dikkat edilmesi gereken önemli sınırlamalar vardır. AVM2'de (ActionScript Virtual Machine 2) çalışan bir ActionScript 3.0 SWF dosyasından farklı olarak, ActionScript 1.0 veya 2.0 için yayınlanmış bir SWF dosyası AVM1'de (ActionScript Virtual Machine 1) çalışır.

Bir ActionScript 3.0 dosyasına bir ActionScript 1.0 veya 2.0 SWF dosyası yüklerken önemli farklar vardır (bir ActionScript 3.0 SWF dosyası yüklemeyle karşılaştırıldığında). Flash Player, önceden yayınlanmış içerikle geriye doğru tam uyumluluk sağlar. Flash Player'ın önceki sürümlerinde çalışan herhangi bir içerik, ActionScript 3.0'ı destekleyen Flash Player sürümlerinde çalışır. Ancak, aşağıdaki sınırlar uygulanır:

- ActionScript 3.0 kodu ActionScript 1.0 veya 2.0'da yazılı bir SWF dosyasını yükleyebilir. Bir ActionScript 1.0 veya 2.0 SWF dosyası başarıyla yüklendiğinde, yüklenen nesne (`Loader.content` özelliği) bir AVM1Movie nesnesidir. AVM1Movie örneği, MovieClip örneğiyle aynı değildir. Bu bir görüntüleme nesnesidir ancak film klibinden farklı olarak, zaman çizelgesiyle ilgili yöntemler veya özellikler içermez. Üst AVM2 SWF dosyasının, yüklenen AVM1Movie nesnesinin özelliklerine, yöntemlerine veya nesnelerine erişimi olamaz.
- ActionScript 1.0 veya 2.0'da yazılmış SWF dosyaları, ActionScript 3.0'da yazılmış SWF dosyalarını yükleyemez. Bu da, Flash 8 veya Flex Builder 1.5 ya da önceki sürümlerde yazılmış SWF dosyalarının ActionScript 3.0 SWF dosyalarını yükleyemediği anlamına gelir.

  Bu kuralın tek istisnası, ActionScript 2.0 SWF dosyası, düzeylerinden birine önceden herhangi bir şey yüklemediği sürece, ActionScript 2.0 SWF dosyasının bir ActionScript 3.0 SWF dosyasıyla kendisini değiştirebilmesidir. ActionScript 2.0 SWF dosyası, `loadMovieNum()` öğesine bir çağrı yapıp `level` parametresine 0 değerini ileterek bunu yapabilir.
- Genelde, ActionScript 3.0'da yazılmış SWF dosyalarıyla birlikte çalışacaksa, ActionScript 1.0 veya 2.0'da yazılmış SWF dosyalarının alınması gerekir. Örneğin, ActionScript 2.0'ı kullanarak bir medya oynatıcısı oluşturduğunuzu varsayın. Medya oynatıcısı, ayrıca ActionScript 2.0'da oluşturulmuş çeşitli içerikleri de yükler. ActionScript 3.0'da yeni içerik oluşturup bu içeriği medya oynatıcısına yükleyemezsiniz. Video oynatıcısını ActionScript 3.0'a almanız gerekir.

  Ancak ActionScript 3.0'da bir ortam oynatıcısı oluşturursanız, bu ortam oynatıcısı ActionScript 2.0 içeriğinizin basit yüklemelerini gerçekleştirebilir.

Aşağıdaki tablolarda, önceki Flash Player sürümlerinin daha yeni içerik yüklenmesi ve kod çalıştırılmasıyla ilgili sınırlamaları ve farklı ActionScript sürümlerinde yazılmış SWF dosyaları arasında çapraz komut dosyası oluşturma sınırlamaları özetlenmektedir.

Desteklenen işlevler	Flash Player 7	Flash Player 8	Flash Player 9 ve 10
Şunlar için yayınlanmış SWF dosyalarını yükleyebilir	7 ve öncesi	8 ve öncesi	9 (veya 10) ve öncesi
Bu AVM'yi içerir	AVM1	AVM1	AVM1 ve AVM2
Şu ActionScript sürümlerinde yazılmış SWF'leri çalıştırır	1.0 ve 2.0	1.0 ve 2.0	1.0 ve 2.0 ve 3.0

Aşağıdaki tabloda yer alan “Desteklenen işlevler”, Flash Player 9 veya sonrasında çalıştırılan içerikleri ifade eder. Flash Player 8 veya öncesinde çalışan içerikler yalnızca ActionScript 1.0 ve 2.0'da yüklenebilir, görüntülenebilir, çalıştırılabilir ve bu içeriklerin çapraz komut dosyası oluşturulabilir.

Desteklenen işlevler	ActionScript 1.0 ve 2.0'da oluşturulan içerik	ActionScript 3.0'da oluşturulan içerik
Şu sürümlerde oluşturulan içerik yüklenebilir ve içerikteki kod çalıştırılabilir	yalnızca ActionScript 1.0 ve 2.0	ActionScript 1.0, 2.0 ve ActionScript 3.0
Şu sürümlerde oluşturulan içeriğin çapraz komut dosyası oluşturulabilir	yalnızca ActionScript 1.0 ve 2.0 (Yerel Bağlantı üzerinden ActionScript 3.0)	Yerel Bağlantı üzerinden ActionScript 1.0 ve 2.0 ActionScript 3.0

Film klibi örneği: RuntimeAssetsExplorer

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript için Dışa Aktar işlevselliği, özellikle birden çok projede kullanışlı olabilecek kütüphaneler için avantajlıdır. Flash Player veya AIR bir SWF dosyasını çalıştırırsa, ActionScript'e dışa aktarılmış semboller, yüklemeyi yapan SWF ile aynı güvenlik sanal alanında bulunan tüm SWF dosyaları için kullanılabilir olur. Bu şekilde tek bir Flash belgesi, yalnızca grafiksel varlıkları barındırma amacıyla tasarlanmış bir SWF dosyası oluşturabilir. Bu teknik özellikle görsel varlıklar üzerinde çalışan tasarımcıların, daha sonra çalışma zamanında grafiksel varlıklar SWF dosyasını yükleyen "sarıcı" bir SWF dosyası oluşturan geliştiricilerle paralel çalışabildiği daha büyük projeler için kullanışlıdır. Grafiksel varlıkların programlama geliştirmesinin ilerlemesine bağlı olmadığı sürümlenmiş dosyalar dizisini korumak için bu yöntemi kullanabilirsiniz.

RuntimeAssetsExplorer uygulaması, RuntimeAsset öğesinin alt sınıfı olan herhangi bir SWF dosyasını yükler ve bu SWF dosyasının kullanılabilir varlıklarına gözatmanıza olanak sağlar. Bu örnekte şunlar gösterilir:

- `Loader.load()` öğesini kullanarak harici bir SWF dosyasını yükleme
- ActionScript için dışa aktarılmış bir kütüphane sembolünün dinamik oluşumu
- MovieClip oynatımının ActionScript denetimi

Başlamadan önce, Flash Player'da çalıştırılacak SWF dosyalarının her birinin aynı güvenlik sanal alanında bulunması gerektiğini unutmayın. Daha fazla bilgi için, bkz. “Güvenlik sanal alanları” sayfa 1055.

Bu örneğin uygulama dosyalarını almak için, Flash Professional Örnekleri dosyasını indirin. RuntimeAssetsExplorer uygulama dosyalarını Samples/RuntimeAssetsExplorer klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
RuntimeAssetsExample.mxml veya RuntimeAssetsExample.fla	Flex (MXML) veya Flash (FLA) uygulaması için kullanıcı arabirimi.
RuntimeAssetsExample.as	Flash (FLA) uygulaması için belge sınıfı.
GeometricAssets.as	RuntimeAsset arabirimini uygulayan örnek bir sınıf.
GeometricAssets.fla	ActionScript için dışa aktarılan sembolleri içeren GeometricAssets sınıfına (FLA'nın belge sınıfı) bağlı bir FLA dosyası.
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	Gezgin konteynerine yüklenecek tüm çalışma zamanı varlığı SWF dosyalarından beklenen zorunlu yöntemleri tanımlayan bir arabirim.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	Dönen bir kutu şeklinde kütüphane sembolünün sınıfı.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	Dönen bir yıldız şeklinde kütüphane sembolünün sınıfı.

Çalışma zamanı kütüphane arabirimi oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Gezginin düzgün şekilde SWF kütüphanesiyle etkileşim kurması için, çalışma zamanı varlığı kütüphanelerinin yapısının biçimlendirilmesi gerekir. Beklenen bir yapıyı ayıran yöntemlerin şeması olması yönüyle bir sınıfa benzeyen ancak sınıftan farklı olarak herhangi bir yöntem gövdesi içermeyen bir arabirim oluşturarak bunu gerçekleştireceğiz. Arabirim, hem çalışma zamanı kütüphanesinin hem de gezginin birbiriyle iletişim kurması için bir yol sağlar. Tarayıcımıza yüklenen çalışma zamanı varlıklarının her SWF dosyası bu arabirimi uygular. Arabirimler ve arabirimlerin nasıl kullanışlı olabileceği hakkında daha fazla bilgi için, *ActionScript 3.0'ı Öğrenme*'deki Arabirimler'e bakın.

RuntimeLibrary arabirimi çok basit olacaktır—yalnızca içe aktarılacak olan ve çalışma zamanı kütüphanesinde kullanılabilir olan sembollerin sınıf yollarının bir dizisini gezgine sağlayabilecek bir işlev gerektirmekteyiz. Bu amaçla, arabirim tek bir yönteme sahiptir: `getAssets()`.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

Varlık kütüphanesi SWF dosyası oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

RuntimeLibrary arabirimi tanımlanarak, SWF dosyalarının başka bir SWF dosyasına yüklenebileceği birden çok varlık kütüphanesi oluşturulması mümkündür. Tek tek varlıkların SWF kütüphanesinin oluşturulması dört görev içerir:

- Varlık kütüphanesi SWF dosyası için bir sınıf oluşturma

- Kütüphanede bulunan tek tek varlıklar için sınıflar oluşturma
- Gerçek grafik varlıkları oluşturma
- Grafik öğelerini sınıflarla ilişkilendirme ve kütüphane SWF dosyasını yayınlama

RuntimeLibrary arabirimini uygulamak için bir sınıf oluşturma

Daha sonra, RuntimeLibrary arabirimini uygulayan bir GeometricAssets sınıfını oluşturacağız. Bu, FLA dosyasının belge sınıfı olacaktır. Bu sınıfın kodu, RuntimeLibrary arabirimine çok benzer—bunlar arasındaki tek fark, sınıf tanımında `getAssets()` yönteminin bir yöntem gövdesine sahip olmasıdır.

```
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" ];
        }
    }
}
```

İkinci bir çalışma zamanı kütüphanesi oluşturacak olsaydık, kendi `getAssets()` uygulamasını sağlayan başka bir sınıfı (örneğin, AnimationAssets) esas alan başka bir FLA oluşturabilirdik.

Her MovieClip varlığı için sınıflar oluşturma

Bu örnek için, özel varlıklara herhangi bir işlevsellik eklemeden yalnızca MovieClip sınıfını genişleteceğiz. Aşağıdaki AnimatingStar kodu, AnimatingBox kodu ile paraleldir:

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

Kütüphaneyi yayınlama

Şimdi yeni bir FLA oluşturup Özellik denetçisinin Belge Sınıfı alanına GeometricAssets öğesini girerek MovieClip tabanlı varlıkları yeni sınıfa bağlayacağız. Bu örneğin amaçları doğrultusunda, 360 kare üzerinden saat yönünde tek bir dönüş gerçekleştirmek için bir ara zaman çizelgesi kullanan iki çok temel şekil oluşturacağız. Hem `animatingBox` hem de `animatingStar` sembolleri, ActionScript için Dışa Aktar olarak ayarlanır ve Sınıf alanı, `getAssets()` uygulamasında belirtilen ilgili sınıf yollarına ayarlanmıştır. Standart MovieClip yöntemlerinin alt sınıfını oluşturmak istediğimizden, `flash.display.MovieClip` öğesinin varsayılan temel sınıfı kalır.

Sembolünüzün dışa aktarma ayarlarını yaptıktan sonra FLA'yı yayınlayın. Şimdi ilk çalışma zamanı kütüphanenize sahipsiniz. Bu SWF dosyası başka bir AVM2 SWF dosyasına yüklenebilir ve AnimatingBox ve AnimatingStar sembolleri, yeni SWF dosyası için kullanılabilir olur.

Kütüphaneyi başka bir SWF dosyasına yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulanacak son işlev, varlık gezgininin kullanıcı arabirimidir. Bu örnekte, çalışma zamanı kütüphanesinin yolu `ASSETS_PATH` adında bir değişken olarak kodlanmıştır. Alternatif olarak, örneğin, sabit sürücünüzde belirli bir SWF dosyasına gözatan bir arabirim oluşturmak için, FileReference sınıfını oluşturabilirsiniz.

Çalışma zamanı kütüphanesi başarıyla yüklendiğinde, Flash Player `runtimeAssetsLoadComplete()` yöntemini çağırır:

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

Bu yöntemde, rl değişkeni yüklenen SWF dosyasını temsil eder. Kod, yüklenen SWF dosyasının `getAssets()` yöntemini çağırarak kullanılabilir olan varlıkların listesini alır ve `populateDropDown()` yöntemini çağırarak bir ComboBox bileşenini kullanılabilir varlıkların bir listesiyle doldurmak için bunları kullanır. Daha sonra da bu yöntem her varlığın tam sınıf yolunu saklar. Kullanıcı arabiriminde Ekle düğmesi tıklatıldığında `addAsset()` yöntemi tetiklenir:

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

bu yöntem, ComboBox öğesinde geçerli olarak seçilmiş varlığın sınıf yolunu (`assetNameCbo.selectedItem.data`) alır ve o varlığın yeni bir örneğini oluşturmak üzere varlığın sınıfına gerçek bir başvuruyu almak için `getDefinitionByName()` işlevini (flash.utils paketinden) kullanır.

Bölüm 17: Hareket aralarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

"Nesnelere animasyon uygulama" sayfa 196 bölümünde, ActionScript'te komut dosyası içeren animasyonun nasıl uygulandığı açıklanır.

Burada, farklı bir animasyon oluşturma tekniğini açıklayacağız: ara hareketler. Bu teknik, Adobe® Flash® Professional'ı kullanarak bir belgede etkileşimli şekilde hareket ayarlayarak hareket oluşturmanıza olanak verir. Daha sonra, çalışma zamanında dinamik ActionScript tabanlı animasyonunuzda bu hareketi kullanabilirsiniz.

Flash Professional, ara hareket uygulayan ActionScript'i otomatik olarak oluşturur ve bunu kopyalayıp yeniden kullanmanız için uygun duruma getirir.

Hareket araları oluşturmak için Flash Professional lisansına sahip olmanız gerekir.

Daha fazla Yardım konusu

fl.motion paketi

Hareket Arası Temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Ara hareketler, animasyon oluşturmanın kolay bir yolunu sağlar.

Ara hareket, konum ve dönüş gibi görüntüleme nesnesi özelliklerini kare kare değiştirir. Ara hareket aynı zamanda çeşitli filtreler ve başka özellikler uygulayıp hareket ederken görüntüleme nesnesinin görünümünü de değiştirir. Ara hareket için ActionScript kodu oluşturan Flash Professional ile etkileşimli olarak ara hareket oluştururuz. Flash uygulamasında, ara hareketi oluşturan ActionScript'i kopyalamak için Hareketi ActionScript 3.0 olarak Kopyala komutunu kullanın. Böylece, çalışma zamanında kendi dinamik animasyonunuzda hareket oluşturmak için ActionScript'i yeniden kullanabilirsiniz.

Hareket arası oluşturma hakkında bilgi almak için *Flash Professional'ı Kullanma* kısmında Hareket Araları bölümüne bakın.

Önemli kavramlar ve terimler

Şu terim, bu özelliğe ilişkin önemli bir terimdir:

Ara hareket Farklı zamanlarda farklı durumlarda bir görüntüleme nesnesinin ara karelerini oluşturan bir yapı; bu, birinci durumun yumuşak şekilde ikinci duruma dönüşmesi görünümünü sağlar. Sahne alanı üzerinde bir görüntüleme nesnesini taşımak ve zamanla nesneye ışıma uygulamak, nesneyi daraltmak, döndürmek, soldurmak veya nesnenin rengini değiştirmek için kullanılır.

Ara hareket komut dosyalarını kopyalama Flash uygulamasında

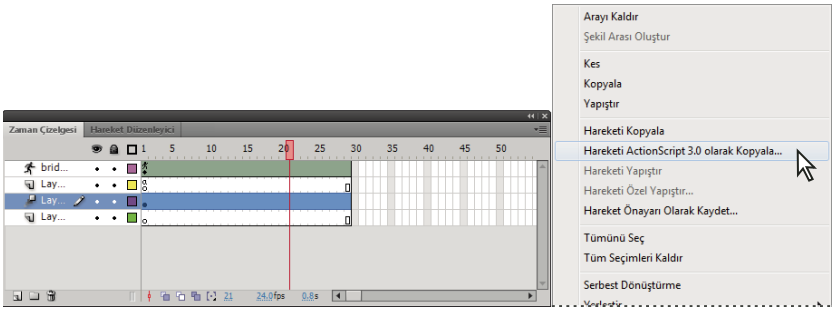
Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Ara doldurma, görüntüleme nesnesini bir zaman çizelgesindeki iki karede farklı durumlarda gösteren ara kareler oluşturur. Birinci karedeki görüntünün, yumuşak şekilde ikinci karedeki görüntüye dönüşmesi görünümünü oluşturur. Ara harekette, görünümdeki değişiklik genellikle görüntüleme nesnesinin konumunun değişmesini ve böylece hareket oluşturulmasını içerir. Ara hareket, görüntüleme nesnesinin yeniden konumlandırılmasına ek olarak, görüntüleme nesnesini döndürebilir, eğriltebilir, yeniden boyutlandırabilir veya nesneye filtreler uygulayabilir.

Görüntüleme nesnesini zaman çizelgesi boyunca anahtar kareler arasında taşıyarak Flash'ta bir ara hareket oluşturun. Flash, bir dosyada kopyalayıp kaydedebildiğiniz ara doldurmayı açıklayan ActionScript kodunu otomatik olarak oluşturur. Hareket arası oluşturma hakkında bilgi almak için *Flash Professional'ı Kullanma* kısmında Hareket Araları bölümüne bakın.

Flash'ta Hareketi ActionScript 3.0 olarak Kopyala komutuna iki şekilde erişebilirsiniz. Birincisi, sahne alanındaki ara doldurma bağlam menüsünden gerçekleşir:

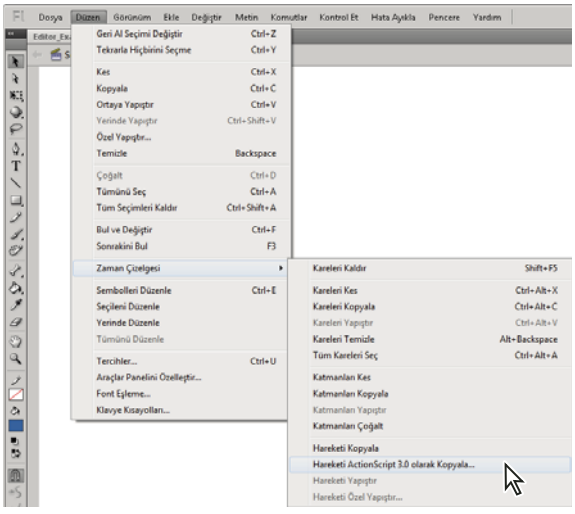
1. Sahne alanında ara hareketi seçin.
2. Sağ tıklatın (Windows) veya Control-tuşuna basıp tıklatın (Macintosh).
3. Hareketi ActionScript 3.0 olarak Kopyala seçeneğini belirleyin . . .



İkincisi, doğrudan Flash Düzen menüsünden komutun seçilmesidir:

1. Sahne alanında ara hareketi seçin.

2 Düzenle > Zaman Çizelgesi > Hareketi ActionScript 3.0 Olarak Kopyala seçeneklerini belirleyin.



Komut dosyasını kopyaladıktan sonra bir dosyaya yapıştırın ve kaydedin.

Ara hareket oluşturup kopyaladıktan ve komut dosyasını kaydettikten sonra, kendi dinamik ActionScript tabanlı animasyonunuzda bunu olduğu gibi kullanabilir veya değiştirebilirsiniz.

Ara hareket komut dosyalarını birleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Flash'tan kopyaladığınız ActionScript kodundaki üstbilgi, ara hareketi desteklemek için gerekli tüm modülleri listeler.

Ara hareket sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Başlıca ara hareket sınıfları, `fl.motion` paketindeki AnimatorFactory, MotionBase ve Motion sınıflarıdır. Ara hareketin işlediği özelliklere bağlı olarak ek sınıflara ihtiyacınız olabilir. Örneğin, ara hareket görüntüleme nesnesini dönüştürürse veya döndürürse, uygun `flash.geom` sınıflarını içe aktarın. Filtre uygularsa, `flash.filter` sınıflarını içe aktarın. ActionScript'te, ara hareket, Motion sınıfının bir örneğidir. Motion sınıfı, görsel bir nesneye uygulanabilen anahtar kare animasyon sırasını saklar. Animasyon verileri; konum, ölçek, döndürme, eğriltme, renk, filtreler ve hareket hızı öğelerini kapsar.

Aşağıdaki ActionScript, örnek adı `Symbol1_2` olan bir görüntüleme nesnesine animasyon uygulamak için Flash'ta oluşturulmuş bir ara hareketten kopyalanmıştır. Bu, `__motion_Symbol1_2` adında bir MotionBase nesnesi için değişken bildirir. MotionBase sınıfı, Motion sınıfının üst öğesidir.

```
var __motion_Symbol1_2:MotionBase;
```

Komut dosyası daha sonra Motion nesnesini oluşturur:

```
__motion_Symbol1_2 = new Motion();
```

Motion nesnesi adları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Önceki durumda, Flash uygulaması Motion nesnesi için otomatik olarak `__motion_Symbol1_2` adını oluşturmuştu. Bu, görüntüleme nesnesi adına `__motion_` önekini ekledi. Böylece, otomatik olarak oluşturulan ad, Flash'ta ara hareketin hedef nesnesinin örnek adını esas alır. Motion nesnesinin `duration` özelliği, hareket arasındaki toplam kare sayısını belirtir:

```
__motion_Symbol1_2.duration = 200;
```

Ara hareketinin kopyalandığı görüntüleme nesnesi örneğinin bir örnek adı yoksa, Flash uygulaması varsayılan olarak bu görüntüleme nesnesi örneğini otomatik şekilde adlandırır.

Flash tarafından oluşturulan ActionScript'i kendi animasyonunuzda yeniden kullandığınızda, Flash uygulamasının ara için otomatik olarak oluşturduğu adı saklayabilirsiniz veya onun yerine başka bir ad verebilirsiniz. Ara adını değiştirirseniz, komut dosyasının tamamında bunu değiştirdiğinizden emin olun.

Alternatif olarak, Flash'ta ara hareketin hedef nesnesine istediğiniz bir adı atayabilirsiniz. Daha sonra ara hareketi oluşturun ve komut dosyasını kopyalayın. Hangi adlandırma yaklaşımını kullanırsanız kullanın, ActionScript kodundaki her Motion nesnesinin benzersiz bir ada sahip olduğundan emin olun.

Animasyonu açıklama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

MotionBase sınıfının `addPropertyArray()` yöntemi, her arası doldurulmuş özelliği açıklamak için bir değerler dizisi ekler.

Potansiyel olarak dizi, ara hareketteki her anahtar kare için tek bir dizi öğesi içerir. Genellikle bu dizilerin bazıları, ara hareketteki toplam anahtar kare sayısından daha az öğe içerir. Dizideki son değer, kalan karelerde değişmediğinde bu durum gerçekleşir.

Dizi argümanının uzunluğu, Motion nesnesinin `duration` özelliğinden daha büyükse, `addPropertyArray()` öğesi, `duration` özelliğinin değerini uygun şekilde ayarlar. Önceden eklenmiş özellikler için anahtar kareler eklemez. Yeni eklenen anahtar kareler, animasyonun fazladan kareleri için kalıcı olabilir.

Motion nesnesinin `x` ve `y` özellikleri, animasyon çalıştırılırken arası doldurulmuş nesnenin değişen konumunu açıklar. Bu koordinatlar, görüntüleme nesnesinin konumu değiştiğinde her anahtar karede değişme olasılığı en yüksek olan değerlerdir. `addPropertyArray()` yöntemiyle ek hareket özellikleri ekleyebilirsiniz. Örneğin, arası doldurulmuş nesne yeniden boyutlandırılırsa, `scaleX` ve `scaleY` değerlerini ekleyin. Eğriltilirse, `scewX` ve `skewY` değerlerini ekleyin. Döndürülürse, `rotationConcat` özelliğini ekleyin.

Şu ara doldurma özelliklerini tanımlamak için, `addPropertyArray()` yöntemini kullanın:

`x`	üst öğesinin koordinat alanında nesnenin dönüştürme noktasının yatay konumu
`y`	üst öğesinin koordinat alanında nesnenin dönüştürme noktasının dikey konumu
`z`	üst öğesinin koordinat alanında nesnenin dönüştürme noktasının derinlik (z ekseni) konumu
`scaleX`	dönüştürme noktasından uygulandığı haliyle nesnenin yüzdesi olarak yatay ölçek
`scaleY`	dönüştürme noktasından uygulandığı haliyle nesnenin yüzdesi olarak dikey ölçek

skewX	dönüştürme noktasından uygulandığı haliyle derece cinsinden nesnenin yatay eğriltme açısı
skewY	dönüştürme noktasından uygulandığı haliyle derece cinsinden nesnenin dikey eğriltme açısı
rotationX	nesnenin orijinal yönlendirmesinden x ekseni boyunca dönüşü
rotationY	nesnenin orijinal yönlendirmesinden y ekseni boyunca dönüşü
rotationConcat	dönüştürme noktasından uygulandığı haliyle önceki yönlendirmeye göre hareketteki nesnenin dönüş (z ekseni) değerleri
useRotationConcat	Ayarlanırsa, addPropertyArray() öğesi hareket için veri sağladığında hedef nesnenin dönmesine neden olur
blendMode	Nesnenin renkleri ile altındaki grafiklerin karışımını belirten BlendMode sınıfı değeri
matrix3D	Anahtar kare için varsa, matrix3D özelliği; 3B ara doldurmalar için kullanılır; kullanılırsa, önceki dönüştürme özelliklerinin tümü yoksayılır
rotationZ	nesnenin orijinal yönlendirmesinden 3B üst konteynerine göre derece cinsinden z ekseni dönüşü; rotationConcat yerine 3B ara doldurmalar için kullanılır

Otomatik olarak oluşturulan komut dosyasına eklenen özellikler, Flash'ta ara harekete atanmış özelliklere bağlıdır. Kendi komut dosyası sürümünüzü özelleştirirken bu özelliklerden bazılarını ekleyebilir, kaldırabilir veya değiştirebilirsiniz.

Şu kod, __motion_Wheel adında bir ara hareketin özelliklerine değerler atar. Bu durumda, arası doldurulmuş görüntüleme nesnesinin konumu değişmez ancak söz konusu görüntüleme nesnesi ara harekette 29 kare boyunca yerinde döner. rotationConcat dizisine atanan birden çok değer, dönüşü tanımlar. Bu ara hareketin diğer özellik değerleri değişmez.

```
__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"]);
```

Sonraki örnekte, Leaf_1 adındaki görüntüleme nesnesi sahne alanı boyunca hareket eder. Bu nesnenin x ve y özelliği dizileri, animasyonun 100 karesinin her biri için farklı değerler içerir. Ayrıca, nesne sahne alanı boyunca hareket ederken z ekseninde döner. rotationZ özellik dizisindeki birden çok öğe, dönüşü belirler.

```
__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,
```

Filtreler dizisini başlatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

`initFilters()` yöntemi filtreleri başlatır. Bunun birinci argümanı, görüntüleme nesnesine uygulanmış tüm filtrelerin tam nitelendirilmiş sınıf adlarından oluşan bir dizidir. Bu filtre adları dizisi, Flash'ta ara hareketin filtreler listesinden oluşturulur. Komut dosyası kopyanızda, `flash.filters` paketindeki herhangi bir filtreyi kaldırabilir veya bu diziye ekleyebilirsiniz. Şu çağrı, hedef görüntüleme nesnesi için filtreler listesini başlatır. `DropShadowFilter`, `GlowFilter` ve `BevelFilter` öğesini uygular ve Motion nesnesindeki anahtar karelerin her birine listeyi kopyalar.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

Filtreler ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

`addFilterPropertyArray()` yöntemi, şu argümanlarla başlatılmış bir filtrenin özelliklerini açıklar:

1. Birinci argüman, filtreyi dizine göre tanımlar. Dizin, `initFilters()` öğesine yapılan önceki çağrıda iletilen filtre sınıfı adları dizisinde filtre adının konumunu ifade eder.
2. İkinci argüman, her anahtar karede o filtre için saklanacak filtre özelliğidir.
3. Üçüncü argüman, belirtilen filtre özelliğinin değeridir.

`initFilters()` öğesine yapılan önceki çağrı dikkate alındığında, `addFilterPropertyArray()` öğesine yapılan şu çağrılar, `DropShadowFilter` öğesinin `blurX` ve `blurY` özelliklerine 5 değerini atar. `DropShadowFilter`, başlatılan filtreler dizisindeki birinci (dizin 0) öğesidir:

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

Sonraki üç çağrı, başlatılan filtre dizisinin ikinci öğesi (dizin 1) olan `GlowFilter` öğesinin kalite, alfa ve renk özelliklerine değerler atar:

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

Sonraki dört çağrı, başlatılan filtreler dizisindeki `BevelFilter` öğesinin `shadowAlpha`, `shadowColor`, `highlightAlpha` ve `highlightColor` öğelerine değerler atar:

```
__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]);
```

ColorMatrixFilter ile renk ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

`ColorMatrixFilter` öğesi başlatıldıktan sonra, ara eklenmiş görüntüleme nesnesinin parlaklık, kontrast, doygunluk ve ton özelliklerini ayarlamak için uygun `AdjustColor` özelliklerini ayarlayabilirsiniz. Genelde `AdjustColor` filtresi, Flash'ta bir ara hareket oluşturulduğu zaman uygulanır; ona ActionScript'in size ait olan kopyasında ince ayar yapabilirsiniz. Şu örnek, hareket ederken görüntüleme nesnesinin ton ve doygunluk özelliklerini dönüştürür.

```
__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);
```

Ara hareketi görüntüleme nesneleriyle ilişkilendirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü, Flash CS3 veya üstünü gerektirir

Son görev, ara hareketin işlediği görüntüleme nesnesi veya nesneleriyle ilişkilendirilmesidir.

AnimatorFactory sınıfı, ara hareket ile hedef görüntüleme nesneleri arasında ilişkilendirmeyi yönetir. AnimatorFactory yapıcısına yapılan argüman, Motion nesnesidir:

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

Hedef görüntüleme nesnesini ara hareketiyle ilişkilendirmek için AnimatorFactory sınıfının `addTarget()` yöntemini kullanın. Flash yorumlarından kopyalanan ActionScript, `addTarget()` satırının yorumunu kaldırır ve bir örnek adı belirtmez:

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

Kopyanızda, ara hareketle ilişkilendirilecek görüntü nesnesini belirtin. Şu örnekte, hedefler `greenWheel` ve `redWheel` olarak belirtilir:

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

`addTarget()` öğesine yapılan birden çok çağrıyı kullanarak, aynı ara harekete sahip birden çok görüntüleme nesnesini ilişkilendirebilirsiniz.

Bölüm 18: Ters kinematikle çalışma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

Ters kinematik (IK), gerçekçi hareket oluşturulmasına yönelik önemli bir tekniktir.

IK, IK armatürü adı verilen bağlanmış bölümlerin bir zinciri içinde koordinasyonlu hareketler oluşturmanıza olanak sağlar, böylece parçalar gerçekçi şekilde birlikte hareket eder. Armatürün parçalarını kemikler ve eklemler oluşturur. Armatürün bitiş noktası verildiğinde, IK her eklemin bu bitiş noktasına ulaşması için gerekli açıyı hesaplar.

Bu açıların manuel olarak hesaplanması sizin için zor olabilir. Bu özelliğin güzelliği, Adobe® Flash® Professional'ı kullanarak etkileşimli şekilde armatürler oluşturabilmenizdir. Daha sonra ActionScript kullanarak bunlara animasyon uygularsınız. Flash Professional'a dahil edilen TK motoru, armatürün hareketlerini açıklamak için hesaplamalar gerçekleştirir. ActionScript kodunuzda hareketi belirli parametrelerle sınırlayabilirsiniz.

TK'nin Flash Professional CS5 sürümüne yeni olan konsept, genellikle en son teknolojiye sahip animasyon uygulamalarıyla ilişkili olan eklem yayıdır. Yeni dinamik Fizik Motoru ile kullanılan bu özellik canlı hareketleri yapılandırmanıza izin verir. Ayrıca, bu efekt hem çalışma zamanında hem de geliştirme sırasında görünür durumdadır.

Ters kinematik armatürleri oluşturmak için, Flash Professional lisansına sahip olmanız gerekir.

Daha fazla Yardım konusu

fl.ik paketi

Ters Kinematik Temelleri

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

Ters kinematik (IK), gerçeğe uygun olarak birbirine göre hareket edecek şekilde parçaları bağlayarak gerçekçi animasyonlar oluşturmanıza olanak sağlar.

Örneğin, IK'yi kullanarak, istediğiniz pozu elde etmek için bacaktaki gerekli eklem hareketlerini gerçekleştirip bir bacağı belirli bir konuma taşıyabilirsiniz. IK, IK armatürü adı verilen bir yapıda zincirlenmiş kemikler çerçevesini kullanır. `fl.ik` paketi, doğal harekete benzer animasyonlar oluşturmanıza yardımcı olur. IK algoritmasının ardındaki fizik kuralları hakkında çok bilgi sahibi olmanız gerekmeden kusursuz şekilde birden çok IK armatürüne animasyon uygulamanızı sağlar.

Flash Professional ile yardımcı eklem ve bağlantılar içeren TK armatürü oluşturun. Böylece çalışma zamanında bunlara animasyon uygulamak için IK sınıflarına erişebilirsiniz.

TK armatürünün nasıl oluşturulduğuyla ilgili ayrıntılı talimatlar için *Flash Professional'ı Kullanma* kısmında Ters kinematikleri kullanma bölümüne bakın.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi bu özellikle ilgili önemli terimler içerir:

Armatür Gerçekçi hareketi simüle etmek için bilgisayar animasyonunda kullanılan ve eklem ve bağlantılardan oluşan bir kinematik zinciri.

Eklem Hayvan iskeletindeki kemiğe benzer şekilde armatürdeki sert bir parça.

Ters Kinematik (TK) Kinematik zinciri veya armatür adı verilen eklemli esnek nesne parametrelerini belirleme işlemi.

Eklem Eklemlerin hareket etmesini sağlamak için oluşturulmuş, iki kemiğin birbirine temas ettiği konum; bu, bir hayvandaki bağlantıya benzer.

Fizik Motoru Animasyona gerçekçi eylemler sağlamak için fizikle ilgili algoritmalardan oluşan bir paket kullanılmıştır.

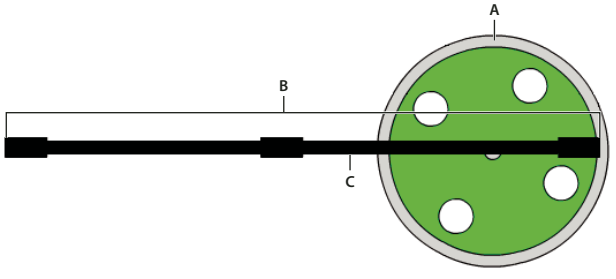
Yay Üst iskelet taşındığında ve sonra zaman içinde artımlı olarak kaybolduğunda bir iskeletin taşıma ve tepki verme kalitesidir.

TK Armatürlerine Animasyon Uygulamaya Genel Bakış

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

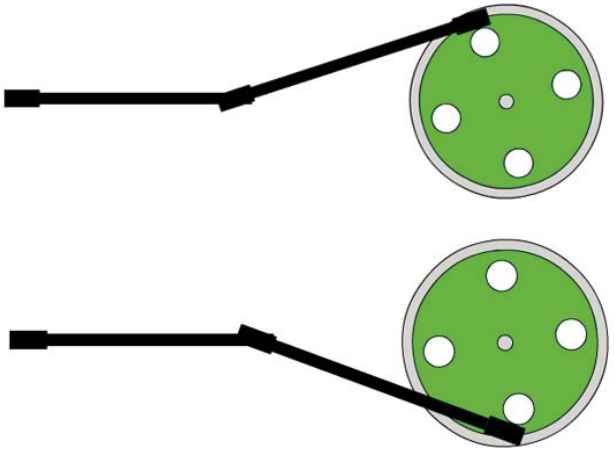
Flash Professional'da bir TK armatürü oluşturduktan sonra, çalışma zamanında bu armatürün hareketini sınırlamak, olaylarını izlemek ve armatüre animasyon uygulamak için `fl.ik` sınıflarını kullanın.

Aşağıdaki şekilde, Wheel adında bir film klibi gösterilmektedir. Aks, `Axle` adında bir IKArmature örneğidir. IKMover sınıfı, tekerlek dönüşüyle eşzamanlı olarak armatürü hareket ettirir. Armatürdeki IKBone, `ikBone2` öğesi, kuyruk ekleminde tekerleğe bağlıdır.



A. *Wheel* ***B.*** *Axle* ***C.*** *ikBone2*

Çalışma zamanında, tekerlek "Animasyonu açıklama" sayfa 340 bölümünde anlatılan `__motion_Wheel` hareket arasıyla bağlantılı olarak döner. IKMover nesnesi, aksın hareketini başlatır ve kontrol eder. Aşağıdaki resimde, dönüşteki farklı karelerde dönen tekerleğe eklenen aks armatürünün iki anlık görüntüsü gösterilmektedir.



Çalışma zamanında, şu ActionScript:

- Armatür ve bileşenleri hakkında bilgi edinir
- Bir IKMover nesnesini başlatır
- Aksı, tekerleğin dönüşüyle birlikte hareket ettiriri

```
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);
    }
}
```

Aksı taşımak için kullanılan IK sınıfları şunlardır:

- IKArmature: eklem ve bağlantı noktalarından oluşan bir ağaç yapısı niteliğindeki armatürü açıklar; Flash Professional ile oluşturulması gerekir
- IKManager: belgedeki tüm TK armatürlerinin kap sınıfı; Flash Professional ile oluşturulması gerekir
- IKBone: IK armatürünün bir parçası
- IKJoint: iki IK kemiği arasındaki bağlantı
- IKMover: armatürlerin IK hareketini başlatır ve kontrol eder

Bu sınıflarla ilgili eksiksiz ve detaylı açıklamalar için bkz. ik package.

TK armatürü hakkında bilgi alma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

Öncelikle, taşımak istediğiniz parçaları oluşturan armatür, kemik ve eklem değişkenlerini bildirin.

Aşağıdaki kod, Axle armatürünün değerini `tree` IKArmature değişkenine atamak için IKManager sınıfının `getArmatureByName()` yöntemini kullanır. Axle armatürü önceden Flash Professional ile oluşturulurdu.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

Aynı şekilde, aşağıdaki kod `ikBone2` kemiğinin IKBone değişkenine atamak için IKArmature sınıfının `getBoneByName()` yöntemini kullanır.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

`ikBone2` kemiğinin kuyruk eklemi, armatürün dönen tekerleğe eklenen bölümüdür.

Aşağıdaki satır, `endEffector` değişkenini bildirir ve bu değişkeni `ikBone2` kemiğinin `tailjoint` özelliğine atar:

```
var endEffector:IKJoint = home.tailjoint;
```

`pos` değişkeni, `endEffector` ekleminin geçerli konumunu saklayan noktadır.

```
var pos:Point = endEffector.position;
```

Bu örnekte, `pos` öğesi, aksın sonundaki eklemin tekerleğe bağlandığı konumudur. Bu değişkenin orijinal değeri, IKJoint öğesinin `position` özelliğinden elde edilir.

IK Taşıyıcısını Başlatma ve IK Taşıyıcısının Hareketini Sınırlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

IKMover sınıfının bir örneği aksı hareket ettirir.

Aşağıdaki satır, `ik` IKMover nesnesini başlatarak taşınacak öğeyi ve hareketin başlangıç noktasını yapıcısına iletir:

```
var ik:IKMover = new IKMover(endEffector, pos);
```

IKMover sınıfının özellikleri, bir armatürün hareketini sınırlamanıza olanak sağlar. Mesafe, yineleme ve hareket süresini esas alarak hareketi sınırlayabilirsiniz.

Aşağıdaki özellik çiftleri bu sınırlamaları zorlar. Çiftler, hareketin sınırlanmış olup olmadığını ve sınırlamayı belirten bir tam sayıyı gösteren Boolean değerini içerir:

Boolean özelliği	Tamsayı özelliği	Sınırı ayarla
`limitByDistance:Boolean`	`distanceLimit:int`	Her yineleme için IK motorunun hareket edeceği maksimum mesafeyi piksel cinsinden belirtir.
`limitByIteration:Boolean`	`iterationLimit:int`	Her hareket için IK motorunun gerçekleştireceği maksimum yineleme sayısını belirtir.
`limitByTime:Boolean`	`timeLimit:int`	Hareketi gerçekleştirmesi için IK motoruna ayrılan maksimum süreyi milisaniye cinsinden belirtir.

Varsayılan olarak, tüm Boolean değerleri `false` değerine ayarlıdır, bundan dolayı hareket siz bir Boolean değerini `true` olacak şekilde açıkça ayarlamadıkça sınırlanmaz. Bir sınırı zorlamak için, uygun özelliği `true` değerine ayarlayın ve ardından karşılık gelen tam sayı özelliği için bir değer belirleyin. Karşılık gelen Boolean özelliğini ayarlamadan sınırlamayı bir değere ayarlarsanız, sınırlama yok sayılır. Bu durumda, IK motoru IKMover öğesinin başka bir sınırına veya hedef konumuna ulaşılıncaya kadar hareket etmeye devam eder.

Aşağıdaki örnekte, armatür hareketinin maksimum mesafesi her yineleme için 0,1 piksel olarak ayarlanmıştır. Her hareket için maksimum yineleme sayısı on olarak ayarlanmıştır.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

Bir IK Armatürünü hareket ettirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

IKMover, tekerleğin olay dinleyicisinin içinde aksı hareket ettirir. Tekerleğin her enterFrame olayında, armatür için yeni bir hedef konum hesaplanır. IKMover, `moveTo()` yöntemini kullanarak kuyruk eklemini hedef konumuna veya `limitByDistance`, `limitByIteration` ve `limitByTime` özellikleri tarafından ayarlanan sınırlamalar içinde gidebildiği son noktaya taşır.

```
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);
    }
}
```

Yayları Kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS5 veya üstünü gerektirir

Flash Professional CS5'teki ters kinematik, eklem yayını destekler. Eklem yayı, geliştirme sırasında ayarlanabilir ve eklem yayı nitelikleri çalışma zamanında eklenebilir veya değiştirilebilir. Yay, eklemin ve bağlantı yerlerinin bir özelliğidir. İki niteliği vardır: yayın miktarını ayarlayan `IKJoint.springStrength` ve kuvvet değerine direnç ekleyen ve yayın aşınma oranını değiştiren `IKJoint.springDamping`.

Yay kuvveti, varsayılan 0 (çok sıkı) ile 100 (çok gevşek ve fizik tarafından kontrol edilir) arasında yüzde cinsinden bir değerdir. Eklemler, bağlantı yerlerinin hareketine yay tepkimesi gösterir. Etkinleştirilmiş başka bir çeviri (dönüş, x veya y) yoktur; yay ayarlarının hiçbir etkisi olmaz.

Yay sönümlemesi varsayılan 0 (dirençsiz) ile 100 (aşırı derecede sönümlü) arasında yüzde cinsinden bir değerdir. Sönüm, bir eklemin ilk hareketi ve dinlenme konumuna dönüşü arasında geçen zaman miktarını değiştirir.

`IKArmature.springsEnabled` özelliğini kontrol ederek bir IKArmature nesnesi için yayların etkin olup olmadığını kontrol edebilirsiniz. Diğer yay özellikleri ve yöntemleri ayrı IKJoint nesnelerine aittir. Bir bağlantı noktası x ve y eksenleri boyunca açısal dönüş ve çeviri açısından etkinleştirilebilir. Bir döndürme bağlantı noktasının yay açısını `IKJoint.setSpringAngle` ve çevirme bağlantı noktasının yay konumunu `IKJoint.setSpringPt` ile konumlandırabilirsiniz.

Bu örnek ada göre bir eklem seçer ve tailJoint öğesini tanımlar. Kod, yayların etkinleştirilip etkinleştirilmediğini anlamak için üst armatürü test eder ve ardından bağlantı yeri için yay özelliklerini belirler.

```
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
    }
}
```

TK Olaylarını Kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü, Flash CS4 veya üstünü gerektirir

IKEvent sınıfı, IK Olayları hakkında bilgi içeren bir olay nesnesi oluşturmanıza olanak sağlar. IKEvent bilgileri, belirtilen süre, mesafe veya yineleme sınırı aşıldığından sonlandırılmış olan hareketi açıklar.

Aşağıdaki kod, zaman sınırı olaylarının izlenmesine yönelik bir olay dinleyicisini ve işleyicisini gösterir. Bu olay işleyicisi, IKMover öğesinin zaman sınırı aşıldığında ortaya çıkan bir olayın süre, mesafe, yineleme sayısı ve eklem özelliklerini bildirir.

```
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);
}
```

Bölüm 19: Üç boyutlu (3B) çalışma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player ve AIR çalışma zamanları, 3B grafikleri iki şekilde destekler: Üç boyutlu görüntüleme nesnelerini Flash görüntüleme listesinde kullanabilirsiniz. Bu, üç boyutlu efektleri Flash içeriğine ve düşük çokgen sayılı nesnelere yönelik olarak eklemek için uygundur. Flash Player 11 ve AIR 3 veya üstü bir sürümde, Stage3D API'sini kullanarak karmaşık 3B sahneleri oluşturabilirsiniz.

Stage3D görünüm penceresi bir görüntüleme nesnesi değildir. Aksine 3B grafikler, Flash görüntüleme listesi altında (ve herhangi bir StageVideo görünüm penceresi düzlemi üzerinde) görüntülenen bir görünüm penceresine oluşturulur. Bir sahne oluşturmak için Flash DisplayObject sınıflarını kullanmak yerine programlanabilir bir 3B ardışık düzeni (OpenGL ve Direct3D'ye benzer) kullanırsınız. Bu ardışık düzen, üçgen verilerini ve dokuları girdi olarak alır ve sağladığınız gölgelendirici programlarını kullanarak sahneyi oluşturur. Donanım hızlandırma, desteklenen sürücülerle uyumlu bir grafik işleme birimi (GPU) istemci bilgisayarında kullanılabilir olduğunda kullanılır.

Stage3D, çok düşük düzeyli bir API sağlamaktadır. Bir uygulamada, Stage3D destekleyen bir 3B altyapısı kullanmanız önerilir. Kendi çerçevenizi oluşturabilirsiniz veya zaten kullanılabilir olan çeşitli ticari ve açık kaynak çerçevelerinden birini kullanabilirsiniz.

Stage3D kullanarak 3B uygulamalar geliştirme ve kullanılabilir 3B altyapıları hakkında daha fazla bilgi için Flash Player Geliştirme Merkezi: Stage 3D alanını ziyaret edin.

3B görüntüleme nesnelerinin temelleri

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

İki boyutlu bir ekrana yansıtılan iki boyutlu (2B) nesne ile üç boyutlu (3B) nesne arasındaki ana fark, nesneye üçüncü bir boyutun eklenmesidir. Üçüncü boyut, nesnenin kullanıcının görüş açısından ileri ve geri hareket etmesini sağlar.

Bir görüntüleme nesnesinin `z` özelliğini açıkça sayısal bir değere ayarladığınızda, nesne otomatik olarak bir 3B dönüştürme matrisi oluşturur. O nesnenin 3B dönüştürme ayarlarını değiştirmek için bu matrisi değiştirebilirsiniz.

Ayrıca 3B dönüş, 2B dönüşten farklıdır. 2B'de, dönüş ekseni her zaman x/y düzlemine dikey gelir, başka bir deyişle z eksenindedir. 3B'de, dönüş ekseni x, y veya z eksenlerinden herhangi birinin etrafında olabilir. Görüntüleme nesnesinin dönüş ve ölçekleme özelliklerinin ayarlanması, görüntüleme nesnesinin 3B alanda hareket etmesini sağlar.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi 3 boyutlu grafikleri programlarken karşılaşacağınız önemli terimleri içerir:

Perspektif 2B düzlemde, derinlik ve mesafe izlenimi vermek için ufuk noktasında birleşen paralel çizgilerin temsili.

İzdüşüm Daha yüksek boyutlu bir nesnenin 2B görüntü üretimi; 3B izdüşüm, 3B noktaları 2B düzlemle eşler.

Döndürme Nesnedeki her noktayı dairesel şekilde hareket ettirerek nesnenin yönlendirmesini (ve genellikle de konumunu) değiştirme.

Dönüştürme Çevirme, döndürme, ölçekleme, eğme veya bu eylemlerin birleşimi yoluyla 3B noktaları ya da nokta kümelerini değiştirme.

Çevirme Nesnedeki her noktayı aynı yönde aynı miktarda hareket ettirerek nesnenin konumunu değiştirme.

Ufuk noktası Doğrusal perspektifte temsil edildiğinde, sonraki paralel çizgilerin buluşuyor gibi göründüğü nokta.

Vektör 3B vektör, x, y ve z Kartezyen koordinatlarını kullanarak üç boyutlu alandaki bir noktayı veya konumu temsil eder.

Köşe Bir köşe noktası.

Dokulandırılmış kafes 3B alanda bir nesneyi tanımlayan herhangi bir nokta.

UV eşleme 3B yüzeye bir doku veya bitmap uygulama yolu. UV eşleme, bir görüntüdeki koordinatlara yatay (U) eksen ve dikey (V) eksen yüzdeleri olarak değerler atar.

T değeri Nesne geçerli görüş açısına doğru veya bu görüş açısından başka yöne doğru hareket ettikçe 3B nesnenin boyutunun belirlenmesine yönelik ölçekleme faktörü.

Ayırma Belirli bir sargı ile yüzeyler oluşturma veya oluşturmama. Yüzey kaldırma işlevini kullanarak, geçerli görüş açısı için görünür olmayan yüzeyleri gizleyebilirsiniz.

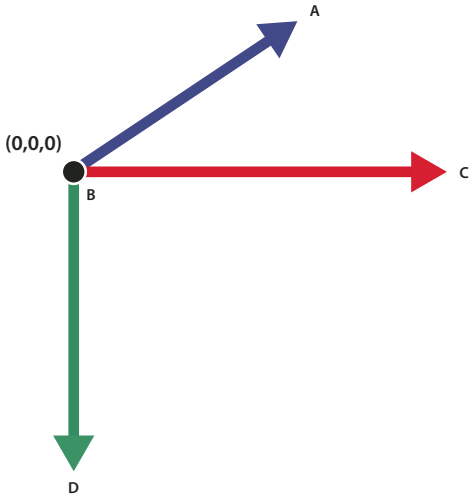
Flash Player ve AIR çalışma zamanında 3B görüntüleme nesnelerini anlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player 10'dan önceki Flash Player sürümlerinde ve Adobe AIR 1.5'ten önceki Adobe AIR sürümlerinde, görüntüleme nesnelerinin, onları 2B düzleme konumlandırmaya yarayan, `x` ve `y` olmak üzere iki özelliği vardı. Flash Player 10 ve Adobe AIR 1.5'ten itibaren her ActionScript görüntüleme nesnesi, z ekseni boyunca nesneyi konumlandırmanızı sağlayan ve genellikle derinlik ve mesafe belirtmek için kullanılan bir `z` özelliğine sahiptir.

Flash Player 10 ve Adobe AIR 1.5, 3B efektler için destek sağlar. Ancak görüntüleme nesneleri yapısı gereği yassıdır. MovieClip nesnesi veya Sprite nesnesi gibi her görüntüleme nesnesi, sonuç olarak tek bir düzlemde iki boyutlu olarak kendisini oluşturur. 3B özellikler, bu düzlemsel nesneleri üç yönde de yerleştirmenize, taşımanıza, döndürmenize ve dönüştürmenize olanak sağlar. Bunlar ayrıca 3B nesneleri 2B görünümde yansıtabilmeniz için 3B noktaları yönetmenize ve 2B x, y koordinatlarına dönüştürmenize olanak sağlar. Bu özellikleri kullanarak çok çeşitli 3B deneyimlerini simüle edebilirsiniz.

ActionScript tarafından kullanılan 3B koordinat sistemi, diğer sistemlerden farklılık gösterir. ActionScript'te 2B koordinatları kullandığınızda, x ekseni boyunca sağa doğru hareket ettiğinizde x değeri artar ve y ekseni boyunca aşağı doğru hareket ettiğinizde de y değeri artar. 3B koordinat sistemi, bu kuralları korur ve görüş açısından hareket etikçe değeri artan bir z ekseni ekler.



ActionScript 3B koordinat sisteminde x, y ve z eksenlerinin pozitif yönleri.
A. *+ Z ekseni* ***B.*** *Kaynak* ***C.*** *+X ekseni* ***D.*** *+Y ekseni*

Not: *Flash Player ve AIR uygulamasının her zaman katmanlarda 3B'yi temsil ettiğini unutmayın. Başka bir deyişle, görüntüleme listesinde A nesnesi B nesnesinin önündeyse, Flash Player veya AIR uygulaması iki nesnenin z ekseni değerlerini dikkate almaksızın her zaman A'yı B'nin önünde oluşturur. Görüntüleme listesi sıralaması ile z ekseni sıralaması arasındaki bu çakışmayı çözümlemek için, 3B görüntüleme nesnelerinin katmanlarını kaydetmek ve yeniden sıralamak için* `transform.getRelativeMatrix 3D()` *yöntemini kullanın. Daha fazla bilgi için bkz. "Görüntünün yeniden sıralanması için Matrix3D nesnelerini kullanma" sayfa 364.*

Şu ActionScript sınıfları, yeni 3B ile ilgili özellikleri destekler:

1. flash.display.DisplayObject sınıfı, `z` özelliğini ve 3B alanda görüntüleme nesnelerinin işlenmesine yönelik yeni dönüş ve ölçekleme özelliklerini içerir. `DisplayObject.local3DToGlobal()` yöntemi, 3B geometriyi 2B düzleme yansıtmanın basit bir yolunu sunar.
2. flash.geom.Vector3D sınıfı, 3B noktaların yönetilmesine yönelik bir veri yapısı olarak kullanılabilir. Bu, vektör matematiğini de destekler.
3. flash.geom.Matrix3D sınıfı, 3B geometrisinin döndürme, ölçekleme ve çevirme gibi karmaşık dönüştürmelerini destekler.
4. flash.geom.PerspectiveProjection sınıfı, 3B geometrisinin 2B görünümüne eşlenmesine yönelik parametreleri denetler.

ActionScript'te 3B görüntülerin simüle edilmesine yönelik iki farklı yaklaşım mevcuttur:

1. 3B alanda düzlemsel nesneleri düzenleme ve bu nesnelere animasyon uygulama. Bu yaklaşım, x, y ve z özellikleri kullanılarak görüntüleme nesnelerine animasyon uygulanmasını veya DisplayObject sınıfı kullanılarak dönüş ve ölçekleme özelliklerinin ayarlanmasını içerir. DisplayObject.transform.matrix3D nesnesi kullanılarak daha karmaşık hareketler elde edilebilir. DisplayObject.transform.perspectiveProjection nesnesi, görüntüleme nesnelerinin 3B perspektifte çizilme şeklini özelleştirir. Öncelikli olarak düzlemlerden oluşan 3B nesnelere animasyon uygulamak istediğinizde bu yaklaşımı kullanın. Bu yaklaşımın örnekleri arasında, 3B görüntü galerileri veya 3B düzlemde düzenlenmiş 2B animasyon nesneleri yer alır.

2 3B geometrisinden 2B üçgenler oluşturma ve bu üçgenleri dokularla oluşturma. Bu yaklaşımı kullanmak için öncelikle 3B nesnelerle ilgili verileri tanımlayıp yönetmeniz ve sonra oluşturma işlemi için bu verileri 2B üçgenlere dönüştürmeniz gerekir. Bitmap dokular bu üçgenlere eşlenebilir ve sonra üçgenler `Graphics.drawTriangles()` yöntemi kullanılarak bir grafik nesnesine çizilir. Bu yaklaşımın örnekleri arasında, dosyadan 3B model verilerinin yüklenip ekranda modelin oluşturulması veya 3B bölgenin oluşturulup üçgen kafesleri olarak çizilmesi yer alır.

3B görüntüleme nesneleri oluşturma ve taşıma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

2B görüntüleme nesnesini 3B görüntüleme nesnesine dönüştürmek için, bu nesnenin `z` özelliğini sayısal bir değere ayarlayabilirsiniz. `z` özelliğine bir değer atadığınızda, görüntüleme nesnesi için yeni bir Transform nesnesi oluşturulur. `DisplayObject.rotationX` veya `DisplayObject.rotationY` özelliklerinin ayarlanmasıyla yeni bir Transform nesnesi oluşturulur. Transform nesnesi, görüntüleme nesnesinin 3B alanda temsil edilme şeklini yöneten bir `Matrix3D` özelliğini içerir.

Şu kod, "leaf" adındaki bir görüntüleme nesnesinin koordinatlarını ayarlar:

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

Yaprak öğesinin Transform nesnesinin `matrix3D` özelliğinde, bu değerlerden kaynaklanan özelliklerin yanı sıra bu değerleri de görebilirsiniz:

```
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);
```

Transform nesnesinin özellikleriyle ilgili bilgi için bkz. Transform sınıfı. Matrix3B nesnesinin özellikleriyle ilgili bilgi için bkz. Matrix3B sınıfı.

3B alanda nesneyi taşıma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`x, y, z` özelliklerini değiştirerek 3B alanda bir nesneyi taşıyabilirsiniz. `z` özelliğinin değerini değiştirdiğinizde, nesne izleyene yaklaşıyor veya uzaklaşıyor gibi görünür.

Şu kod, bir olaya yanıt olarak `z` özelliklerinin değerini değiştirerek iki elipsi z ekseni boyunca ileri geri hareket ettirir. `ellipse2`, `ellipse1` öğesinden daha hızlı hareket eder: her Frame olayında `ellipse1` öğesinin `z` özelliği 10'un katıyla artırılırken, bunun `z` özelliği 20'nin katıyla artırılır:

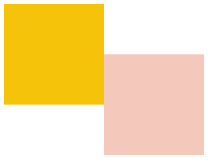
```
var depth:int = 1000;

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;
    }
}
```

3B alanda nesneyi döndürme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Nesnenin dönüş özelliklerini nasıl ayarladığınıza bağlı olarak, bir nesneyi üç farklı şekilde döndürebilirsiniz: `rotationX`, `rotationY` ve `rotationZ`.

Aşağıdaki şekil, döndürülmemiş iki kareyi gösterir:

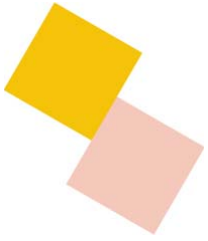


Sonraki şekil, y ekseninde döndürmek için karelerin kabının `rotationY` özelliğini artırdığınızda iki kareyi gösterir. İki karenin kabının veya üst görüntüleme nesnesinin döndürülmesiyle iki kare döndürülür:

```
container.rotationY += 10;
```

Sonraki şekil, karelerin kabının `rotationX` özelliğini ayarladığınızda ne olduğunu gösterir. Bu, x eksenindeki kareleri döndürür.

Sonraki şekil, karelerin kabının `rotationZ` özelliğini artırdığınızda ne olduğunu gösterir. Bu, z eksenindeki kareleri döndürür.



Görüntüleme nesnesi 3B alanda aynı anda hareket edip dönebilir.

3B nesneleri 2B görünümde yansıtma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`flash.geom` paketindeki PerspectiveProjection sınıfı, 3B alanda görüntüleme nesneleri hareket ettirilirken basit perspektif uygulanmasının basit bir yolunu sağlar.

3B alanınız için açıkça bir perspektif projeksiyonu oluşturmazsanız, 3B motor, kökte varolan varsayılan bir PerspectiveProjection nesnesini kullanır ve tüm alt öğeleriyle doldurulur.

PerspectiveProjection nesnesinin 3B alanı nasıl görüntülediğini tanımlayan üç özellik şunlardır:

- `fieldOfView`
- `projectionCenter`
- `focalLength`

`fieldOfView` ile `focalLength` birbirine bağlı olduğundan, bunlardan herhangi birinin değerinin değiştirilmesi otomatik olarak diğerinin de değerini değiştirir.

`fieldOfView` değeri verildiğinde `focalLength` öğesini hesaplamak için kullanılan formül şudur:

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

Genellikle, `fieldOfView` özelliğini açıkça değiştirirsiniz.

Görüş alanı

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

PerspectiveProjection sınıfının `fieldOfView` özelliğini işleyerek, izleyiciye yaklaşan 3B görüntüleme nesnesin daha büyük ve izleyiciden uzaklaşan nesnenin de daha küçük görünmesini sağlayabilirsiniz.

`fieldOfView` özelliği, perspektif projeksiyonunun kuvvetini belirleyen 0 ile 180 derece arasında bir açı belirtir. Bu değer arttıkça, z ekseni etrafında hareket eden görüntüleme nesnesine uygulanan deformasyon daha güçlü olur. Düşük bir `fieldOfView` değeri, çok küçük bir ölçeklemeye neden olur ve nesnelerin alanda yalnızca hafifçe geriye hareket ediyor gibi görünmesini sağlar. Yüksek bir `fieldOfView` değeri, daha fazla deformasyon ve daha büyük çapta bir hareket görünümü sağlar. 179.9999... derece maksimum değeri, oldukça ileri derecede balıkgözü kamera merceği efektiyle sonuçlanır. `fieldOfView` için maksimum değer 179.9999... ve minimum değer 0.00001... şeklindedir. Tam 0 ve 180, geçersiz değerlerdir.

Projeksiyon merkezi

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`projectionCenter` özelliği, perspektif projeksiyonunda ufuk noktasını temsil eder. Bu, sahne alanının sol üst köşesindeki varsayılan kayıt noktasına (0,0) uzaklık olarak uygulanır.

Bir nesne, izleyiciden uzağa hareket ediyormuş gibi göründükçe, ufuk noktasına doğru eğer ve sonunda gözden kaybolur. Sonsuz uzunlukta bir holü düşünün. Hole tepeden baktığınızda, duvarların kenarları holün aşağısındaki bir ufuk noktasında birleşir.

Ufuk noktası, sahne alanının merkezindeyse, hol, merkezdeki bir noktaya doğru kaybolur. `projectionCenter` özelliğinin varsayılan değeri, sahne alanının merkezidir. Örneğin, öğelerin sahne alanının solunda ve 3B alanın da sağda görüntülenmesini istiyorsanız, `projectionCenter` öğesini, sahne alanının sağındaki bir noktaya ayarlayarak bu noktayı 3B görüntüleme alanınızın ufuk noktası yapabilirsiniz.

Odak uzaklığı

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`focalLength` özelliği, görüş açısının başlangıç noktası (0,0,0) ile görüntüleme nesnesinin z eksenindeki konumu arasındaki mesafeyi temsil eder.

Odak uzaklığı, dar bir görünüme ve nesneler arasında sıkıştırılmış mesafelere sahip bir telefoto lensine benzer. Kısa bir odak uzaklığı, çok fazla deformasyonlu geniş bir görünüm elde etmenizi sağlayan geniş açılı lense benzer. Orta büyüklükte bir odak uzaklığı, gözle görünen boyuta yaklaştırır.

Genellikle görüntü nesnesi hareket ettikçe perspektif dönüştürmesi sırasında `focalLength` dinamik olarak yeniden hesaplanır ancak siz bunu açıkça da ayarlayabilirsiniz.

Varsayılan perspektif projeksiyonu değerleri

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Kökte oluşturulan varsayılan PerspectiveProjection nesnesi şu değerlere sahiptir:

- `fieldOfView: 55`
- `perspectiveCenter: stagewidth/2, stageHeight/2`
- `focalLength: stageWidth/ 2 * ( cos(fieldOfView/2) / sin(fieldOfView/2) )`

Kendi PerspectiveProjection nesnenizi oluşturmadığınızda kullanılan değerler bunlardır.

`projectionCenter` ve `fieldOfView` öğesini değiştirme amacıyla kendi PerspectiveProjection nesnenizi başlatabilirsiniz. Bu durumda, yeni oluşturulan nesnenin varsayılan değerleri, 500 x 500 varsayılan sahne alanı boyutu esas alınarak şunlardır:

- `fieldOfView: 55`
- `perspectiveCenter: 250,250`
- `focalLength: 480.24554443359375`

Örnek: Perspektif projeksiyonu

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Şu örnek, 3B alan oluşturmak için perspektif projeksiyonunun kullanılmasını gösterir. Bu, `projectionCenter` özelliği üzerinden ufuk noktasını ve alanın perspektif projeksiyonunu nasıl değiştirebileceğinizi gösterir. Bu değiştirme, 3B alanın beraberinde getirdiği deformasyon ile `focalLength` ve `fieldOfView` öğelerinin yeniden hesaplanmasını zorlar.

Bu örnek:

1. Artı imleci içeren bir daire olarak `center` adında bir hareketli grafik oluşturur
2. Kökün `transform` özelliğinin `perspectiveProjection` özelliğinin `projectionCenter` özelliğine `center` hareketli grafiğinin koordinatlarını atar
3. `center` nesnesinin konumunu takip edecek şekilde `projectionCenter` öğesini değiştiren işleyicileri çağıran fare olayları için olay dinleyicileri ekler
4. Perspektif alanının duvarlarını oluşturan akordeon stilinde dört kutu oluşturur

Bu örneği (ProjectionDragger.swf) test ettiğinizde, etraftaki daireyi farklı konumlara sürükleyin. Ufuk noktası daireyi takip eder ve bıraktığınız yerde kalır. Alanın genişlemesini çevreleyen ve projeksiyon merkezini sahne alanının merkezinden uzağa taşıdıkça deforme edilen kutuları izleyin.

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. ProjectionDragger uygulama dosyaları, Samples/ProjectionDragger klasöründedir.

```
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;
        }
    }
}
```

Daha karmaşık perspektif projeksiyonu için Matrix3D sınıfını kullanın.

Karmaşık 3B dönüştürmeler gerçekleştirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Matrix3D sınıfı, bir koordinat alanı içinde 3B noktaları dönüştürmenize veya 3B noktaları bir koordinat alanından diğerine eşlemenize olanak sağlar.

Matrix3D sınıfını kullanmak için matris matematiğini anlamanız gerekmez. En yaygın dönüştürme işlemlerinin çoğu, sınıfın yöntemleri kullanılarak işlenebilir. Matristeki her öğenin değerlerini açıkça ayarlama veya hesaplama konusunda endişelenmeniz gerekmez.

Görüntüleme nesnesinin `z` özelliğini sayısal bir değere ayarladıktan sonra, görüntüleme nesnesinin Transform nesnesinin Matrix3D özelliğini kullanarak, nesnenin dönüştürme matrisini alabilirsiniz:

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

Görüntüleme nesnesi üzerinde çevirme, döndürme, ölçekleme ve perspektif projeksiyonu gerçekleştirerek Matrix3D nesnesinin yöntemlerini kullanabilirsiniz.

3B noktaların yönetilmesine yönelik `x`, `y` ve `z` özellikleriyle Vector 3D sınıfını kullanın. Bu, bir yön ve büyüklüğe sahip olan, fizikteki bir uzamsal vektörü temsil eder. Vector3D sınıfının yöntemleri, ekleme, nokta üretimi ve çapraz ürün hesaplamaları gibi uzamsal vektörlerle yaygın hesaplamaları gerçekleştirmenizi sağlar.

Not: *Vector3D sınıfı, ActionScript Vector sınıfıyla ilgili değildir. Vector3D sınıfı, 3B noktaların tanımlanmasına ve işlenmesine yönelik özellikler ve yöntemler içerirken vector sınıfı, türlenmiş nesnelerin dizilerini destekler.*

Matrix3D nesneler oluşturma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`Matrix3D` nesnelerini oluşturmanın ve almanın üç ana yolu vardır:

- Yeni bir matris başlatmak için `Matrix3D()` yapıcı yöntemini kullanın. `Matrix3D()` yapıcısı, 16 sayısal değer içeren bir `Vector` nesnesini alır ve değerlerin her birini bir matris hücresine yerleştirir. Örneğin:

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- Görüntüleme nesnesinin `z` özelliğinin değerini ayarlayın. Daha sonra, o nesnenin `transform.matrix3D` özelliğinden dönüştürme matrisini alın.
- Kök görüntüleme nesnesindeki `perspectiveProjection.toMatrix3D()` yöntemini çağırarak sahne alanındaki 3B nesnelerin görüntülenmesini denetleyen Matrix3D nesnesini alın.

Birden çok 3B dönüştürmesi uygulama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Matrix3D nesnesi kullanarak bir defada çok sayıda 3B dönüştürme uygulayabilirsiniz. Örneğin, bir küpü döndürmek, ölçeklemek ve sonra taşımak isterseniz, küpün her bir ucuna üç ayrı dönüştürme uygulayabilirsiniz. Ancak tek bir Matrix3D nesnesinde birden çok dönüştürmeyi önceden hesaplamak ve sonra noktaların her birinde tek bir matris dönüştürmesi gerçekleştirmek çok daha etkilidir.

Not: *Matris dönüştürmelerinin uygulanma sırası önemlidir. Matris hesaplamaları komütatif değildir. Örneğin, önce bir döndürme ve sonra bir çevirme uygulanması, aynı çevirmenin önce ve aynı döndürmenin sonra uygulanmasından farklı bir sonuç verir.*

Şu örnek, birden çok 3B dönüştürme uygulanmasının iki yolunu gösterir.

```
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;
        }
    }
}
```

`doTransforms()` yönteminde, birinci kod bloğu, dikdörtgen şeklinin dönüşünü, ölçeklemesini ve konumunu değiştirmek için DisplayObject özelliklerini kullanır. İkinci kod bloğu ise, aynı dönüştürmeleri yapmak için Matrix3D sınıfının yöntemlerini kullanır.

`Matrix3D` yöntemlerinin kullanılmasının ana avantajı, tüm hesaplamaların ilk olarak matriste gerçekleştirilmesidir. Bunlar daha sonra `transform.matrix3D` özelliği ayarlandığında yalnızca bir defa görüntüleme nesnesine uygulanır. DisplayObject özelliklerinin ayarlanması, kaynak kodun okunmasını biraz daha kolaylaştırır. Ancak, döndürme veya ölçekleme özelliği her ayarlandığında, birden çok hesaplamaya neden olur ve birden çok görüntüleme nesnesi özelliğini değiştirir.

Kodunuz, görüntüleme nesnelerine birden çok defa aynı karmaşık dönüştürmeleri uygulayacaksa, Matrix3D nesnesini bir değişken olarak kaydedin ve sonra bunu art arda uygulayın.

Görüntünün yeniden sıralanması için Matrix3D nesnelerini kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Daha önce de belirtildiği gibi, görüntüleme listesindeki görüntüleme nesnelerinin katmanlama sırası, göreceli z eksenlerine bakılmaksızın görüntü katmanlama sırasını belirler. Animasyonunuz, görüntüleme nesnelerinin özelliklerini görüntüleme listesi sıralamasından farklı bir sıralamaya dönüştürürse, izleyici, z ekseni katmanlamasına karşılık gelmeyen görüntüleme nesnesini görebilir. Bu nedenle, izleyiciden uzaklaşıyor gibi görünen bir nesne, izleyiciye daha yakındaki bir nesnenin önünde görünebilir.

3B görüntüleme nesnelerinin katmanlamasının, nesnelerin göreceli derinliklerine karşılık geldiğinden emin olmak için, şunun gibi bir yaklaşım kullanın:

1. Alt 3B görüntüleme nesnelerinin göreceli `z-axes` öğelerini almak için, Transform nesnesinin `getRelativeMatrix 3D()` yöntemini kullanın.
2. Nesneleri görüntüleme listesinden kaldırmak için `removeChild()` yöntemini kullanın.
3. Görüntüleme nesnelerini, göreceli z ekseni değerlerini esas alarak sıralayın.
4. Alt öğeleri, ters sıralamayla geri görüntüleme listesine eklemek için `addChild()` yöntemini kullanın.

Bu yeniden sıralama, nesnelerinizin göreceli z eksenlerine göre görüntülenmesini sağlar.

Şu kod, bir 3B kutunun altı yüzünün doğru görüntülenmesini zorlar. Kutuya döndürmeler uygulandıktan sonra kutunun yüzlerini yeniden sıralar:

```
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);
    }
}
```

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Uygulama dosyaları, Samples/ReorderByZ klasöründedir.

3B efektler için üçgenleri kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

3B modeller, üçgenlerin bir alanda toplanmasıyla temsil edildiği için ActionScript'te `Graphics.drawTriangles()` yöntemini kullanarak bitmap dönüşümleri yapabilirsiniz. (Ancak, Flash Player ve AIR uygulaması derinlik arabelleğini desteklemez, bu nedenle görüntüleme nesneleri doğası gereği yassı veya 2B olmaya devam eder. Bu, "Flash Player ve AIR çalışma zamanında 3B görüntüleme nesnelerini anlama" sayfa 353 bölümünde bu açıklanmaktadır.) `Graphics.drawTriangles()` yöntemi, bir üçgen yolu çizmek için koordinatlar kümesini aldığından, `Graphics.drawPath()` yöntemine benzer.

`Graphics.drawPath()` öğesini kullanmaya alışmak için, bkz. "Yol Çizme" sayfa 239.

`Graphics.drawTriangles()` yöntemi, üçgen yolu için nokta konumlarını belirtmek üzere bir Vector.<Number> öğesi kullanır:

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number>
= null, culling:String = "none"):void
```

`drawTriangles()` öğesinin birinci parametresi 8`vertices` parametresi), zorunlu olan tek parametredir. Bu parametre, üçgenlerinizin çizildiği koordinatları tanımlayan sayıların bir vektörüdür. Her üç koordinat kümesi (altı sayı) bir üçgen yolunu temsil eder. Her üçgen üç koordinat çifti (üç tane x/y değerleri kümesi) gerektirdiğinden, `indices` parametresi olmadan, vektör uzunluğu her zaman altının bir faktörü olmalıdır. Örneğin:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

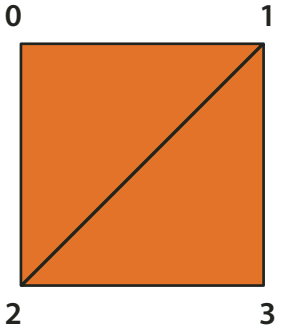
Bu üçgenlerin hiçbiri bir nokta paylaşmaz ancak paylaşsalardı, birden çok üçgen için `vertices` vektöründe değerleri yeniden kullanmak üzere ikinci `drawTriangles()` parametresi olan `indices` kullanılabilirdi.

`indices` parametresini kullanırken, `indices` değerlerinin doğrudan `vertices` dizi öğeleriyle ilgili olan dizinler değil, nokta dizinleri olmasına dikkat edin. Başka bir deyişle, `indices` tarafından tanımlandığı şekilde `vertices` vektöründeki bir dizin gerçekte 2'ye bölünen gerçek dizindir. `vertices` vektörünün üçüncü noktası için, söz konusu noktanın birinci sayısal değeri 4 vektör dizininde başlasa da, örneğin `indices` değeri olarak 2'yi kullanın.

Örneğin, `indices` parametresini kullanarak çapraz kenarı paylaşacak şekilde iki üçgeni birleştirin:

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

Şimdi iki üçgen kullanılarak bir kare çizilmiş olsa da, `vertices` vektöründe yalnızca dört noktanın belirtilmiş olduğuna dikkat edin. `indices` öğesi kullanılarak, iki üçgen tarafından paylaşılan iki nokta, her üçgen için yeniden kullanılır. Bu da, genel kesişme noktaları sayısını 6'dan (12 sayı) 4'e (8 sayı) düşürür:



vertices parametresi kullanılarak iki üçgenle çizilmiş bir kare

Bu teknik, çoğu noktanın birden çok üçgen tarafından paylaşıldığı daha büyük üçgen kafesleri için kullanışlıdır.

Üçgenlere tüm dolgular uygulanabilir. Dolgular, diğer şekillere olduğu gibi sonuçta elde edilen üçgene de uygulanır.

Bitmap'leri dönüştürme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bitmap dönüştürmeleri, üç boyutlu bir nesne üzerinde perspektif veya "doku" görünümü sağlar. Özel olarak, görüntünün ufuk noktasına yaklaştıkça daralıyor gibi görünmesi için ufuk noktasına doğru bir bitmap'i deforme edebilirsiniz. Veya, üç boyutlu bir nesneye için yüzey oluşturmak üzere iki boyutlu bitmap kullanarak o üç boyutlu nesne üzerinde doku veya "sargı" görünümü sağlayabilirsiniz.

Bir ufuk noktası ve bitmap ile sarılmış üç boyutlu bir nesne kullanan iki boyutlu yüzey.

UV eşleme

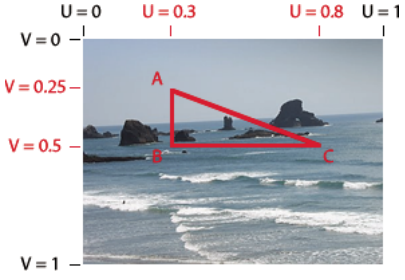
Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Dokularla çalışmaya başladıktan sonra `drawTriangles()` öğesinin uvtData parametresinden yararlanmak istersiniz. Bu parametre, bitmap dolgular için UV eşleme ayarlamanıza olanak sağlar.

UV eşleme, nesneleri dokulaştırma yöntemidir. Bu iki değeri esas alır: U yatay (x) değeri ve V dikey (y) değeri. Bunlar piksel değerlerini değil, yüzdeleri esas alır. 0 U ve 0 V, görüntünün sol üst kısmı iken, 1 U ve 1 V de görüntünün sağ alt kısmıdır:

Bitmap görüntüde UV 0 ve 1 konumları

Üçgen vektörlerinin bir görüntüdeki ilgili konumlarla ilişkilendirilmesi için söz konusu üçgen vektörlerine UV koordinatları verilebilir:



Bitmap görüntünün üçgen alanının UV koordinatları

UV değerleri, üçgenin noktalarıyla tutarlı kalır:

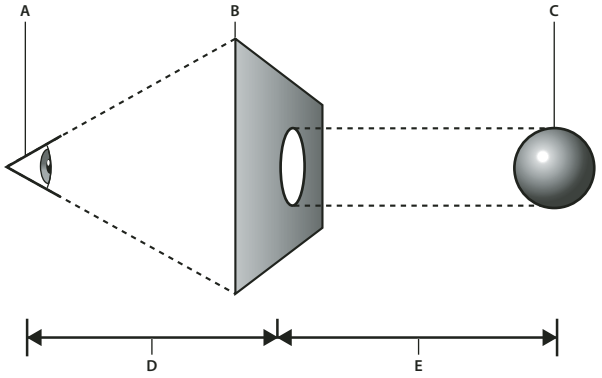
Üçgenin kesişme noktaları hareket eder ve bitmap, tek bir noktanın UV değerlerini aynı tutmak için deforme olur

ActionScript 3B dönüştürmeler, bitmap'le ilişkilendirilmiş üçgene uygulanırken, bitmap görüntü, UV değerlerini esas alan üçgene uygulanır. Bu nedenle, matris hesaplamalarını kullanmak yerine, üç boyutlu efektler oluşturmak için UV değerlerini ayarlayın.

`Graphics.drawTriangles()` yöntemi ayrıca üç boyutlu dönüştürmeler için isteğe bağlı bir bilgiyi de kabul eder: T değeri. uvtData öğesindeki T değeri, 3B perspektifi veya daha net olmak gerekirse, ilişkilendirilmiş tepenin ölçek faktörünü temsil eder. UVT eşleme, UV eşlemeye perspektif düzeltmesi ekler. Örneğin, bir nesne, 3B alanda "orijinal" boyutunun %50'si kadar görünecek şekilde görüş açısından uzağa konumlandırılırsa, o nesnenin T değeri 0,5 olur. 3B alanda nesneleri temsil etmek için üçgenler çizildiğinden, bunların z ekseni boyunca konumları T değerlerini belirler. T değerini belirleyen eşitlik şöyledir:

```
T = focalLength/(focalLength + z);
```

Bu eşitlikte, focalLength bir odak uzaklığını veya görünümde sağlanan perspektif miktarını dikte eden hesaplanmış "ekran" konumunu temsil eder.



Odak uzaklığı ve z değeri
***A.** görüş açısı **B.** ekran **C.** 3B nesne **D.** focalLength değeri **E.** z değeri*

Temel şekilleri daha uzak mesafede görünecek şekilde ölçeklemek için T değeri kullanılır. Bu genellikle 3B noktaları 2B noktalara dönüştürmek için kullanılan değerdir. UVT verileri olması durumunda, perspektif içeren bir üçgen içindeki noktalar arasında bitmap'i ölçeklemek için de bu kullanılır.

UVT değerlerini tanımladığınızda, T değeri doğrudan tepe için tanımlanan UV değerlerini izler. T'nin eklenmesiyle, `uvtData` parametresindeki her üç değer (U, V ve T), `vertices` parametresindeki her iki değerle (x ve y) eşleştirilir. UV değerleri tek olduğunda, uvtData.length == vertices.length. T değeri eklendiğinde, uvtData.length = 1.5*vertices.length.

Şu örnek, UVT verileri kullanılarak 3B alanda döndürülen bir düzlemi gösterir. Bu örnek, BitmapData nesnesine atanabilmesi için ocean.jpg görüntüsünü yüklemek üzere ocean.jpg adında bir görüntüyü ve ImageLoader adında bir “yardımcı” sınıfı kullanır.

Burada ImageLoader sınıf kaynağı verilmiştir (bu kodu ImageLoader.as adındaki bir dosyaya kaydedin):

```
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);
    }
    }
}
```

Burada da görüntünün bir ufuk noktasına doğru daralıyor ve dönüyor gibi görünmesini sağlamak için üçgenleri, UV eşlemeyi ve T değerlerini kullanan ActionScript verilmiştir. Bu kodu Spinning3dOcean.as adındaki bir dosyaya kaydedin:

```
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
```

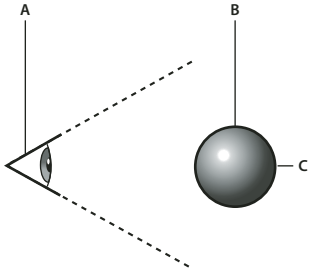
```
            t1 = focalLength/(focalLength + z1);
            t2 = focalLength/(focalLength + z2);
            t3 = focalLength/(focalLength + z3);
            t4 = focalLength/(focalLength + z4);

            // 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);
        }
    }
}
```

Bu örneği test etmek için, bu iki sınıf dosyasını “ocean.jpg” adındaki bir görüntüyle aynı dizine kaydedin. Orijinal bitmap'in, 3B alanda gittikçe kayboluyor ve dönüyor gibi görünmesi için nasıl dönüştürüldüğünü görebilirsiniz.

Yüzey kaldırma

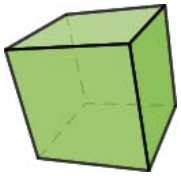
Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Yüzey kaldırma, geçerli görüş açısından gizlendikleri için oluşturucunun, üç boyutlu bir nesnenin hangi yüzeylerini oluşturmaması gerektiğini belirleyen işlemdir. 3B alanda, üç boyutlu nesnenin "arkasındaki" yüzey, görüş açısından gizlenir:



3B nesnenin arkası, görüş açısından gizlenir.
***A.** görüş açısı **B.** 3B nesne **C.** üç boyutlu nesnenin arkası*

Doğası gereği tüm üçgenler, boyutu, şekli veya konumuna bakılmaksızın her zaman oluşturulur. Yüzey kaldırma, Flash Player veya AIR uygulamasının 3B nesnenizi doğru şekilde oluşturmasını sağlar. Ayrıca daire oluşturma işlemini azaltmak için bazen oluşturucu tarafından bazı üçgenlerin atlanmasını istersiniz. Alanda dönen bir küpü göz önünde bulundurun. Görüş açısı içinde olmayan kenarlar, küpün diğer tarafındaki diğer yöne baktığından, asla o küpün üç kenarından fazlasını görmezsiniz. Bu kenarlar görünmeyeceğinden, oluşturucunun bunları çizmemesi gerekir. Yüzey kaldırma olmadan, Flash Player veya AIR uygulaması hem ön hem de arka kenarları oluşturur.



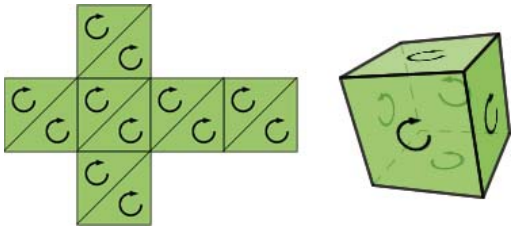
Küpte ise geçerli görüş açısından görünmeyen kenarlar vardır

Bu nedenle, `Graphics.drawTriangles()` yöntemi, yüzey kaldırma değeri oluşturmak için dördüncü bir parametre içerir:

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

culling parametresi, `TriangleCulling` numaralandırma sınıfındaki bir değerdir: `TriangleCulling.NONE`, `TriangleCulling.POSITIVE` ve `TriangleCulling.NEGATIVE`. Bu değerler, nesnenin yüzeyini tanımlayan üçgen yolunun yönüne bağlıdır. Yüzey kaldırmanın belirlenmesine yönelik ActionScript API'si, 3B şeklin tüm dışarı bakan üçgenlerinin aynı yol yönünde çizildiğini varsayar. Üçgenin bir yüzü ters döndüğünde, yol yönü de değişir. Bu noktada, üçgenin yüzeyi kaldırılabilir. (Oluşturma işleminden çıkarılır.)

Böylece, `POSITIVE` öğesinin bir `TriangleCulling` değeri, pozitif yol yönüne (saat yönüne) sahip üçgenleri kaldırır. `NEGATIVE` öğesinin `TriangleCulling` değeri, negatif (saat yönünün tersi) yol yönüne sahip üçgenleri kaldırır. Küp olması durumunda, öne bakan yüzeyler pozitif yol yönüne sahip olurken, arkaya bakan yüzeyler negatif yol yönüne sahip olur:



Yol yönünü göstermek için "sarılmamış" bir küp. "Sarılmadığında", arka taraftaki yol yönü ters çevrilir.

Yüzey kaldırmanın nasıl çalıştığını görmek için, “UV eşleme” sayfa 366 bölümündeki önceki örnekle başlayın, `drawTriangles()` yönteminin culling parametresini `TriangleCulling.NEGATIVE` olarak ayarlayın:

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

Görüntünün "arka" tarafının nesne döndükçe oluşturulmadığını göreceksiniz.

Bölüm 20: Metinle Çalışmanın temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe® Flash® Player veya Adobe® AIR™ uygulamasında metni ekranda görüntülemek için, TextField sınıfının bir örneğini ya da Flash Metin Motoru sınıflarını kullanın. Bu sınıflar, metin oluşturmanıza, görüntülemenize ve biçimlendirmenize olanak sağlar. Alternatif olarak, Flash Metin Motoru sınıflarını temel alan bir bileşen kütüphanesi olan ancak kullanım kolaylığı sağlamak amacıyla tasarlanan Metin Mizanpajı Çerçevesi'ni (TLF) kullanabilirsiniz. Mobil aygıtlarda, metin girişi için StageText sınıfını kullanabilirsiniz.

Metin alanları için belirli bir içerik oluşturabilir veya metin için kaynak belirleyebilir ve daha sonra söz konusu metnin görünümünü ayarlayabilirsiniz. Ayrıca, kullanıcı metin girdiğinde veya bir köprü metni bağlantısını tıklattığında kullanıcı olaylarına yanıt verebilirsiniz.

Hem TextField sınıfı hem de Flash Text Engine sınıfları, Flash Player ve AIR'de metin görüntülemenize ve yönetmenize izin verir. TextField sınıfını ekran ve girdi için metin nesnesi oluşturmak üzere kullanabilirsiniz. TextField sınıfı TextArea ve TextInput gibi diğer metin tabanlı bileşenler için temeli sağlar. TextFormat sınıfını, TextField nesneleri için karakter ve paragraf biçimlendirmesini ayarlamak için kullanabilir, Textfield.styleSheet özelliğini ve StyleSheet sınıfını kullanarak Basamaklı Stil Sayfaları (CSS) uygulayabilirsiniz. Bir metin alanına doğrudan gömülü medya (film klipleri, SWF dosyaları, GIF dosyaları, PNG dosyaları ve JPEG dosyaları) içerebilen HTML biçimli metin atayabilirsiniz.

Flash Player 10 ve Adobe AIR 1.5'ten itibaren kullanılabilir olan Flash Metin Motoru, gelişmiş metin metriği, biçimlendirme ve iki yönlü metin denetimi için düşük düzeyde destek sağlar. Ayrıca, iyileştirilmiş metin akışı ve geliştirilmiş dil desteği sunar. Metin öğelerini oluşturmak ve yönetmek için Flash Text Engine özelliğini kullanabilirsiniz, ancak bu özellik aslen metin işleme bileşenlerini oluşturma işleminin temeli olarak tasarlanmıştır ve yüksek düzeyde programlama uzmanlığı gerektirir. Flash Text Engine özelliğine dayanan metin işleme bileşenini içeren Metin Mizanpajı Çerçevesi, yeni metin motorunun gelişmiş özelliklerini kullanmak için daha kolay bir yol sağlar. Metin Mizanpajı Çerçevesi, tamamen ActionScript 3.0'da oluşturulmuş olan genişletilebilir bir kütüphanedir. Varolan TLF bileşenini veya kendi metin bileşeninizi oluşturmak için çerçeveyi kullanabilirsiniz.

AIR 3'ten itibaren kullanılabilir olan StageText sınıfı, yerel bir metin girişi alanı sağlar. Bu alan, aygıt işletim sistemi tarafından sağlandığından, aygıt kullanıcılarının en çok alışkın olduğu deneyimi sağlar. StageText örneği, bir görüntüleme nesnesi değildir. Bunu, görüntüleme listesine eklemek yerine, görünüm penceresi adı verilen sahne alanına bir örnek, sahne alanı ve görüntüleme alanı atarsınız. StageText örneği, herhangi bir görüntüleme nesnesinin önünde görüntülenir.

Daha fazla bilgi için, bkz.:

- "TextField sınıfını kullanma" sayfa 375
- "Flash Metin Motoru'nu kullanma" sayfa 400
- "Metin Mizanpajı Çerçevesini Kullanma" sayfa 430
- Native text input with StageText (StageText ile yerel metin girişi)

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi metin işlemeyle ilgili önemli terimleri içerir:

Basamaklı stil sayfaları XML (veya HTML) biçiminde yapılandırılmış içeriğe yönelik stilleri ve biçimlendirmeyi belirlemek için standart bir sözdizimi.

Aygıt fontu Kullanıcının makinesine yüklenen bir font.

Dinamik metin alanı İçeriği ActionScript tarafında değiştirilebilen ancak kullanıcı girdisi tarafından değiştirilemeyen bir metin alanı

Gömülü font Karakter anahat verileri, uygulama SWF dosyasında saklanan bir font.

HTML metni ActionScript kullanılarak bir metin alanına girilen, gerçek metin içeriğinin yanı sıra HTML biçimlendirme etiketlerini de içeren metin içeriği.

Girdi metin alanı İçeriği, kullanıcı girdisi veya ActionScript tarafından değiştirilebilen bir metin alanı.

Karakter aralığı Sözcüklerdeki aralığı daha oranlı ve metni daha kolay okunur hale getirmek için karakter çiftleri arasındaki aralığa yönelik ayarlama.

Statik metin alanı SWF dosyası çalışırken içeriği değiştirilemeyen, geliştirme aracında oluşturulan bir metin alanı.

Metin satırı metriği Bir metin alanındaki metin içeriğinin, metnin satır taban çizgisi, karakterlerin üst kısmının yüksekliği, harflerin alt çıkıntıları (bazı küçük harflerin, satır taban çizgisini aşan kısımları) vb. gibi çeşitli parçalarının boyutlarının ölçümleri.

İzleme Yoğunluğu artırmak veya azaltmak ve metni daha okunabilir hale getirmek için harf grupları veya metin blokları arasındaki aralığa yönelik ayarlama.

Bölüm 21: TextField sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

TextField sınıfının bir örneğini Adobe® Flash® Player veya Adobe ®AIR™ uygulamalarında ekranda metin görüntülemek ya da bir metin girdi alanı oluşturmak için kullanabilirsiniz. TextField sınıfı TextArea bileşenleri veya TextInput bileşenleri gibi diğer metin tabanlı bileşenler için temeldir.

Metin alanı içeriği, SWF dosyasında önceden belirtilmiş, bir metin dosyasından veya veri tabanından yüklenmiş ya da uygulamanızla etkileşim kuran bir kullanıcı tarafından girilmiş olabilir. Bir metin alanı içinde metin, oluşturulmuş HTML içine gömülü görüntüler içeren oluşturulmuş HTML içeriği olarak görünebilir. Metin alanının bir örneğini oluşturduktan sonra, metnin görünümünü denetlemek için TextFormat veya StyleSheet gibi flash.text sınıfları kullanabilirsiniz. flash.text paketinde ActionScript'te metin oluşturma, yönetme ve biçimlendirmeyle ilgili hemen hemen tüm sınıflar bulunur.

Formatlamayı bir TextFormat nesnesiyle tanımlayarak ve söz konusu nesneyi metin alanına atayarak metni formatlayabilirsiniz. Metin alanınız HTML metni içeriyorsa, metin alanı içeriğinin belirli parçalarına stil atamak için metin alanına bir StyleSheet nesnesi uygulayabilirsiniz. TextFormat nesnesi veya StyleSheet nesnesi, metnin görünümünü tanımlayan, renk, boyut ve ağırlık gibi özellikler içerir. TextFormat nesnesi, özellikleri bir metin alanının tüm içeriğine veya bir metin aralığına atar. Örneğin, aynı metin alanı içinde, bir cümle kalın kırmızı metinken, sonraki cümle mavi italik metin olabilir.

flash.text paketindeki sınıfların yanı sıra, metinle ilgili kullanıcı eylemlerine yanıt vermek için flash.events.TextEvent sınıfını kullanabilirsiniz.

Daha fazla Yardım konusu

"Metin biçimlerini atama" sayfa 383

"HTML metnini görüntüleme" sayfa 377

"Basamaklı stil sayfaları uygulama" sayfa 383

Metin görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe Flash Builder ve Flash Professional gibi geliştirme araçları, metin görüntüleme için metinle ilgili bileşenler veya metin araçları da dahil olmak üzere birkaç seçenek sağlasa da, programlama yoluyla metin görüntülemenin en basit yöntemi bir metin alanı kullanmaktır.

Metin türleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir metin alanındaki metnin türü, kaynağına göre belirlenir:

- Dinamik metin

  Dinamik metin, bir metin dosyası, XML dosyası veya uzak web hizmeti gibi, harici bir kaynaktan yüklenen içeriği içerir.

- Girdi metni

  Girdi metni, bir kullanıcı tarafından girilen herhangi bir metin veya kullanıcının düzenleyebildiği dinamik metindir. Girdi metnini formatlamak için bir stil sayfası ayarlayabilir veya girdi içeriği için metin alanına özellik atamak üzere flash.text.TextFormat sınıfını kullanabilirsiniz. Daha fazla bilgi için "Metin girdisi yakalama" sayfa 381 bölümünü inceleyin.

- Statik metin

  Statik metin yalnızca Flash Professional üzerinden oluşturulur. ActionScript 3.0'ı kullanarak bir statik metin örneği oluşturamazsınız. Ancak, varolan bir statik metin örneğini işlemek için StaticText ve TextSnapshot gibi ActionScript sınıflarını kullanabilirsiniz. Daha fazla bilgi için, bkz. "Statik metinle çalışma" sayfa 389.

Metin alanı içeriğini değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`flash.text.TextField.text` özelliğine bir dize atayarak dinamik metin tanımlayabilirsiniz. Aşağıdaki gibi, bir dizeyi doğrudan özelliğe atayabilirsiniz:

```
myTextField.text = "Hello World";
```

Ayrıca aşağıdaki örnekte olduğu gibi, `text` özelliğine, komut dosyanızdaki bir değişkenden bir değer de atayabilirsiniz.

```
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;
        }
    }
}
```

Alternatif olarak, `text` özelliğine, uzak bir değişkenden bir değer atayabilirsiniz. Uzak kaynaklardan metin değerleri yüklemek için üç seçeneğiniz bulunmaktadır:

- flash.net.URLLoader ve flash.net.URLRequest sınıfları, yerel veya uzak bir konumdan metin için değişken yükler.

- `FlashVars` niteliği, SWF dosyasını barındıran HTML sayfasına gömülüdür ve metin değişkenleri için değerler içerebilir.
- flash.net.SharedObject sınıfı, değerlerin kalıcı olarak saklanmasını yönetir. Daha fazla bilgi için, bkz. "Yerel verileri saklama" sayfa 709.

HTML metnini görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

flash.text.TextField sınıfı, metin dizenizi, içeriği formatlamak için HTML etiketleri içeren bir metin dizesi olarak tanımlamak için kullanabileceğiniz bir `htmlText` özelliğine sahiptir. Aşağıdaki örnekte olduğu gibi, Flash Player veya AIR uygulamasının metni HTML olarak oluşturması için dize değerinizi `htmlText` özelliğine (`text` özelliğine değil) atamanız gerekir:

```
var myText:String = "<p>This is <b>some</b> content to <i>render</i> as <u>HTML</u> text.</p>";
myTextBox.htmlText = myText;
```

Flash Player ve AIR, `htmlText` özelliği için bir HTML etiketi ve varlığı altkümesini destekler. ActionScript 3.0 Başvurusu'ndaki `flash.text.TextField.htmlText` özellik tanımı, desteklenen HTML etiketleri ve varlıkları hakkında ayrıntılı bilgiler sağlar.

`htmlText` özelliğini kullanarak içeriğinizi belirledikten sonra, içeriğinizin formatlamasını yönetmek için stil sayfalarını veya `textformat` etiketini kullanabilirsiniz. Daha fazla bilgi için, bkz. "Metin biçimlendirme" sayfa 383.

Metin alanlarında görüntü kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İçeriğinizi HTML metni olarak görüntülemenin diğer bir avantajı, metin alanına görüntü ekleyebilmenizdir. `img` etiketini kullanarak yerel veya uzak bir görüntüye başvurabilir ve görüntünün ilişkili metin alanı içinde görünmesini sağlayabilirsiniz.

Aşağıdaki örnek, `myTextBox` adlı bir metin alanı oluşturur ve SWF dosyasıyla aynı dizinde saklanan ve bir göz içeren JPG görüntüsünü görüntülenen metne dahil eder:

```
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;
        }
    }
}
```

`img` etkiketi, JPEG, GIF, PNG ve SWF dosyalarını destekler.

Bir metin alanında kayan metin

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Birçok durumda, metniniz, metni görüntüleyen metin alanından daha uzun olabilir. Veya bir kullanıcının bir defada görüntülenebilecek şekilde daha fazla metin girmesine olanak sağlayan bir girdi alanına sahip olabilirsiniz. Dikey veya yatay olarak uzun içeriği yönetmek için flash.text.TextField sınıfının kaydırma ile ilişkili özelliklerini kullanabilirsiniz.

Kaydırma ile ilişkili özellikler arasında, `TextField.scrollV`, `TextField.scrollH` ile `maxScrollV` ve`maxScrollH` bulunur. Fare tıklatması veya tuşa basılması gibi olaylara yanıt vermek için bu özellikleri kullanın.

Aşağıdaki örnek, belirli bir boyuta sahip olan ve alanın bir defada görüntüleyebileceğinden daha fazla metin içeren bir metin alanı oluşturur. Kullanıcı metin alanını tıklattığında, metin dikey olarak kayar.

```
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++;
        }
    }
}
```

Metni seçme ve işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dinamik metni veya girdi metnini seçebilirsiniz. TextField sınıfının metin seçim özellikleri ve yöntemleri, işlenecek metin aralığını ayarlamak için dizin konumları kullandığından, içeriği bilmeseniz de programlama yoluyla dinamik metni veya girdi metnini seçebilirsiniz.

Not: *Flash Professional'da, bir statik metin alanındaki seçilebilir seçeneği belirlerseniz, dışa aktarılan ve görüntüleme listesine yerleştirilen metin alanı normal bir dinamik metin alanı olur.*

Metin seçme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`flash.text.TextField.selectable` özelliği, varsayılan olarak `true` değerine sahiptir ve `setSelection()` yöntemini kullanarak programlama yoluyla metin seçebilirsiniz.

Örneğin, bir metin alanındaki belirli bir metni, kullanıcı metin alanını tıklattığında seçilmek üzere ayarlayabilirsiniz:

```
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);
}
```

Benzer şekilde, bir metin alanındaki metnin ilk görüntülendiği zaman seçilmesini istiyorsanız, metin alanı görüntüleme listesine eklendiğinde çağrılan bir olay işleyicisi işlevi oluşturun.

Kullanıcı tarafından seçilen metni yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Programlama yoluyla metin seçecek şekilde ayarlanamamaları için "salt okunur" olan TextField `selectionBeginIndex` ve `selectionEndIndex` özellikleri, kullanıcının geçerli seçimini yakalamak için kullanılabilir. Bunlara ek olarak, girdi metni alanları, `caretIndex` özelliğini kullanabilir.

Örneğin aşağıdaki kod, kullanıcı tarafından seçilen metnin dizin değerlerini izler:

```
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);
}
```

Metin görünümünü değiştirmek için seçime bir TextFormat nesne özellikleri koleksiyonu uygulayabilirsiniz. Seçilen metne bir TextFormat nesne özellikleri koleksiyonu uygulama hakkında daha fazla bilgi için, bkz. “Bir metin alanında metin aralıklarını formatlama” sayfa 386.

Metin girdisi yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varsayılan olarak, bir metin alanının `type` özelliği `dynamic` olarak ayarlanmıştır. TextFieldType sınıfını kullanarak `type` özelliğini `input` olarak ayarlarsanız, kullanıcı girdisini toplayabilir ve değeri uygulamanızın diğer bölümlerinde kullanmak için kaydedebilirsiniz. Girdi metinleri, formlar ve programın başka bir yerinde kullanılmak üzere kullanıcı tarafından bir metin değeri tanımlanmasını isteyen herhangi bir uygulama için kullanışlıdır.

Örneğin, aşağıdaki kod `myTextBox` adlı bir girdi metni alanı oluşturur. Kullanıcı alana metin girdiğinde, `textInput` olayı tetiklenir. `textInputCapture` adlı bir olay işleyicisi, girilen metnin dizesini yakalar ve bu dizeye bir değişken atar. Flash Player veya AIR, yeni metni `myOutputBox` adlı başka bir metin alanında görüntüler.

```
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;
        }

    }
}
```

Metin girdisini kısıtlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Girdi metni alanları genellikle formlar veya uygulamalardaki iletişim kutuları için kullanıldığından, kullanıcının bir metin alanına girebileceği karakter türlerini sınırlamak ve hatta parola gibi bir metni gizli tutmak isteyebilirsiniz. flash.text.TextField sınıfı, kullanıcı girdisini denetlemek için ayarlayabileceğiniz bir `displayAsPassword` özelliğine ve bir `restrict` özelliğine sahiptir.

`displayAsPassword` özelliği, kullanıcı yazarken basitçe metni gizler (bir dizi yıldız işareti şeklinde görüntüler). `displayAsPassword` özelliği `true` değerine ayarlandığında, Kes ve Kopyala komutları ve bunların ilgili klavye kısayolları çalışmaz. Aşağıdaki örnekte gösterildiği gibi, `displayAsPassword` özelliğini, arka plan ve renk gibi diğer özelliklerle aynı şekilde atarsınız:

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

`restrict` özelliği, kullanıcının girdi metni alanına hangi karakterleri yazmasına izin verildiğini belirtmeniz gerektiğinden, biraz daha karmaşıktır. Belirli harflere, sayılara veya harf, sayı ve karakter aralıklarına izin verebilirsiniz. Aşağıdaki kod, kullanıcının metin alanına yalnızca büyük harfler (sayıları ve özel karakterleri de giremeyecek şekilde) girmesine izin verir:

```
myTextBox.restrict = "A-Z";
```

ActionScript 3.0, aralıkları tanımlamak için kısa çizgi, hariç tutulan karakterleri tanımlamak için de şapka kullanır. Bir girdi metni alanında neyin kısıtlanmış olduğunu tanımlama hakkında daha fazla bilgi için, ActionScript 3.0 Başvurusu'ndaki `flash.text.TextField.restrict` özellik girişine bakın.

Not: *`flash.text.TextField.restrict` özelliğini kullanırsanız, çalışma zamanı kısıtlı harfleri otomatik olarak izin verilen büyük harf/küçük harf durumuna dönüştürür. `fl.text.TLFTextField.restrict` özelliğini kullanırsanız (yani bir TLF metin alanı kullanırsanız), çalışma zamanı kısıtlı harfleri yok sayar.*

Metin biçimlendirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Metnin görüntülenmesini programlama yoluyla formatlamak için birkaç seçeneğiniz bulunmaktadır. `TextFIeld.thickness`, `TextField.textColor` ve `TextField.textHeight` gibi özellikleri doğrudan TextField örneğinde ayarlayabilirsiniz.Veya metin alanının içeriğini `htmlText` özelliğini kullanarak belirleyebilir ve `b`, `i` ve `u` gibi desteklenen HTML etiketlerini kullanabilirsiniz. Ancak aynı zamanda, düz metin içeren metin alanlarına TextFormat nesneleri veya `htmlText` özelliğini içeren metin alanlarına StyleSheet nesneleri uygulayabilirsiniz. TextFormat ve StyleSheet nesnelerinin kullanılması, uygulamanız genelinde metin görünümü açısından en fazla denetimi ve tutarlılığı sağlar. Bir TextFormat veya StyleSheet nesnesi tanımlayabilir ve bunu uygulamanızdaki birçok metin alanına veya tüm metin alanlarına uygulayabilirsiniz.

Metin biçimlerini atama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

TextFormat sınıfını, birkaç farklı metin görüntüsü özelliği ayarlamak ve bunları bir TextField nesnesinin tüm içeriğine veya bir metin aralığına uygulamak için kullanabilirsiniz.

Aşağıdaki örnek, bir TextFormat nesnesini bir TextField nesnesinin tümüne uygular ve ikinci bir TextFormat nesnesini söz konusu TextField nesnesinin içindeki bir metin aralığına uygular:

```
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);
```

`TextField.setTextFormat()` yöntemi, yalnızca zaten metin alanında görüntülenen metni etkiler. TextField içeriği değişirse, uygulamanızın formatlamayı yeniden uygulamak için `TextField.setTextFormat()` yöntemini tekrar çağırması gerekebilir. Ayrıca, TextField `defaultTextFormat` özelliğini, kullanıcı tarafından girilen metin için kullanılacak formatı belirtecek şekilde de ayarlayabilirsiniz.

Basamaklı stil sayfaları uygulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Metin alanları, düz metin veya HTML formatlı metin içerebilir. Düz metin, örneğin `text` özelliğinde saklanır ve HTML metni, `htmlText` özelliğinde saklanır.

CSS stili bildirimlerini, birçok farklı metin alanına uygulayabileceğiniz metin stillerini tanımlamak için kullanabilirsiniz. CSS stil bildirimleri, uygulama kodunuzda oluşturulabilir veya harici bir CSS dosyasından çalışma zamanında yüklenebilir.

flash.text.StyleSheet sınıfı, CSS stillerini işler. StyleSheet sınıfı, sınırlı bir CSS özellikleri kümesini tanır. StyleSheet sınıfının desteklediği stil özelliklerinin ayrıntılı listesi için, ActionScript 3.0 Başvurusu'ndaki flash.textStylesheet girişine bakın.

Aşağıdaki örnekte gösterildiği gibi, CSS'i kodunuzda oluşturabilir ve bu stilleri bir StyleSheet nesnesi kullanarak HTML metnine uygulayabilirsiniz:

```
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);
```

Örnek kod, bir StyleSheet nesnesi oluşturduktan sonra, bir stil bildirim özellikleri kümesini tutmak için basit bir nesne oluşturur. Daha sonra bu, yeni stili ".darkred" adıyla stil sayfasına ekleyen `StyleSheet.setStyle()` yöntemini çağırır. Ardından, StyleSheet nesnesini TextField `styleSheet` özelliğine atayarak stil sayfası formatlamasını uygular.

CSS stillerinin etkili olması için, stil sayfasının, `htmlText` özelliği ayarlanmadan önce TextField nesnesine uygulanması gerekir.

Tasarım gereği, bir stil sayfasına sahip bir metin alanı düzenlenebilir değildir. Bir girdi metni alanınız varsa ve buna bir stil sayfası atarsanız; metin alanı, stil sayfasının özelliklerini gösterir, ancak kullanıcının metin alanına yeni metin girmesine izin vermez. Ayrıca, aşağıdaki ActionScript API'lerini, atanmış bir stil sayfasına sahip bir metin alanında kullanamazsınız:

- `TextField.replaceText()` yöntemi
- `TextField.replaceSelectedText()` yöntemi
- `TextField.defaultTextFormat` özelliği
- `TextField.setTextFormat()` yöntemi

Bir metin alanına bir stil sayfası atanır, ancak daha sonra `TextField.styleSheet` özelliği `null` olarak ayarlanırsa, hem `TextField.text` hem de `TextField.htmlText` özelliği, önceden atanan stil sayfasının formatlamasını dahil etmek üzere etiketler ve nitelikler ekler. Orijinal `htmlText` özelliğini korumak için, stil sayfasını `null` olarak ayarlamadan önce özelliği bir değişkenin içine kaydedin.

Harici bir CSS dosyası yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Formatlamaya yönelik CSS yaklaşımı, çalışma zamanında harici bir dosyadaki CSS bilgilerini yükleyebildiğinizde daha güçlü hale gelir. CSS verileri uygulama için harici olduğunda, ActionScript 3.0 kaynak kodunuzu değiştirmeye gerek duymadan metninizin görsel stilini uygulamanızda değiştirebilirsiniz. Uygulamanız konuşlandırıldıktan sonra, uygulamanın görünümünü değiştirmek için, uygulama SWF dosyasını yeniden konuşlandırmaya gerek duymadan harici bir CSS dosyasını değiştirebilirsiniz.

`StyleSheet.parseCSS()` yöntemi, CSS verileri içeren bir dizeyi StyleSheet nesnesinde stil bildirimlerine dönüştürür. Aşağıdaki örnek, harici bir CSS dosyasının nasıl okunacağını ve stil bildirimlerinin bir TextField nesnesine nasıl uygulanacağını gösterir:

İlk olarak, yüklenecek olan example.css adlı CSS dosyasının içeriği aşağıda verilmiştir:

```
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;
}
```

Bunu, example.css dosyasını yükleyen ve stilleri TextField içeriğine uygulayan bir sınıfın ActionScript kodu izler:

```
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;
        }
    }
}
```

CSS verileri yüklendiğinde, `onCSSFileLoaded()` yöntemi, stil bildirimlerini StyleSheet nesnesine aktarmak için `StyleSheet.parseCSS()` yöntemini çalıştırır ve çağırır.

Bir metin alanında metin aralıklarını formatlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`setTextFormat()`, flash.text.TextField sınıfının kullanışlı bir yöntemidir. `setTextFormat()` yöntemini kullanarak, kullanıcı girdisine yanıt vermek amacıyla bir metin alanının bir parçasının içeriğine, kullanıcılara belirli girişlerin zorunlu olduğunu hatırlatması veya kullanıcı metnin bir parçasını seçtiğinde bir metin parçasının alt bölümündeki vurguyu değiştirmesi gereken formlar gibi belirli özellikler atayabilirsiniz.

Aşağıdaki örnek, kullanıcı metin alanını tıklattığında `myTextField` içeriğinin bir parçasının görünümünü değiştirmek için bir karakter aralığı üzerinde `TextField.setTextFormat()` kullanır:

```
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);
}
```

Gelişmiş metin oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, flash.text package ile, gömülü fontlar, kenar yumuşatma ayarları, alfa kanal denetimi ve diğer belirli ayarlar da dahil olmak üzere, görüntülenen metnin özelliklerini denetlemek için çeşitli sınıflar sağlar. ActionScript 3.0 Başvurusu, CSMSettings, Font ve TextRenderer sınıfları da dahil olmak üzere, bu sınıfların ve özelliklerin ayrıntılı açıklamalarını sağlar.

Gömülü fontları kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanızdaki TextField için belirli bir font belirttiğinizde, Flash Player veya AIR aynı ada sahip bir aygıt fontu (kullanıcının bilgisayarında bulunan bir font) arar. Söz konusu fontu sistemde bulamazsa veya kullanıcı bu fontun aynı ada sahip olan biraz farklı bir sürümüne sahipse, metin görüntüsü amaçladığınızdan çok farklı görünebilir. Varsayılan olarak, metin Times Roman fontunda görünür.

Kullanıcının tam olarak doğru fontu gördüğünden emin olmak için, söz konusu fontu uygulama SWF dosyanıza gömebilirsiniz. Gömülü fontlar birkaç açıdan faydalıdır:

- Gömülü font karakterlerine, özellikle büyük metinlerde kenarlarının daha yumuşak görünmesini sağlayacak şekilde kenar yumuşatma uygulanmıştır.
- Gömülü font kullanan metni döndürebilirsiniz.
- Gömülü font metni, saydam veya yarı saydam yapılabilir.
- Gömülü fontlarla `kerning` CSS stilini kullanabilirsiniz.

Gömülü fontları kullanmanın getirdiği en büyük sınırlama, uygulamanızın dosya boyutunu veya indirme boyutunu artırmasıdır.

Uygulama SWF dosyanıza bir font dosyası gömmeye yönelik tam yöntem, geliştirme ortamınıza göre değişiklik gösterir.

Bir fontu gömdükten sonra, TextField öğesinin doğru gömülü fontu kullandığından şu şekilde emin olabilirsiniz:

- TextField öğesinin `embedFonts` özelliğini `true` değerine ayarlayın.
- Bir TextFormat nesnesi oluşturun, `fontFamily` özelliğini gömülü fontun adına ayarlayın ve TextFormat nesnesini TextField öğesine uygulayın. Gömülü bir font belirtirken, `fontFamily` özelliği yalnızca tek bir as içermelidir; birden çok font adı içeren virgül sınırlı bir liste kullanamaz.
- TextField öğeleri veya bileşenler için font ayarlamak üzere CSS stilleri kullanılıyorsa, `font-family` CSS özelliğini gömülü fontun adına ayarlayın. Bir gömülü font belirtmek istiyorsanız, `font-family` özelliği bir ad listesi değil, tek bir ad içermelidir.

Flash'ta font gömme

Flash Professional, TrueType fontları ve Type 1 Postscript fontları dahil olmak üzere, neredeyse sisteminize yüklediğiniz tüm fontlardan herhangi birini gömmenize izin verir.

Bir uygulamaya, aşağıdaki yöntemler de dahil olmak üzere birçok şekilde font gömebilirsiniz:

- Sahne Alanı'nda bir TextField öğesinin font ve stil özelliklerini ayarlama ve Gömülü Fontlar onay kutusunu tıklatma

- Bir font sembolü oluşturma ve buna başvurma
- Gömülü font sembolleri içeren bir çalışma zamanı paylaşımlı kütüphane oluşturup kullanma

Uygulamalara font gömme hakkında daha fazla ayrıntı için, *Flash'ı Kullanma* bölümündeki "Dinamik metin alanları ve girdi metni alanları için gömülü fontlar" konusuna bakın.

Flex'te font gömme

Bir Flex uygulamasına, aşağıdaki yöntemler de dahil olmak üzere birçok şekilde font gömebilirsiniz:

- Bir komut dosyasında `[Embed]` meta veri etiketini kullanma
- `@font-face` stil bildirimini kullanma
- Font için bir sınıf kurun ve gömmek için `[Embed]` etiketini kullanın.

Bir Flex uygulamasına yalnızca TrueType fontlarını doğrudan gömebilirsiniz. Type 1 Postscript fontları gibi diğer biçimlerdeki fontlar, ilk olarak Flash Professional kullanılarak bir SWF dosyasına gömülebilir ve daha sonra bu SWF dosyası, Flex uygulamanızda kullanılabilir. Flex'te SWF dosyalarından gömülü font kullanma hakkında daha fazla ayrıntı için, *Flex 4'ü kullanma* bölümündeki "SWF dosyalarından font gömme" başlığına bakın.

Daha fazla Yardım konusu

Tutarlı metin görünümü için font gömme

Peter deHaan: Embedding fonts (Fontları gömme)

Divillysausages.com: AS3 Font embedding masterclass (AS3 Font gömme ana sınıfı)

Keskinliği, kalınlığı ve yumuşatmayı denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varsayılan olarak Flash Player veya AIR, keskinlik, kalınlık ve kenar yumuşatma gibi metin görüntüsü denetimlerinin ayarlarını, metin yeniden boyutlandırıldığında, renk değiştirdiğinde veya çeşitli arka planlar üzerinde görüntülendiğinde belirler. Çok küçük veya çok büyük metinle çalıştığınız ya da metinle çeşitli arka planlar üzerinde çalıştığınız durumlar gibi bazı durumlarda, bu ayarlar üzerindeki denetiminizi korumak isteyebilirsiniz. `flash.text.TextRenderer` sınıfını ve CSMSettings sınıfı gibi ilişkili sınıflarını kullanarak, Flash Player veya AIR ayarlarınızı geçersiz kılabilirsiniz. Bu sınıflar size, gömülü metnin oluşturma kalitesi üzerinde kesin denetim sunar. Gömülü fontlar hakkında daha fazla bilgi için, "Gömülü fontları kullanma" sayfa 387 konusuna bakın.

Not: *`flash.text.TextField`.antiAliasType özelliği, keskinliği, kalınlığı veya gridFitType özelliğini ayarlayabilmeniz ya da `TextRenderer.setAdvancedAntiAliasingTable()` yöntemini kullanabilmeniz için `AntiAliasType.ADVANCED` değerine sahip olmalıdır.*

Aşağıdaki örnek, `myFont` adında gömülü bir fontu kullanarak görüntülenen metne sürekli kontur modülasyonu (CSM) özellikleri ve formatlaması uygular. Kullanıcı görüntülenen metni tıklattığında, Flash Player veya Adobe AIR özel ayarlar uygular:

```
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);
}
```

Statik metinle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Statik metin yalnızca Flash Professional ile oluşturulur. ActionScript'i kullanarak programlama yoluyla statik metin oluşturamazsınız. Statik metin, metnin kısa olduğu ve değişmesinin (dinamik metnin değişebilmesi gibi) amaçlanmadığı durumlarda kullanışlıdır. Statik metni, Flash Professional'da Sahne Alanı üzerinde çizilmiş bir daire veya kare gibi bir grafik öğeye benzer bir öğe olarak düşünün. Statik metin dinamik metinden daha fazla sınırlamaya sahiptir, ancak ActionScript 3.0, StaticText sınıfını kullanarak statik metnin özellik değerlerini okumanıza izin verir. Statik metinden veri okumak için TextSnapshot sınıfını da kullanabilirsiniz.

StaticText sınıfıyla statik metin alanlarına erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tipik olarak, Sahne Alanı'na yerleştirilmiş bir statik metin örneğiyle etkileşim kurmak için Flash Professional'ın Eylemler panelindeki flash.text.StaticText sınıfını kullanırsınız. Ayrıca, statik metin içeren bir SWF dosyasıyla etkileşim kuran ActionScript dosyalarıyla da çalışabilirsiniz. Her iki durumda da programlama yoluyla bir statik metin örneği oluşturamazsınız. Statik metin Flash Professional ile oluşturulur.

Varolan bir statik metin alanına başvuru oluşturmak için, görüntüleme listesindeki öğeleri yineleyin ve bir değişken atayın. Örneğin:

```
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);
}
}
```

Bir statik metin alanına başvuru elde ettikten sonra, söz konusu alanın özelliklerini ActionScript 3.0'da kullanabilirsiniz. Aşağıdaki kod, Zaman Çizelgesi'ndeki bir kareye eklenmiştir ve `myFieldLabel` adlı bir değişkenin bir statik metin başvurusuna atandığını varsaymaktadır. `myField` adlı bir dinamik metin alanı, `myFieldLabel` öğesinin `x` ve `y` değerlerine göre konumlandırılmıştır ve `myFieldLabel` öğesinin değerini tekrar görüntüler.

```
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
```

TextSnapshot sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varolan bir statik metin alanı örneğiyle programlama yoluyla çalışmak istiyorsanız, bir flash.display.DisplayObjectContainer öğesinin `textSnapshot` özelliğiyle çalışmak üzere flash.text.TextSnapshot sınıfını kullanabilirsiniz. Başka bir deyişle, `DisplayObjectContainer.textSnapshot` özelliğinden bir TextSnapshot örneği oluştururusunuz. Böylece, statik metnin verilerini almak veya parçalarını seçmek için söz konusu örneğe yöntem uygulayabilirsiniz.

Örneğin, Sahne Alanı'na "TextSnapshot Example" metnini içeren bir statik metin alanı yerleştirin. Zaman Çizelgesi'nin Kare 1'ine aşağıdaki ActionScript'i ekleyin:

```
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);
```

TextSnapshot sınıfı, söz konusu metni uygulamanın başka bir bölümünde bir değer olarak kullanmak istiyorsanız, yüklü bir SWF dosyasındaki statik metin alanlarından metin almak için kullanışlıdır.

TextField Örneği: Gazete stili metin biçimlendirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Haber Mizanpajı örneği, metni, basılı bir gazetedeki bir haber gibi görünecek şekilde formatlar. Girdi metni bir başlığı, alt başlığı ve öykünün gövdesini içerebilir. Bir görüntü genişliği ve yüksekliği verildiğinde, bu Haber Mizanpajı örneği, başlığı ve alt başlığı görüntüleme alanının tam genişliğini alacak şekilde formatlar. Öykü metni, iki veya daha fazla sütuna dağıtılır.

Bu örnek, aşağıdaki ActionScript programlama tekniklerini gösterir:

- TextField sınıfını genişletme
- Harici bir CSS dosyası yükleme ve uygulama
- CSS stillerini TextFormat nesnelerine dönüştürme
- Metin görüntüsü boyutu hakkında bilgi edinmek için TextLineMetrics sınıfını kullanma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Haber Mizanpajı uygulama dosyalarını Samples/NewsLayout klasörü içinde bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
NewsLayout.mxml veya NewsLayout.fla	Flex (MXML) veya Flash (FLA) uygulaması için kullanıcı arabirimi.
com/example/programmingas3/newslayout/StoryLayoutComponent.as	StoryLayout örneğini yerleştiren bir Flex UIComponent sınıfı.
com/example/programmingas3/newslayout/StoryLayout.as	Görüntülenecek haberin tüm bileşenlerini düzenleyen ana ActionScript sınıfı.
com/example/programmingas3/newslayout/FormattedTextField.as	TextField sınıfının, kendi TextFormat nesnesini yöneten bir alt sınıfı.
com/example/programmingas3/newslayout/HeadlineTextField.as	FormattedTextField sınıfının, font boyutlarını istenen bir genişliğe sığacak şekilde ayarlayan bir alt sınırı.
com/example/programmingas3/newslayout/MultiColumnTextField.as	Metni iki veya daha fazla sütuna bölen bir ActionScript sınıfı.
story.css	Mizanpaj için metin stilleri tanımlayan bir CSS dosyası.

Harici CSS dosyasını okuma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Haber Mizanpajı uygulaması, yerel bir XML dosyasından öykü metni okuyarak başlar. Ardından, başlık, alt başlık ve ana metin için formatlama bilgilerini sağlayan harici bir CSS dosyasını okur.

CSS dosyası, öykü için standart bir paragraf stili ile başlık ve alt başlık için sırasıyla h1 ve h2 stilleri olmak üzere üç stil tanımlar.

```
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;
}
```

Harici CSS dosyasını okumak için kullanılan teknik, "Harici bir CSS dosyası yükleme" sayfa 384 konusunda açıklanan teknikle aynıdır. CSS dosyası yüklenince, uygulama aşağıda gösterilen `onCSSFileLoaded()` yöntemini çalıştırır.

```
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();
}
```

`onCSSFileLoaded()` yöntemi, bir StyleSheet nesnesi oluşturur ve bu nesnenin girdi CSS verilerini ayrıştırmasını sağlar. Öykünün ana metni, bir StyleSheet nesnesini doğrudan kullanabilen MultiColumnTextField nesnesinde görüntülenir. Ancak başlık alanları, formatlaması için bir TextFormat nesnesi kullanan HeadlineTextField sınıfını kullanır.

`onCSSFileLoaded()` yöntemi, bir CSS stil bildirimini her iki HeadlineTextField nesnesiyle kullanım için bir TextFormat nesnesine dönüştürmek üzere iki defa `getTextStyle()` yöntemini çağırır.

```
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;
}
```

Özellik adları ve özellik değerlerinin anlamı, CSS stil bildirimleri ve TextFormat nesneleri arasında farklılık gösterir. `getTextStyle()` yöntemi, CSS özellik değerlerini TextFormat nesnesi tarafından beklenen değerlere çevirir.

Öykü öğelerini sayfada düzenleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

StoryLayout sınıfı, başlık, alt başlık ve ana metin alanlarını gazete stili düzenlemesi şeklinde formatlar ve düzenler. `displayText()` yöntemi, ilk olarak çeşitli alanlar oluşturur ve yerleştirir.

```
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;
...
```

Her bir alan, `y` özelliğini önceki alanın `y` özelliği artı yüksekliğine eşit olacak şekilde ayarlayarak önceki alanın altına yerleştirilir. Bu dinamik yerleşim hesaplaması, HeadlineTextField nesneleri ve MultiColumnTextField nesneleri yüksekliğini içeriğinin sığacağı şekilde değiştirebildiği için gereklidir.

Font boyutunu alan boyutuna sığacak şekilde değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Piksel cinsinden bir genişlik ve görüntülenecek maksimum satır sayısı verildiğinde, HeadlineTextField metnin alana sığmasını sağlamak için font boyutunu değiştirir. Metin kısaysa, font boyutu tabloid stili bir başlık oluşturacak şekilde büyük olur. Metin uzunsa, font boyutu daha küçük olur.

Aşağıda gösterilen `HeadlineTextField.fitText()` yöntemi, font boyutlandırma işlemini yapar:

```
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;
    }
}
```

`HeadlineTextField.fitText()` yöntemi, fontu boyutlandırmak için basit bir yinelemeli teknik kullanır. lk olarak, metindeki her karakter için ortalama bir piksel sayısı tahmin eder ve buradan yola çıkarak başlangıç punto boyutunu hesaplar. Daha sonra font boyutunu değiştirir ve maksimum metin satırı sayısından daha fazlasını oluşturacak şekilde metinde sözcük kaydırma uygulanıp uygulanmadığını denetler. Çok fazla satır varsa, font boyutunu küçültmek ve yeniden denemek için `shrinkText()` yöntemini çağırır. Çok fazla satır yoksa, font boyutunu büyütmek ve yeniden denemek için `growText()` yöntemini çağırır. Font boyutunun bir punto daha büyütülmesinin çok sayıda satıra yol açacağı noktada işlem durdurulur.

Metni birden çok sütuna bölme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

MultiColumnTextField sınıfı, daha sonra gazete sütunları gibi düzenlenen birden çok TextField nesnesine metni yayar.

`MultiColumnTextField()` yapıcısı ilk olarak, burada gösterildiği gibi her sütun için bir tane olacak şekilde TextField nesnelerinin bir dizisini oluşturur:

```
    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);
    }
```

Her bir TextField nesnesi, `addChild()` yöntemiyle diziye ve görüntüleme listesine eklenir.

StoryLayout `text` özelliği veya `styleSheet` özelliği her değiştiğinde, metni yeniden görüntülemek için `layoutColumns()` yöntemini çağırır. `layoutColumns()` yöntemi, verilen mizanpaj genişliğine metnin tamamını sığdırmak için gereken doğru piksel yüksekliğini öğrenmek üzere `getOptimalHeight()` yöntemini çağırır.

```
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;
    }
}
```

İlk olarak `getOptimalHeight()` yöntemi, her sütunun genişliğini hesaplar. Ardından dizideki ilk TextField nesnesinin genişliğini ve `htmlText` özelliğini ayarlar. `getOptimalHeight()` yöntemi, metindeki sözcük kaydırması uygulanmış toplam satır sayısını bulmak için bu ilk TextField nesnesini kullanır ve bu noktadan yola çıkarak her sütunda kaç satır olması gerektiğini tanımlar. Daha sonra ilk satırdaki metnin boyutuyla ilgili ayrıntıları içeren bir TextLineMetrics nesnesini almak için `TextField.getLineMetrics()` yöntemini çağırır. `TextLineMetrics.height` özelliği, harflerin üst çıkıntısı, harflerin alt çıkıntısı ve satır aralığı da dahil olacak şekilde, bir metin satırının piksel cinsinden tam yüksekliğini temsil eder. MultiColumnTextField nesnesinin optimum yüksekliği; sütun başına satır sayısı ile satır yüksekliğinin çarpımına, bir TextField nesnesinin üstünde ve altında bulunan ikişer piksellik kenarlık için 4 değeri eklenerek elde edilir.

Tam `layoutColumns()` yönteminin kodu şu şekildedir:

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

`getOptimalHeight()` yöntemi çağrılarak `preferredHeight` özelliği ayarlandıktan sonra, `layoutColumns()` yöntemi TextField nesnelerini yineleyerek her birinin yüksekliğini `preferredHeight` değerine ayarlar. Daha sonra `layoutColumns()` yöntemi, tek bir alanda kaydırma gerçekleşmeyecek ve başka bir alanı izleyen alanlardaki metin, önceki alandaki metnin sona erdiği yerden başlayacak şekilde, her alana yeterli sayıda metin satırı dağıtır. Metin hizalama stili "yasla" olarak ayarlanmışsa, bir alandaki metnin son satırını yaslamak için `justifyLastLine()` yöntemi çağrılır. Aksi takdirde bu son satır, paragraf sonu satırı olarak değerlendirilir ve yaslanmazdı.

Bölüm 22: Flash Metin Motoru'nu kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player 10 ve Adobe® AIR™ 1.5'ten itibaren kullanılabilir olan Adobe ® Flash ® Metin Motoru, gelişmiş metin metriği, biçimlendirme ve iki yönlü metin denetimi için düşük düzeyde destek sağlar. İyileştirilmiş metin akışı ve geliştirilmiş dil desteği sunar. Basit metin öğeleri oluşturmak ve yönetmek için kullanılabilse de, FTE'nin tasarlanmasındaki asıl amaç, geliştiricilerin metin işleme bileşenleri oluşturmaları için bir temel sağlamaktır. Bu nedenle, Flash Metin Motoru, kullanıcının daha gelişmiş bir programlama uzmanlığı düzeyinde olduğunu varsayar. Basit metin öğeleri görüntülemek için, bkz. “TextField sınıfını kullanma” sayfa 375.

Bir FTE temelli bir metin işleme bileşeni içeren Metin Mizanpajı Çerçevesi, gelişmiş özelliklerini kullanmak için daha kolay bir yol sunar. Metin Mizanpajı Çerçevesi, tamamen ActionScript 3.0'da oluşturulmuş olan genişletilebilir bir kütüphanedir. Varolan TLF bileşenini veya kendi metin bileşeninizi oluşturmak için çerçeveyi kullanabilirsiniz. Daha fazla bilgi için bkz. “Metin Mizanpajı Çerçevesini Kullanma” sayfa 430.

Daha fazla Yardım konusu

flash.text.engine paketi

Metin oluşturma ve görüntüleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Metin Motoru'nu oluşturan sınıflar, metin oluşturmanıza, biçimlendirmenize ve denetlemenize olanak sağlar. Aşağıdaki sınıflar, Flash Metin Motoru ile metin oluşturmak ve görüntülemek için temel bina bloklarıdır:

- TextElement/GraphicElement/GroupElement - bir TextBlock örneğinin içeriğini içerir
- ElementFormat - bir TextBlock örneğinin içeriği için biçimlendirme niteliklerini belirtir
- TextBlock - bir metin paragrafı oluşturmak için kullanılan fabrika
- TextLine - TextBlock'tan oluşturulan bir metin satırı

Metni görüntülemek için, biçimlendirme karakteristiklerini belirleme amacıyla bir ElementFormat nesnesi kullanarak bir Dizeden bir TextElement nesnesi oluşturun. TextElement öğesini bir TextBlock nesnesinin `içerik` özelliğine atayın. `TextBlock.createTextLine()` yöntemini çağırarak görüntülenecek metnin satırlarını oluşturun. `createTextLine()` yöntemi, belirlenen genişliğe sığabilecek Dize miktarını içeren TextLine nesnesini geri getirir. Yöntemi dizenin tamamı satırlar halinde olacak şekilde biçimlendirilene kadar çağırın. Oluşturulacak satır kalmadığında, TextBlock nesnesinin textLineCreationResult özelliği `TextLineCreationResult.COMPLETE` değerine atanır. Satırları göstermek için, onları görüntüleme listesine ekleyin (uygun `x` ve `y` konum değerleriyle).

Örneğin aşağıdaki kod, varsayılan biçim ve font değerleriyle "Hello World! This is Flash Text Engine!" metnini görüntülemek için bu FTE sınıflarını kullanır. Bu basit örnekte, yalnızca tek bir metin satırı oluşturulur.

```
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;
        }
    }
}
```

`createTextLine()` parametreleri, yeni satırın başlayacağı satırı ve piksel cinsinden satırın genişliğini belirtir. Yeni satırın başlayacağı satır genellikle önceki satırdır, ancak ilk satır söz konusu olduğunda bu `null` olacaktır.

GraphicElement ve GroupElement nesneleri ekleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir görüntü veya grafik öğe görüntülemek amacıyla TextBlock nesnesine bir GraphicElement nesnesi atayabilirsiniz. Bir grafik veya görüntüden GraphicElement sınıfının bir örneğini oluşturmanız ve örneği `TextBlock.content` özelliğine atamanız yeterlidir. Normalde yaptığınız gibi `TextBlock.createTextline()` öğesini çağırarak metin satırını oluşturun. Aşağıdaki örnek, biri bir GraphicElement nesnesi, diğeri ise bir TextElement nesnesi içeren iki metin satırı oluşturur.

```
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;
        }
    }
}
```

TextElement, GraphicElement ve diğer GroupElement nesnelerinden bir grup oluşturmak için bir GroupElement nesnesi oluşturabilirsiniz. Bir GroupElement, bir TextBlock nesnesinin`içerik` özelliğine atanabilir. `GroupElement()` yapıcısının parametresi, metne, grafiğe ve grubu oluşturan grup öğelerine işaret eden bir Vektördür. Aşağıdaki örnek, iki grafik öğesini ve bir metin öğesini gruplandırır ve bunları bir birim olarak metin bloğuna atar.

```
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;
        }
    }
}
```

Metni değiştirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir TextBlock örneğindeki metni, `TextBlock.content` özelliğine atadığınız TextElement öğesindeki metni değiştirmek için `TextElement.replaceText()` öğesini çağırarak değiştirebilirsiniz.

Aşağıdaki örnek, satırın başlangıcına ilk olarak satırın başına metin eklemek, sonra satırın sonuna metin eklemek ve son olarak satırın ortasındaki metni değiştirmek için `replaceText()` öğesini kullanır.

```
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);
        }
    }
}
```

`replaceText()` yöntemi, `beginIndex` ve `endIndex` parametrelerinin belirttiği metni, `newText` parametresinin belirttiği metinle değiştirir. `beginIndex` ve `endIndex` parametrelerinin değerleri aynıysa, `replaceText()` belirtilen metni bu konuma ekler. Aksi takdirde, `beginIndex` ve `endIndex` parametrelerinin belirttiği karakterleri yeni metinle değiştirir.

FTE'de Olay İşleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Diğer görüntüleme nesnelerinde olduğu gibi, bir TextLine örneğine de olay dinleyicileri ekleyebilirsiniz. Örneğin, bir kullanıcının ne zaman fareyi bir metin satırı üzerine getirdiğini veya satırı tıklattığını algılayabilirsiniz. Aşağıdaki örnek, bu olayların her ikisini de algılar. Fareyi satırın üzerine getirdiğinizde, imleç bir düğme imlecine dönüşür ve satırı tıklattığınızda rengi değişir.

```
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);
        }
    }
}
```

Olayları yansıtma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir metin bloğundaki veya metin bloğunun bir kısmındaki olayları bir olay göndericisine de yansıtabilirsiniz. İlk olarak, bir EventDispatcher örneği oluşturun ve bunu bir TextElement örneğinin `eventMirror` nesnesine atayın. Metin bloğu tek bir metin öğesinden oluşuyorsa, metin motoru, olayları metin bloğunun tamamı için yansıtır. Metin bloğu birden çok metin öğesinden oluşuyorsa, metin motoru, olayları yalnızca `eventMirror` özelliği ayarlanmış olan TextElement örnekleri için yansıtır. Aşağıdaki örnekteki metin, üç öğeden oluşur: "Click" sözcüğü, "here" sözcüğü ve "to see me in italic" dizesi. Örnek, ikinci metin öğesi olan "here" sözcüğüne bir olay göndericisi atar ve bir olay dinleyicisi olan `clickHandler()` yöntemini ekler. `clickHandler()` yöntemi, metni italik olarak değiştirir. Ayrıca, üçüncü metin öğesinin içeriğini de "Click here to see me in normal font!" olarak değiştirir.

```
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);
        }
    }
}
```

`mouseOverHandler()` ve `mouseOutHandler()` işlevleri, imleç "here" sözcüğünün üzerinde olduğunda imleci bir düğme imleci olarak ayarlar ve imleç sözcüğün üzerinde olmadığında imleci tekrar ok olarak ayarlar.

Metin biçimlendirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`TextBlock` nesnesi, metin satırları oluşturmak için kullanılan bir fabrikadır. Bir `TextBlock` öğesinin içeriği, `TextElement` nesnesi üzerinden atanır. `ElementFormat` nesnesi, metnin biçimlendirmesini işler. ElementFormat sınıfı, satır taban çizgisi hizalaması, karakter aralığı, izleme, metin dönüşü ve font boyutu, rengi ve büyük/küçük harf durumu gibi özellikleri tanımlar. Ayrıca, "Fontlarla çalışma" sayfa 412 konusunda ayrıntılı olarak ele alınan bir `FontDescription` öğesini de içerir.

ElementFormat nesnesini kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`ElementFormat` nesnesinin yapıcısı, bir `FontDescription` da dahil olmak üzere çok sayıda isteğe bağlı parametre alır. Bu özellikleri yapıcının dışında da ayarlayabilirsiniz. Aşağıdaki örnek, basit metin satırının tanımlanması ve görüntülenmesinde çeşitli nesnelerin ilişkisini gösterir:

```
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);
        }
    }
}
```

Font rengi ve saydamlık (alfa)

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`ElementFormat` nesnesinin `renk` özelliği, font rengini ayarlar. Değer, rengin RGB bileşenlerini temsil eden bir tam sayıdır. Örneğin kırmızı için 0xFF0000 ve yeşil için 0x00FF00. Varsayılan siyahtır (0x000000).

`alpha` özelliği, bir öğenin (`TextElement` hem de `GraphicElement`) alfa saydamlığını ayarlar. Değerler 0 (tamamen saydam) ile 1 (tamamen opak, varsayılan budur) arasında değişebilir. `alpha` özelliği 0 olan öğeler görünmez ancak yine de etkindir. Bu değer, miras alınan alfa değerleriyle çarpılarak öğeyi daha saydam hale getirir.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

Satır taban çizgisi hizalaması ve kayması

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir satırdaki en büyük metnin fontu ve boyutu, o metnin baskın satır taban çizgisini belirler. `TextBlock.baselineFontDescription` ve `TextBlock.baselineFontSize` öğelerini ayarlayarak bu değerleri geçersiz kılabilirsiniz. Baskın satır taban çizgisini, metin içindeki birkaç satır taban çizgisinden biriyle hizalayabilirsiniz. Bu taban çizgileri arasında, harfin üst çıkıntı çizgisi ve alt çıkıntı çizgisi veya ideografik üst, orta veya alt kısım yer alır.

***A.** Harfin üst çıkıntısı **B.** Satır taban çizgisi **C.** Harfin alt çıkıntısı **D.** x yüksekliği*

`ElementFormat` nesnesinde, satır taban çizgisi ve hizalama özelliklerini üç özellik belirler. `alignmentBaseline` özelliği, bir `TextElement` veya `GraphicElement` öğesinin ana satır taban çizgisini ayarlar. Satır taban çizgisi, öğenin "sığdırılacağı" satırdır ve tüm metnin baskın satır taban çizgisi bu konuma göre hizalanır.

`dominantBaseline` özelliği, öğenin satır üzerindeki dikey konumunu belirleyen çeşitli satır taban çizgilerinden hangisinin kullanılacağını belirtir. Varsayılan değer `TextBaseline.ROMAN` olsa da bu, `IDEOGRAPHIC_TOP` veya `IDEOGRAPHIC_BOTTOM` satır taban çizgileri baskın olacak şekilde de ayarlanabilir.

`baselineShift` özelliği, satır taban çizgisini y ekseninde belirli bir piksel sayısı kadar hareket ettirir. Normal (döndürülmemiş) metinde pozitif bir değer, satır taban çizgisini aşağı ve negatif bir değer, satır taban çizgisini yukarı hareket ettirir.

Tipografi Büyük/Küçük Harf Durumu

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`ElementFormat` öğesinin `TypographicCase` özelliği, büyük harf, küçük harf veya küçük büyük harfler gibi metnin büyük/küçük harf durumunu belirtir.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

Metni döndürme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir metin bloğunu veya bir metin parçası içindeki glifleri 90° değerinde artışlarla döndürebilirsiniz. TextRotation sınıfı, hem metin bloğunun hem de glif dönüşünün ayarlanmasına yönelik şu sabitleri tanımlar:

Sabit	Değer	Açıklama
AUTO	“auto”	Saatin ters yönünde 90 derece dönüş belirtir. Genellikle yalnızca dönüş gerektiren glifleri döndürmek için dikey Asya metinleriyle kullanılır.
ROTATE_0	“rotate_0”	Dönüş olmadığını belirtir.
ROTATE_180	“rotate_180”	180 derece dönüş belirtir.
ROTATE_270	“rotate_270”	270 derece dönüş belirtir.
ROTATE_90	“rotate_90”	Saat yönünde 90 derece dönüş belirtir.

Bir metin bloğundaki metin satırlarını döndürmek için, metin satırını oluşturmak üzere `TextBlock.createTextLine()` yöntemini çağırmadan önce `TextBlock.lineRotation` özelliğini ayarlayın.

Metin bloğundaki veya parçadaki glifleri döndürmek için, `ElementFormat.textRotation` özelliğini, gliflerin döndürülmesini istediğiniz derece sayısına ayarlayın. Glif, bir karakteri veya birden çok gliften oluşan bir karakterin parçasını oluşturan şekildir. Örneğin, "a" harfi ve "i" harfinin üzerindeki nokta birer gliftir.

Satır içindeki karakterleri döndürmeden satırları dikey olacak şekilde döndürmek istediğiniz bazı Asya dillerinde gliflerin döndürülmesi mümkündür. Asya metinlerini döndürme hakkında daha fazla bilgi için, bkz. “Doğu Asya metnini yaslama” sayfa 416.

Aşağıda, Asya metinlerinde olduğu gibi, hem bir metin bloğunun hem de metin bloğundaki gliflerin döndürülmesine bir örnek verilmiştir. Bu örnekte bir Japonca font da kullanılmaktadır:

```
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);
        }
    }
}
```

ElementFormat öğesini kilitleme ve klonlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir `ElementFormat` nesnesi herhangi bir `ContentElement` türüne atandığında, buna ait `locked` özelliği otomatik olarak `true` değerine ayarlanır. Kilitli bir `ElementFormat` nesnesini değiştirmeyi denemek, bir `IllegalOperationError` atılmasına neden olur. En iyi uygulama, bu gibi bir nesneyi bir `TextElement` örneğine aramadan önce tam olarak tanımlamaktır.

Varolan bir `ElementFormat` örneğini değiştirmek istiyorsanız, ilk olarak `locked` özelliğini denetleyin. Değeri `true` ise, nesnenin kilitlenmemiş bir kopyasını oluşturmak için `clone()` yöntemini kullanın. Bu kilitlenmemiş nesnenin özellikleri değiştirilebilir ve daha sonra nesne `TextElement` örneğine atanabilir. Bundan oluşturulan yeni satırların tümü yeni biçimlendirmeye sahip olur. Söz konusu aynı nesneden oluşturulan ve eski biçimi kullanan önceki satırlar değiştirilmez.

```
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);
        }
    }
}
```

Fontlarla çalışma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`FontDescription` nesnesi, font yüzünü ve bunun bazı özelliklerini tanımlamak için `ElementFormat` ile birlikte kullanılır. Bu özellikler arasında font adı, ağırlığı, duruşu, oluşturması ve fontun nasıl bulunacağı (aygıt fontu ve gömülü font) yer alır.

Not: *FTE, Type 1 fontlarını veya Type 3, ATC, sfnt-wrapped CID ya da Naked CID gibi bitmap fontlarını desteklemez.*

Font özelliklerini tanımlama (FontDescription nesnesi)

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`FontDescription` nesnesinin `fontName` özelliği, tek bir ad veya virgülle ayrılmış bir ad listesi olabilir. Örneğin, “Arial, Helvetica, _sans” gibi bir listede, metin motoru, ilk olarak "Arial", daha sonra "Helvetica" ve ilk iki fonttan birini bulamazsa son olarak "_sans" arar. Font adları kümesi, üç genel aygıt fontu adını içerir: “_sans”, “_serif” ve “_typewriter”. Bunlar oynatma sistemine bağlı olarak belirli aygıt fontlarıyla eşlenir. Aygıt fontlarını kullanan tüm font açıklamalarında bunlar gibi varsayılan adları belirtmek faydalı bir uygulamadır. `fontName` belirtilmemişse, varsayılan olarak “_serif” kullanılır.

`fontPosture` özelliği, varsayılan (`FontPosture.NORMAL`) veya italik (`FontPosture.ITALIC`) olarak ayarlanabilir. `fontWeight` özelliği, varsayılan (`FontWeight.NORMAL`) veya kalın (`FontWeight.BOLD`) olarak ayarlanabilir.

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

Gömülü fontlar ve aygıt fontları

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`FontDescription` nesnesinin `fontLookup` özelliği, metin motorunun metni oluşturmak için bir aygıt fontu mu, yoksa gömülü font mu arayacağını belirtir. Bir aygıt fontu (`FontLookup.DEVICE`) belirtilirse, çalışma zamanı, fontu oynatma sisteminde arar. Bir gömülü fontun (`FontLookup.EMBEDDED_CFF`) belirtilmesi, çalışma zamanının, SWF dosyası içinde belirtilen ada sahip bir gömülü font aramasına neden olur. Yalnızca gömülü CFF (Kompakt Font Biçimi) fontları bu ayarla çalışır. Belirtilen font bulunamazsa, bir geri dönüş aygıt fontu kullanılır.

Aygıt fontları, SWF dosya boyutunun daha küçük olmasını sağlar. Gömülü fontlar, platformlar arasında daha fazla aslına uygunluk sağlar.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

Oluşturma modu ve katkı

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

CFF (Kompakt Font Biçimi) oluşturma, Flash Player 10 ve AIR 1.5'ten itibaren kullanılabilir. Bu font oluşturma türü, metni daha okunaklı hale getirir ve fontların küçük boyutlarda daha yüksek kalitede görüntülenmesine olanak sağlar. Bu ayar yalnızca gömülü fontlar için geçerlidir: `FontDescription`, `renderingMode` özelliği için varsayılan olarak bu ayarı (`RenderingMode.CFF`) kullanır. Flash Player 7 veya önceki sürümler tarafından kullanılan oluşturma türüyle eşleşmesi için bu ayarı `RenderingMode.NORMAL` olarak ayarlayabilirsiniz.

CFF oluşturma seçildiğinde, ikinci bir özellik olan `cffHinting`, fontun yatay gövdelerinin alt piksel ızgarasına nasıl sığdırıldığını denetler. Varsayılan değer olan `CFFHinting.HORIZONTAL_STEM`, CFF katkısını kullanır. Bu özelliğin `CFFHinting.NONE` olarak ayarlanması, animasyon veya büyük font boyutları için uygun olacak şekilde katkıyı kaldırır.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

FontDescription öğesini kilitleme ve klonlama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bir `FontDescription` nesnesi bir `ElementFormat` öğesine atandığında, bunun `locked` özelliği otomatik olarak `true` değerine ayarlanır. Kilitli bir `FontDescription` nesnesini değiştirmeyi denemek, bir `IllegalOperationError` atılmasına neden olur. En iyi uygulama, bu gibi bir nesneyi `ElementFormat` örneğine atamadan önce tam olarak tanımlamaktır.

Varolan bir `FontDescription` öğesini değiştirmek istiyorsanız, ilk olarak `locked` özelliğini denetleyin. Değeri `true` ise, nesnenin kilitlenmemiş bir kopyasını oluşturmak için `clone()` yöntemini kullanın. Bu kilitlenmemiş nesnenin özellikleri değiştirilebilir ve daha sonra nesne `ElementFormat` öğesine atanabilir. Bu `TextElement` öğesinden oluşturulan yeni satırların tümü yeni biçimlendirmeye sahip olur. Söz konusu aynı nesneden oluşturulan önceki satırlar değiştirilmez.

```
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);
        }
    }
}
```

Metni denetleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

FTE, yaslama ve karakter aralığının işlenmesine yönelik yeni bir metin biçimlendirme denetimleri kümesi sağlar (karakter aralığı ve izleme). Satırların bölünme şeklinin denetlenmesi ve satırların içindeki sekme duraklarının ayarlanmasına yönelik özellikler de vardır.

Metni yaslama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Metnin yaslanması, sözcükler ve bazen harfler arasındaki aralığın ayarlanması yoluyla bir paragraftaki tüm satırların aynı uzunluğa getirilmesini sağlar. Böylece metin iki tarafta da aynı hizada olurken, sözcükler ve harfler arasındaki aralıklar değişiklik gösterir. Gazete ve dergilerdeki metin sütunları genellikle yaslanır.

SpaceJustifier sınıfındaki `lineJustfication` özelliği, bir metin bloğundaki satırların yaslanmasını denetlemenize olanak sağlar. LineJustification sınıfı, yaslama seçeneğini belirtmek için kullanabileceğiniz sabitler tanımlar: `ALL_BUT_LAST`, metnin son satırı dışındaki tüm satırlarını yaslar; `ALL_INCLUDING_LAST`, son satır da dahil olmak üzere metnin tamamını yaslar; varsayılan değer olan `UNJUSTIFIED` ise metni yaslanmamış şekilde bırakır.

Metni yaslamak için, `lineJustification` özelliğini SpaceJustifier sınıfının bir örneğine ayarlayın ve bu örneği bir TextBlock örneğinin `textJustifier` özelliğine atayın. Aşağıdaki örnek, metnin son satırı hariç tüm satırlarının yaslandığı bir paragraf oluşturur.

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

Sözcüklerin yanı sıra harfler arasında da farklı aralıklar kullanmak için, `SpaceJustifier.letterspacing` özelliğini `true` değerine ayarlayın. Harf aralığının etkinleştirilmesi, bazen basit yaslama sonucu ortaya çıkabilen, sözcükler arasındaki görünümü bozan boşlukları azaltır.

Doğu Asya metnini yaslama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Doğu Asya metninin yaslanmasında dikkate alınması gereken ek noktalar vardır. Bu metinler yukarıdan aşağıya doğru yazılabilir ve kinsoku olarak da bilinen bazı karakterler, satırın başında ya da sonunda bulunamaz. JustificationStyle sınıfı, bu karakterleri işlemeye yönelik seçenekleri belirten aşağıdaki sabitleri tanımlar: `PRIORITIZE_LEAST_ADJUSTMENT`, en iyi sonucu hangisinin oluşturduğuna bağlı olarak, yaslamayı satırın genişletilmesine veya sıkıştırılmasına dayandırır. `PUSH_IN_KINSOKU`, yaslamayı satırın sonundaki kinsokunun sıkıştırılmasına veya herhangi bir kinsoku yoksa ya da bu boşluk yeterli değilse kinsokunun genişletilmesine dayandırır.

PUSH_OUT_ONLY, yaslamayı satırın genişletilmesine dayandırır. Bir dikey Asya metni bloğu oluşturmak için, TextBlock.lineRotation özelliğini TextRotation.ROTATE_90 olarak ve ElementFormat.textRotation özelliğini varsayılan değer olan TextRotation.AUTO olarak ayarlayın. textRotation değerinin AUTO olarak ayarlanması, satır döndürüldüğünde metindeki gliflerin yana dönmek yerine dikey kalmasını sağlar. AUTO ayarı, glifin Unicode özellikleri tarafından belirtildiği gibi, yalnızca tam genişlikteki glifler ve geniş glifler için saat yönünün tersine 90° dönüş sağlar. Aşağıdaki örnek, dikey bir Japonca metin bloğu görüntüler ve bunu PUSH_IN_KINSOKU seçeneğini kullanarak yaslar.

```
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
```

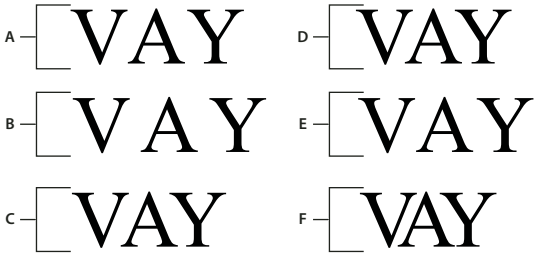
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            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;
            }
        }
    }
}
```

Karakter aralığı ve izleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Karakter aralığı ve izleme, bir metin bloğundaki bitişik karakter çiftlerinin arasındaki mesafeyi etkiler. Karakter aralığı, "WA" veya "Va" gibi karakter çiftlerinin birbirinin yakınına nasıl "yerleştirildiğini" denetler. Karakter aralığı, `ElementFormat` nesnesinde ayarlanır. Varsayılan olarak etkindir (`Kerning.ON`) ve yalnızca Kanji, Hiragana veya Katakana dışındaki karakterler arasında karakter aralığı uygulanacak şekilde OFF ya da AUTO olarak ayarlanabilir.

İzleme, metin bloğundaki tüm karakterlerin arasına belirli sayıda piksel ekler veya karakterlerin arasından belirli sayıda piksel çıkarır ve `ElementFormat` nesnesinden ayarlanır. Hem gömülü fontlarla hem de aygıt fontlarıyla çalışır. FTE iki izleme özelliğini destekler: bir karakterin sol tarafında piksel ekleme/çıkarma işlemi yapan `trackingLeft` ve sağ tarafında ekleme/çıkarma işlemi yapan `trackingRight`. Karakter aralığı kullanılıyorsa, her karakter çifti için karakter aralığı değerlerine izleme değeri eklenir veya karakter aralığı değerlerinden izleme değeri çıkarılır.



***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;
```

Kaydırılan metin için satır kesmeleri

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`ElementFormat` nesnesinin `breakOpportunity` özelliği, kaydırılan metin birden çok satıra kesildiğinde, kesme için hangi karakterlerin kullanılabileceğini belirler. Varsayılan değer olan `BreakOpportunity.AUTO`, sözcükler arasında ve kısa çizgilerde kesme gibi standart Unicode özelliklerini kullanır. `BreakOpportunity.ALL` değerinin kullanılması, tüm karakterlerin satır kesme olanağı olarak değerlendirilmesini sağlar ve bu, bir yolu izleyen metin gibi efektler oluşturmak için kullanışlıdır.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

Sekme durakları

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Metin bloğunda sekme durakları ayarlamak için, TabStop sınıfının örneklerini oluşturarak sekme duraklarını tanımlayın. `TabStop()` yapıcısının parametreleri, metnin sekme durağıyla nasıl hizalandığını belirtir. Bu parametreler, sekme durağının konumunu belirtir ve ondalık hizalaması için, dize olarak ifade edilen hizalama değerini belirtir. Genellikle bu değer bir ondalık işaretidir ancak virgül, dolar işareti veya örneğin Yen ya da Euro sembolü de olabilir. Aşağıdaki kod satırı, tab1 adında bir sekme durağı oluşturur.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

Bir metin bloğu için sekme durakları oluşturduktan sonra, bunları bir TextBlock örneğinin `tabStops` özelliğine atayın. Ancak `tabStops` özelliği bir Vektör gerektiğinden, ilk olarak Vektör öğesini oluşturmanız ve sekme duraklarını buna eklemeniz gerekir. Vektör, metin bloğuna bir sekme durağı kümesi atamanıza olanak sağlar. Aşağıdaki örnek bir `Vector<TabStop>` örneği oluşturur ve buna bir TabStop nesnesi kümesi ekler. Ardından sekme duraklarını bir TextBlock örneğinin `tabStops` özelliğine ayar.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Vektörler hakkında daha fazla bilgi için, bkz. “Dizilerle çalışma” sayfa 25.

Aşağıdaki örnek, her bir TabStop hizalama seçeneğinin etkisini gösterir.

```
package {

    import flash.text.engine.*;
    import flash.display.Sprite;

    public class TabStopExample extends Sprite
    {
        public function TabStopExample()
        {
            var format:ElementFormat = new ElementFormat();
            format.fontDescription = new FontDescription("Arial");
            format.fontSize = 16;

            var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
            tabStops.push(
                new TabStop(TabAlignment.START, 20),
                new TabStop(TabAlignment.CENTER, 140),
                new TabStop(TabAlignment.DECIMAL, 260, "."),
                new TabStop(TabAlignment.END, 380));
            var textBlock:TextBlock = new TextBlock();
            textBlock.content = new TextElement(
                "\tt1\tt2\tt3\tt4\n" +
                "\tThis line aligns on 1st tab\n" +
                "\t\t\t\tThis is the end\n" +
                "\tThe following fragment centers on the 2nd tab:\t\t\n" +
                "\t\tit's on me\t\t\n" +
                "\tThe following amounts align on the decimal point:\n" +
                "\t\t\t45.00\t\n" +
                "\t\t\t75,320.00\t\n" +
                "\t\t\t6,950.00\t\n" +
                "\t\t\t7.01\t\n", format);

            textBlock.tabStops = tabStops;
            var yPosition:Number = 60;
            var previousTextLine:TextLine = null;
            var textLine:TextLine;
            var i:int;
            for (i = 0; i < 10; i++) {
                textLine = textBlock.createTextLine(previousTextLine, 1000, 0);
                textLine.x = 20;
                textLine.y = yPosition;
                addChild(textLine);
                yPosition += 25;
                previousTextLine = textLine;
            }
        }
    }
}
```

Flash Text Engine örneği: Mizanpaj

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Bu programlama örneği, Flash Metin Motoru'nu basit bir gazete sayfası düzenlemek için kullanımdayken gösterir. Bu sayfa, büyük bir başlık, bir alt başlık ve birden çok sütun içeren bir gövde bölümü içerir.

İlk olarak bir FLA dosyası oluşturun ve aşağıdaki kodu varsayılan katmanın 2 numaralı karesine ekleyin:

```
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, bu örnekte denetleyici komut dosyasıdır. İçeriği belirler, harici bir stil sayfasından stil bilgilerini okur ve bu stilleri ElementFormat nesnelerine atar. Ardından başlığı, alt başlığı ve birden çok sütun içeren metin öğelerini oluşturur.

```
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, metin blokları oluşturmak için bir temel sınıf olarak kullanılır. Ayrıca, font boyutunu ve büyük/küçük harf durumunu değiştirmek için yardımcı program işlevleri de içerir.

```
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, FormattedTextBlock sınıfını genişletir ve başlık oluşturmak için kullanılır. Sayfa üzerinde tanımlanan bir boşluğun içine metin sığdırmak için bir işlev içerir.

```
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, birden çok sütun içeren bir tasarımda, metin biçimlendirmesini işler. Bir TextBlock nesnesinin nasıl esnek bir şekilde metin satırları oluşturmaya, biçimlendirmeye ve yerleştirmeye yarayan bir fabrika olarak kullanıldığını gösterir.

```
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);
        }

    }
}
```

Bölüm 23: Metin Mizanpajı Çerçevesini Kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Metin Mizanpajı Çerçevesi'ne Genel Bakış

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Metin Mizanpajı Çerçevesi (TLF) genişletilebilir bir ActionScript kütüphanesidir. TLF, Adobe® Flash® Player 10 ve Adobe® AIR® 1.5'teki metin motoru üzerine kuruludur. TLF web'de yenilikçi tipografi için gelişmiş tipografik ve metin mizanpajı özellikleri sunar. Çerçeve Adobe® Flex® veya Adobe® Flash® Professional ile birlikte kullanılabilir. Geliştiriciler mevcut bileşenleri kullanabilir veya genişletebilirler ya da kendi metin bileşenlerini oluşturmak için çerçeveyi kullanabilirler.

TLF aşağıdaki özelliklere sahiptir:

- Çiftyönlü metin, dikey metin ve Arapça, İbranice, Çince, Japonca, Korece, Tay Dili, Lao Dili, Vietnam Dili ve diğerlerini içeren 30'dan fazla yazma komut dosyası
- Birden fazla sütun ve bağlı kaplar arasında metin seçme düzenleme ve akışını sağlama
- Dikey metin, Tate-Chu-Yoko (dikey metin içinde yatay) ve Doğu Asya tipografisi için yaslayıcı
- Karakter aralığı, bitişik harfler, büyük/küçük dizgi karakteri, basamağın büyük/küçük karakter durumu, basamak genişliği, isteğe bağlı tireler
- Kesme, kopyalama, yapıştırma, geri alma ve düzenleme için standart klavye ve fare hareketleri
- Metin içeriğini, mizanpajı değiştirmek ve özel metin bileşenlerini işaretlemek ve oluşturmak için zengin geliştirici API'leri
- Özel işaretçileri ve numaralandırma biçimlerini içeren sağlam liste desteği
- Satır içi görüntüler ve konumlandırma kuralları

TLF, Flash Player 10'da sunulan Flash Metin Motoru (FTE) üzerine kurulu bir ActionScript 3.0 kütüphanesidir. FTE'ye, Flash Player 10 Uygulama Programlama Arabirimi'nin (API) parçası olan `flash.text.engine` paketinden erişilebilir.

Ancak Flash Player API'si, metin motoruna çok düşük düzeyde erişim sağlar; bu da bazı görevlerin buna oranla büyük miktarda kod gerektirebileceği anlamına gelir. TLF düşük düzeyli kodu daha basit API'lerin içine kapsüller. TLF ayrıca FTE tarafından tanımlanan temel yapı taşlarını kullanımı daha kolay bir sistemde düzenleyen kavramsal bir yapı sağlar.

FTE'nin aksine, TLF Flash Player'da yerleşik değildir. Onun yerine, tamamen ActionScript 3.0'da yazılan bir bağımsız bileşen kütüphanesidir. Çerçeve genişletilebilir olduğundan, belirli ortamlar için özelleştirilebilir. Hem Flash Professional hem de Flex SDK TLF çerçevesine dayalı bileşenler içerir.

Daha fazla Yardım konusu

"Flow" TLF biçimlendirme uygulaması

Karmaşık alfabe desteği

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

TLF karmaşık alfabe desteği sağlar. Karmaşık alfabe desteği sağdan sola komut dosyalarını görüntüleme ve düzenleme özelliğini içerir. TLF ayrıca Arapça ve İbranice gibi soldan sağa ve sağdan sola alfabelerin bir karışımını görüntüleme ve düzenleme özelliğini de sağlar. Çerçeve yalnızca Çince, Japonca ve Kore dili için dikey metin mizanpajını desteklemekle kalmaz, aynı zamanda tate-chu-yoko (TCY öğeleri) özelliğini de destekler. TCY öğeleri dikey metin öbeklerine gömülü yatay metin bloklarıdır. Aşağıdaki komut dosyaları desteklenir:

- Latin (İngilizce, İspanyolca, Fransızca, Vietnamca, vb.)
- Yunanca, Kiril, Ermenice, Gürcüce ve Etiyopya Dili
- Arapça ve İbranice
- Han kavram resimleri ve Kana (Çince, Japonca ve Kore Dili) ve Hangul Johab (Kore Dili)
- Tay dili, Lao dili ve Khmer dili
- Devanagari dili, Bengal dili, Gurmukhi dili, Malayalam dili, Telugu dili, Tamil dili, Gucerat dili, Oriya dili, Kannada dili ve Tibet dili
- Tifinag dili, Yi dili, Çeroki dili, Canadian Syllabics, Deseret, Shaw alfabesi, Vai, Tagalog dili, Hanunoo, Buhid ve Tagbanwa

Flash Professional ve Flex'te Metin Mizanpajı Çerçevesini Kullanma

Flash'ta özel bileşenler oluşturmak için doğrudan TLF sınıflarını kullanabilirsiniz. Ek olarak, Flash Professional CS5, TLF işlevselliğini kapsayan yeni bir sınıf sağlar: fl.text.TLFTextField ActionScript'te TLF'nin gelişmiş metin görüntüleme özelliklerini kullanan metin alanları oluşturmak için TLFTextField sınıfını kullanın. TextField sınıfıyla bir metin alanı oluşturduğunuz şekilde bir TLFTextField nesnesi oluşturun. Sonra, TLF sınıflarından gelişmiş biçimlendirme atamak için `textFlow` özelliğini kullanın.

Ayrıca metin aracını kullanarak sahne alanında TLFTextField örneği oluşturmak için de Flash Professional'ı kullanabilirsiniz. Ardından TLF sınıflarını kullanarak metin alanı içeriğinin biçimlendirmesini ve mizanpajını kontrol etmek için ActionScript'i kullanabilirsiniz. Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki TLFTextField sınıfına bakın.

Flex'te çalışıyorsanız TLF sınıflarını kullanın. Daha fazla bilgi için bkz. “Metin Mizanpajı Çerçevesini Kullanma” sayfa 431.

Metin Mizanpajı Çerçevesini Kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flex'te çalışıyor veya özel metin bileşenleri oluşturuyorsanız TLF sınıflarını kullanın. TLF tamamen textLayout.swc kütüphanesinin içinde bulunan bir ActionScript 3.0 kütüphanesidir. TLF kütüphanesi, on paket olarak düzenlenmiş yaklaşık yüz adet ActionScript 3.0 sınıfı ve arabirimi içerir. Bu paketler flashx.textLayout paketinin alt paketleridir.

Metin Mizanpajı Çerçevesi sınıfları

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

TLF sınıfları üç kategoride gruplandırılabilir:

- Veri yapıları ve biçimlendirme sınıfları
- Görüntü oluşturma sınıfları
- Kullanıcı etkileşimi sınıfları

Veri yapıları ve biçimlendirme sınıfları

Aşağıdaki paketler TLF için veri yapılarını ve biçimlendirme sınıflarını içerir:

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

TLF'nin ana veri yapısı, öğeler paketinde tanımlanan metin akışı hiyerarşisidir. Bu yapı içinde, biçim paketi ile metin akışlarına stiller ve nitelikler atayabilirsiniz. Ayrıca dönüştürme paketiyle, metnin veri yapısına nasıl içe aktarıldığını ve yapıdan nasıl dışa aktarıldığını da kontrol edebilirsiniz.

Görüntü oluşturma sınıfları

Aşağıdaki paketler TLF için görüntü oluşturma sınıflarını içerir:

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

Bu paketteki sınıflar Flash Player tarafından görüntülemek için metin oluşturma işlemini kolaylaştırır. Fabrika paketi, statik metin görüntülemek için basit bir yol sunar. Kap paketi, dinamik metin için görüntüleme kaplarını tanımlayan sınıfları ve arabirimleri içerir. Oluşturma paketi, kaplardaki dinamik metnin konumlandırılmasına ve görüntülenmesine yönelik teknikleri tanımlar.

Kullanıcı etkileşimi sınıfları

Aşağıdaki paketler TLF için kullanıcı etkileşimi sınıflarını içerir:

- flashx.textLayout.edit
- flashx.textLayout.operations
- flashx.textLayout.events

Düzenleme ve işlemler paketleri veri yapıları içinde saklanan metnin düzenlemesine izin vermek için kullanabileceğiniz sınıflar tanımlar. Events paketi olay işleme sınıfları içerir.

Metin Mizanpajı Çerçevesi ile metin oluşturmak için genel adımlar

Aşağıdaki adımlar Metin Mizanpajı Çerçevesi ile metin oluşturmanın genel sürecini açıklar:

1. Biçimlendirilmiş metni TLF veri yapılarına içe aktarın. Daha fazla bilgi için bkz. “TLF ile metin yapılandırma” sayfa 436 ve “TLF ile metin biçimlendirme” sayfa 440.
2. Metin için bir veya daha fazla bağlı görüntüleme nesnesi kabı oluşturun. Daha fazla bilgi için bkz. “TLF ile metin kaplarını yönetme” sayfa 442.

3 Veri yapıları içindeki metni kaplarla ilişkilendirin ve düzenleme ve kaydırma seçeneklerini ayarlayın. Daha fazla bilgi için bkz. "TLF ile metin seçimini, düzenlemeyi ve geri almayı etkinleştirme" sayfa 443.

4 Yeniden boyutlandırma (veya diğer) olayına karşılık olarak metnin yeniden akışını sağlamak üzere bir olay işleyicisi oluşturun. Daha fazla bilgi için bkz. "TLF ile olay işleme" sayfa 443.

Metin Mizanpajı Çerçevesi örneği: Haber mizanpajı

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Aşağıdaki örnek basit bir gazete sayfası düzenlemek için TLF kullanımını gösterir: Bu sayfa, büyük bir başlık, bir alt başlık ve birden çok sütun içeren bir gövde bölümü içerir:

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

TLFNewsLayout sınıfı iki metin kabı kullanır. Bir kap başlığı ve alt başlığı ve diğeri üç sütunlu gövde metnini görüntüler. Kolaylık için metin örneğin içine TLF Biçimlendirmesi metni olarak sabit bir şekilde kodlanmıştır. `headlineMarkup` değişkeni hem başlık hem de alt başlığı içerir ve `bodyMarkup` değişkeni gövde metnini içerir. TLF Biçimlendirmesi ile ilgili daha fazla bilgi için bkz. “TLF ile metin yapılandırma” sayfa 436.

Başladıktan bir süre sonra `onAddedToStage()` işlevi başlık metnini TLF'nin ana veri yapısı olan bir TextFlow object nesnesine içe aktarır:

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

Daha sonra, kap için bir Sprite nesnesi oluşturulur ve bir denetleyici oluşturulup kap ile ilişkilendirilir:

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

Biçimlendirmeyi, kaydırmayı ve diğer seçenekleri ayarlaması için denetleyici başlatılır. Denetleyici, metinin aktığı kabın sınırlarını tanımlayan geometri içerir. Biçimlendirme seçeneklerini bir TextLayoutFormat nesnesi içerir:

```
hContainerFormat = new TextLayoutFormat();
```

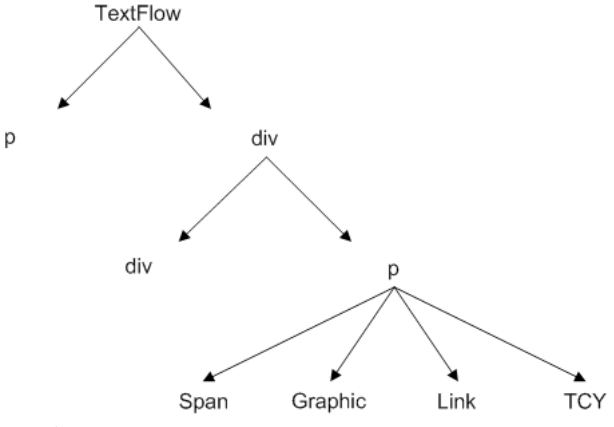
Denetleyici akış oluşturucusuna atanır ve işlev kabı görüntüleme listesine ekler. Gerçek kompozisyon ve kapların görüntülenmesi `resizeHandler()` yöntemine ertelenir. Aynı adım sıralaması gövde TextFlow nesnesini başlatmak için gerçekleştirilir.

`resizeHandler()` yöntemi kapları oluşturmak için uygun olan alanı ölçer ve buna göre kapları boyutlandırır. `compose()` yöntemine yapılan ilk çağrı başlık kabının uygun yüksekliğinin hesaplanmasına olanak verir. `resizeHandler()` yöntemi o zaman başlık kabını `updateAllControllers()` yöntemiyle yerleştirip görüntüleyebilir. Son olarak, `resizeHandler()` yöntemi, başlık kabının boyutunu gövde metni kabının yerleşimini ayarlamak için kullanır.

TLF ile metin yapılandırma

TLF metni temsil etmek için hiyerarşik bir ağaç kullanır. Ağaçtaki her düğüm, öğe paketinde tanımlanan bir sınıfın bir örneğidir. Örneğin, ağacın kök düğümü her zaman TextFlow sınıfının bir örneğidir. TextFlow sınıf, tam bir metin öyküsünü temsil eder. Öykü tek bir birim veya akış olarak algılanan bir metin veya diğer öğelerin koleksiyonudur. Tek bir hikaye görüntülenmek için birden fazla sütun veya metin kabı gerektirebilir.

Kök düğümün dışındaki öğeler, serbest bir şekilde XHTML öğelerini temel alır. Aşağıdaki şekil çerçevenin hiyerarşisini gösterir:



TextFlow Hiyerarşisi

Metin Mizanpajı Çerçevesi biçimlendirmesi

TLF Biçimlendirmesi ile ilgili işlem yaparken TLF'nin yapısını anlamak faydalıdır. TLF Biçimlendirmesi TLF'nin bir parçası olan metnin XML temsilidir. Çerçeve diğer XML biçimlerini desteklese de, TLF Biçimlendirmesi özellikle TextFlow hiyerarşisinin yapısına bağlı olması açısından benzersizdir. Bu işaretleme biçimini kullanarak dışa XML aktarırsanız, XML bu hiyerarşi korunarak dışa aktarılır.

TLF Biçimlendirmesi, bir TextFlow hiyerarşisinde metnin en uygun temsilini sağlar. Biçimlendirme dili TextFlow hiyerarşisinin her bir temel öğesi için etiket sağlar ve ayrıca TextLayoutFormat sınıfı içinde mevcut olan tüm biçimlendirme sınıfları için nitelik sağlar.

Aşağıdaki tablo TLF İşaretleyicisi'nde kullanılabilen etiketleri içerir.

Öğe	Açıklama	Alt Öğeler	Sınıf
textflow	İşaretleyicinin kök öğesi.	div, p	TextFlow
div	Bir TextFlow öğesindeki bir bölünme. Bir paragraf grubu içerebilir.	div, list, p	DivElement
p	Bir paragraf.	a, tcy, span, img, tab, br, g	ParagraphElement
a	Bir bağlantı.	tcy, span, img, tab, br, g	LinkElement
tcy	Yatay metin akışı (dikey TextFlow öğesinde kullanılır)	a, span, img, tab, br, g	TCYElement
span	Bir paragraf içinde bulunan bir metin öbeği.		SpanElement
img	Bir paragraf içindeki bir görüntü.		InlineGraphicElement
sekme	Bir sekme karakteri.		TabElement
br	Bir kesme karakteri. Paragrafta bir satırı bitirmek için kullanılır. Metin bir sonraki satırdan devam eder ancak aynı paragrafta kalır.		BreakElement
linkNormalFormat	Normal durumdaki bağlantılar için kullanılan biçimlendirme niteliklerini tanımlar.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	Fare bir bağlantı üzerindeyken etkin durumdaki bağlantılar için kullanılan biçimlendirme niteliklerini tanımlar.	TextLayoutFormat	TextLayoutFormat
linkHoverFormat	Fare bir bağlantının sınırları dahilinde olduğunda (üzerine geldiğinde), üzerine gelinen bağlantılar için kullanılan biçimlendirme niteliklerini tanımlar.	TextLayoutFormat	TextLayoutFormat
li	Bir list öğesi üyesi. Bir list öğesi içinde olması gerekir.	div, li, list, p	ListItemElement
list	Bir liste. Listeler yuvalanabilir veya birbirine bitişik olarak yerleştirilebilir. List öğelerine farklı etiketleme ve numaralandırma şemaları uygulanabilir.	div, li, list, p	ListElement
g	Bir grup öğesi. Bir paragraftaki öğeleri gruplandırmak için kullanılır. Paragraf düzeyinin altında öğeleri yuvalamanıza izin verir.	a, tcy, span, img, tab, br, g	SubParagraphGroupElement

Daha fazla Yardım konusu

TLF 2.0 Lists Markup (TLF 2.0 Listelerini Biçimlendirme)

TLF 2.0 SubParagraphGroupElements and typeName (TLF 2.0 SubParagraphGroupElements ve typeName)

Numaralı ve madde işaretli listeler kullanma

Metin denetimlerinize madde işaretli listeler eklemek için ListElement ve ListItemElement sınıflarını kullanabilirsiniz. Madde işaretli listeler yuvalanabilir ve farklı madde işaretleri (veya işaretçiler) ve otomatik numaralandırmanın yanı sıra ana hat stili numaralandırma kullanmak için özelleştirilebilir.

Metin akışlarınızda listeler oluşturmak için `<list>` etiketini kullanın. Daha sonra listedeki her liste öğesi için `<list>` etiketi içindeki `<li>` etiketlerini kullanın. ListMarkerFormat sınıfını kullanarak madde işaretlerinin görünümünü özelleştirebilirsiniz.

Aşağıdaki örnek basit listeler oluşturur:

```
<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>
```

Aşağıdaki örnekte gösterildiği gibi listeleri diğer listelerin içine yuvalayabilirsiniz:

```
<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>
```

Listedeki işaretçi türünü özelleştirmek için ListElement sınıfının `listStyleType` özelliğini kullanın. Bu özellik ListStyleType sınıfı tarafından tanımlanan herhangi bir değer olabilir (`check`, `circle`, `decimal` ve `box` gibi). Aşağıdaki örnek çeşitli işaretçi türleri ve özel sayaç artışlarıyla listeler oluşturur:

```
<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>
```

Sayacı tanımlamak için ListMarkerFormat sınıfını kullanırsınız. Sayacın artışını tanımlamanın yanı sıra, ayrıca `counterReset` özelliğiyle sıfırlayarak sayacı özelleştirebilirsiniz.

Listelerinizdeki işaretçilerin görünümünü ListMarkerFormat öğesinin `beforeContent` ve `afterContent` özelliklerini kullanarak daha fazla özelleştirebilirsiniz. Bu özellikler, işaretçinin içeriğinden önce ve sonra görüntülenen içeriğe uygulanır.

Aşağıdaki örnek işaretçiden önce “XX” dizesini ve işaretçiden sonra “YY” dizesini ekler:

```
<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>
```

`content` özelliği işaretçi biçiminin daha fazla özelleştirilmesini tanımlayabilir. Aşağıdaki örnek sıralı, büyük harfli Romen sayısal işaretçisini görüntüler:

```
<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>
```

Önceki örnekte gösterildiği gibi `content` özelliği ayrıca bir sonek ekleyebilir: işaretçiden sonra ancak `afterContent` öğesinden önce görüntülenen bir dize. Akışa XML içeriği sağlarken bu dizeyi eklemek için dizeyi tırnak işaretleri (`"<string>"`) yerine `&quote;` HTML varlıklarının içine sarın.

Daha fazla Yardım konusu

TLF 2.0 Lists Markup (TLF 2.0 Listelerini Biçimlendirme)

TLF'de dolgu kullanma

Her FlowElement öğesi, her öğenin içerik alanının konumunu ve içerik alanları arasındaki boşluğu denetlemek için kullandığınız dolgu özelliklerini destekler.

Bir öğenin toplam genişliği içeriğinin genişliği artı `paddingLeft` ve `paddingRight` özelliklerinin toplamıdır. Bir öğenin toplam yüksekliği içeriğinin yüksekliği artı `paddingTop` ve `paddingBottom` özelliklerinin toplamıdır.

Dolgu, kenarlık ile içeriğin arasındaki boşluktur. Dolgu özellikleri şunlardır: `paddingBottom`, `paddingTop`, `paddingLeft` ve `paddingRight`. Doldurma TextFlow nesnesinin yanı sıra aşağıdaki alt öğelere de uygulanabilir:

- div
- img
- li
- list
- p

Dolgu özellikleri span öğelerine uygulanamaz.

Aşağıdaki örnek TextFlow öğesinde dolgu özelliklerini ayarlar:

```
<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">
```

Dolgu özellikleri için geçerli değerler sayı (piksel olarak), “auto” veya “inherit” değerleridir. Varsayılan değer “auto” değeridir. Bu ListElement dışındaki tüm öğeler için otomatik olarak hesaplandığı ve 0 olarak ayarlandığı anlamına gelir. ListElements için `listAutoPadding` özelliğinin değerinin kullanıldığı listenin başlangıç tarafı dışında “auto” değeri 0'dır. `listAutoPadding` öğesinin varsayılan değeri 40'tır. Bu, listelere varsayılan bir girinti verir.

Dolgu özellikleri varsayılan olarak devralmaz. “auto” ve “inherit” değerleri FormatValue sınıfı tarafından tanımlanan sabitlerdir.

Doldurma özellikleri negatif değerler olabilir.

Daha fazla Yardım konusu

Padding changes in TLF 2.0 (TLF 2.0'daki dolgu değişiklikleri)

TLF ile metin biçimlendirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

flashx.textLayout.formats paketi metin akışı hiyerarşisi ağacında herhangi bir FlowElement öğesine biçim atamanıza izin veren arabirimler ve sınıflar içerir. Biçimlendirmeyi uygulamanın iki yolu vardır: Bağımsız olarak belirli bir biçim atayabilirsiniz veya özel bir biçimlendirme nesnesiyle bir grup biçimi eş zamanlı olarak atayabilirsiniz.

ITextLayoutFormat arabirimi bir FlowElement öğesine uygulanabilen tüm biçimleri içerir. Bazı biçimler, bir metin kabının veya paragrafının tamamına uygulanır, ancak mantıksal olarak karakterlere tek tek uygulanmaz. Örneğin, yaslama ve sekme durakları gibi biçimler tam paragraflara uygulanır, ancak bağımsız karakterlere uygulanamaz.

Özelliklerle FlowElement öğesine biçim atama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Doğrudan atama ile herhangi bir FlowElement öğesinde biçimleri ayarlayabilirsiniz. FlowElement sınıfı ITextLayoutFormat arabirimini uygular, bu nedenle FlowElement sınıfının herhangi bir alt sınıfı da o arabirimi uygulamalıdır.

Örneğin, aşağıdaki kod bir ParagraphElement örneğine bağımsız biçimlerin nasıl atanacağını gösterir:

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

Bir FlowElement öğesine TextLayoutFormat sınıfıyla biçim atama

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

TextLayoutFormat sınıfıyla bir FlowElement öğesine biçimler uygulayabilirsiniz. Bu sınıfı istediğiniz tüm biçimlendirme değerlerini içeren özel bir biçimlendirme nesnesi oluşturarak kullanabilirsiniz. Daha sonra bu nesneyi herhangi bir FlowElement nesnesinin `format` özelliğine atayabilirsiniz. TextLayoutFormat ve FlowElement öğelerinin ikisi de ITextLayoutFormat arabirimini uygular. Bu düzenleme iki sınıfın da kesin olarak aynı biçim özelliklerini içermesini sağlar.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki TextLayoutFormat sınıfına bakın.

Biçim devralma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Biçimler metin akışı hiyerarşisi aracılığıyla devralınır. Bir TextLayoutFormat örneğini alt öğelere sahip bir FlowElement örneğine atarsanız, çerçeve *basamaklama* adı verilen bir işlem başlatır. Bir basamaklama sırasında, çerçeve FlowElement öğenizden miras alan hiyerarşideki her düğümü tekrarlı bir şekilde inceler. Sonra, miras alınan değerlerin her biçimlendirme özelliğine atanıp atanmayacağına karar verir. Basamaklama sırasında şu kurallar geçerlidir:

1. Özellik değerleri yalnızca birinci dereceden bir ana öğeden (bazen üst öğe adı verilir) miras alınır.
2. Özellik değerleri, yalnızca bir özelliğin bir değere sahip olmadığı durumlarda miras alınır (başka bir deyişle değer `undefined` durumundadır).
3. Niteliğin değeri "inherit" olarak veya `flashx.textLayout.formats.FormatValue.INHERIT` sabitine ayarlanmamışsa, bazı nitelikler tanımsız durumdayken veri miras almaz.

Örneğin TextFlow düzeyinde `fontSize` değerini ayarlarsanız, ayar TextFlow içindeki tüm öğelere uygulanır. Başka bir deyişle, değerler metin akışı hiyerarşisine basamaklı olarak uygulanır. Ancak, doğrudan öğeye yeni bir değer atayarak belirli bir öğedeki değeri geçersiz kılabilirsiniz. Buna aykırı bir örnek olarak, TextFlow düzeyi için `backgroundColor` değerini ayarlarsanız, TextFlow öğesinin alt öğeleri o değeri miras almaz. `backgroundColor` özelliği bir basamaklama sırasında üst özelliklerinden miras almayan bir özelliktir. Her bir alt öğedeki `backgroundColor` özelliğini `flashx.textLayout.formats.FormatValue.INHERIT` olarak ayarlayarak bu davranışı geçersiz kılabilirsiniz..

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki TextLayoutFormat sınıfına bakın.

TLF ile metin içe aktarma ve dışa aktarma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

flashx.textLayout.conversion* paketindeki TextConverter sınıfıTLF'ye metin içe aktarmanıza ve TLF'den metin dışa aktarmanıza izin verir. Metni SWF dosyasında derlemek yerine çalışma zamanında yüklemeyi planlıyorsanız bu sınıfı kullanın. Bu sınıfı ayrıca TextFlow örneği içinde saklanan metni bir String veya XML nesnesine dışa aktarmak için de kullanabilirsiniz.

İçe ve dış aktarma işlemleri, basit prosedürlerdir. Her ikisi de TextConverter sınıfının parçası olan `export()` yöntemini veya `importToFlow()` yöntemini çağırırsınız. Her iki yöntem de statiktir; başka bir deyişle, yöntemleri TextConverter sınıfının bir örneğinde değil, TextConverter sınıfında çağırırsınız.

flashx.textLayout.conversion paketindeki sınıflar metni saklamayı seçtiğiniz konum konusunda önemli derecede esneklik sağlar. Örneğin, metninizi veritabanında saklarsanız, metni görüntülemek için çerçeveye içe aktarabilirsiniz. Ardından metni değiştirmek ve değiştirilmiş metni veritabanınıza dışa aktarmak için the flashx.textLayout.edit paketindeki sınıfları kullanabilirsiniz.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki flashx.textLayout.conversion sınıfına bakın.

TLF ile metin kaplarını yönetme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Metin TFL veri yapılarına depolandığında Flash Player bunu görüntüleyebilir. Akış hiyerarşisinde depolanan metin Flash Player'ın görüntüleyebileceği bir biçime dönüştürülmelidir. TLF bir akıştan görüntüleme nesneleri oluşturmak için iki yol sunar. Daha basit olan ilk yaklaşım statik metinleri görüntülemek için uygundur. Daha karmaşık olan ikinci yaklaşım seçilebilir ve düzenlenebilir dinamik metin oluşturmanıza izin verir. Her iki durumda da metin, sonuç olarak Flash Player 10'da flash.text.engine paketinin parçası olan TextLine sınıfının örneklerine dönüştürülür.

Statik metin oluşturma

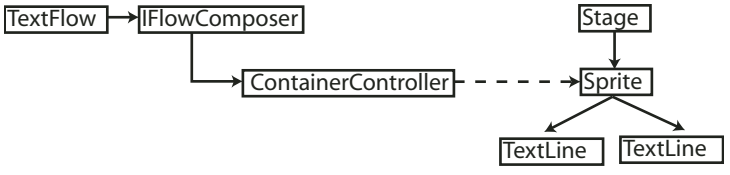
Basit yaklaşım, flashx.textLayout.factory paketinde bulunabilen TextFlowTextLineFactory sınıfını kullanır. Bu yaklaşımın avantajı, basit olmasının yanı sıra, FlowComposer yaklaşımından daha küçük bir bellek izine sahip olmasıdır. Bu yaklaşım kullanıcının düzenlemesi, seçmesi veya kaydırması gerekmeyen statik metin için önerilebilir.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki TextFlowTextLineFactory sınıfına bakın.

Dinamik metin ve kaplar oluşturma

Metnin görüntüsü üzerinde TexFlowTextLineFactory tarafından sağlanandan daha fazla kontrol istiyorsanız akış oluşturucu kullanın. Örneğin, akış oluşturucusuyla kullanıcılarınız metni seçebilir ve düzenleyebilir. Daha fazla bilgi için bkz. “TLF ile metin seçimini, düzenlemeyi ve geri almayı etkinleştirme” sayfa 443.

Akış oluşturucusu, flashx.textLayout.compose paketindeki StandardFlowComposer sınıfının bir örneğidir. Akış oluşturucusu TextFlow öğesinin TextLine örneklerine dönüşümünü ve bu TextLine örneklerinin bir veya daha fazla kaba yerleştirilmesini yönetir.



Bir IFlowComposer, sıfır veya daha fazla ContainerController öğesine sahiptir.

Her TextFlow nesnesi, IFlowComposer arabirimini uygulayan bir karşılık gelen nesneye sahiptir. IFlowComposer nesnesine `TextFlow.flowComposer` özelliğinden erişilebilir. Bu özellik ile IFlowComposer arabirimi tarafından tanımlanan yöntemleri çağırabilirsiniz. Bu yöntemler metni bir veya daha fazla kapla ilişkilendirmenize ve metni bir kapta görüntülemeye hazırlamanıza izin verir.

Kap, DisplayObjectContainer sınıfının bir alt sınıfı olan Sprite sınıfının bir örneğidir. Bu sınıfların her ikisi de Flash Player görüntüleme listesi API'sinin parçasıdır. Bir kap, TextLineFactory sınıfı ile kullanılan daha gelişmiş bir sınırlama dikdörtgeni formudur. Sınırlama Rectangle öğesi gibi, kap da TextLine örneklerinin göründüğü alanı tanımlar. Sınırlama dikdörtgeninin aksine, bir kabın karşılık gelen “denetleyici” nesnesi vardır. Denetleyici bir kap veya kap grubu için kaydırmayı, kompozisyonu, bağlamayı, biçimlendirmeyi ve olay işlemeyi yönetir. Her kapta flashx.textLayout.container paketinde ContainerController sınıfının bir örneği olan karşılık gelen bir denetleyici nesnesi vardır.

Metin görüntülemek için kabı yönetmek ve kabı bir akış oluşturucusuyla ilişkilendirmek için controller nesnesi oluşturun. Kabı ilişkilendirdikten sonra, görüntülenebilmesi için metni oluşturun. Buna bağlı olarak, kaplar iki duruma sahiptir: kompozisyon ve görüntüleme. Metin oluşturma, metin akışı hiyerarşisindeki metni TextLine örneklerine dönüştürme ve bu örneklerin kaba sığıp sığmayacağını hesaplama işlemidir. Görüntüleme, Flash Player görüntüleme listesini güncelleme işlemidir.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda IFlowComposer, StandardFlowComposer ve ContainerController öğelerine bakın.

TLF ile metin seçimini, düzenlemeyi ve geri almayı etkinleştirme

Flash Player 9.0 ve üstü, Adobe 1.0 ve üstü

Metin seçme ve düzenleme yeteneği metin akışı düzeyinde denetlenir. TextFlow sınıfının her örneğiyle ilişkili bir etkileşim yöneticisi vardır. Bir TextFlow nesnesinin etkileşim yöneticisine nesnenin `TextFlow.interactionManager` özelliğinden erişebilirsiniz. Metin seçimini etkinleştirmek için, SelectionManager sınıfının bir örneğini `interactionManager` özelliğine atayın. Hem metin seçimi hem de düzenlemeyi etkinleştirmek için, SelectionManager sınıfının bir örneği yerine EditManager sınıfının bir örneğini atayın. Geri alma işlemlerini etkinleştirmek için, UndoManager sınıfının bir örneğini oluşturun ve EditManager öğesine ilişkin yapıcıyı çağırırken onu bir argüman olarak dahil edin. UndoManager sınıfı, kullanıcının en son düzenleme etkinliklerinin geçmişini tutar ve kullanıcının belirli düzenlemeleri geri almasına veya yinelemesine izin verir. Bu üç sınıfın üçü de düzenleme paketinin birer parçasıdır.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda SelectionManager, EditManager ve UndoManager öğelerine bakın.

TLF ile olay işleme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

TextFlow aşağıdakiler de dahil olmak üzere bir çok durumda olay gönderir:

- Metin veya mizanpaj değiştiğinde
- Bir işlem başlamadan önce ve işlem tamamlandıktan sonra
- Bir FlowElement nesnesinin durumu değiştiğinde
- Bir oluşturma işlemi tamamlandığında

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki flashx.textLayout.events sınıfına bakın.

Daha fazla Yardım konusu

TLF FlowElement and LinkElement Events and EventMirrors (TLF FlowElement ve LinkElement Events ve EventMirrors)

Metin içindeki görüntüleri konumlandırma

Metin içinde InlineGraphicElement öğesini konumlandırmak için aşağıdaki özellikleri kullanırsınız:

- InlineGraphicElement sınıfının `float` özelliği
- FlowElement sınıfının `clearFloats` özelliği

`float` özelliği grafiğin ve çevresindeki metnin yerleşimini denetler. `clearFloats` özelliği paragraf öğelerinin yerleşimini `float` öğesine göre denetler.

Bir metin öğesinin içindeki görüntünün konumunu denetlemek için `float` özelliğini kullanırsınız. Aşağıdaki örnek bir paragrafa görüntü ekler ve metnin sağ tarafta sarılmasını sağlamak için sola hizalar:

```
<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>
```

`float` özelliği için geçerli olan değerler şunlardır: “left”, “right”, “start”, “end” ve “none”. Float sınıfı bu sabitleri tanımlar. Varsayılan değer “none” değeridir.

`clearFloats` özelliği normalde görüntünün etrafını saracak alt paragrafların başlangıç konumunu ayarlamak istediğiniz durumlarda faydalıdır. Örneğin, ilk paragraftan daha büyük bir görüntüye sahip olduğunuzu varsayın. İkinci paragrafın görüntüden *sonra* başladığından emin olmak için `clearFloats` özelliğini ayarlayın.

Aşağıdaki örnekte ilk paragraftaki metinden daha uzun olan bir görüntü kullanılmıştır. İkinci paragrafın metin bloğundaki görüntüden başlamasını sağlamak için bu örnek `clearFloats` özelliğini ikinci paragrafta “end” olarak ayarlar.

```
<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>
```

`clearFloats` özelliği için geçerli olan değerler şunlardır: “left”, “right”, “end”, “start”, “none” ve “both”. ClearFloats sınıfı bu sabitleri tanımlar. Ayrıca `clearFloats` özelliğini FormatValue sınıfı tarafından tanımlanan “inherit” değerine de ayarlayabilirsiniz. Varsayılan değer “none” değeridir.

Daha fazla Yardım konusu

TLF Floats (TLF Kaydırmaları)

Bölüm 24: Sesle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript, sarmal, etkileşimli uygulamalar oluşturmak üzere tasarlanmıştır ve güçlü sarmal özellikteki uygulamalarda genellikle gözden kaçan önemli bir öğe sestir. Video oyununa ses efektleri ve uygulama kullanıcı arabirimine ses geribildirimi ekleyebilir, hatta uygulamanın merkezinde yalnızca ses bulunacak şekilde Internet üzerinden yüklenen mp3 dosyalarını çözümleyen bir program oluşturabilirsiniz.

Harici ses dosyaları yükleyebilir ve bir SWF'de gömülü olan seslerle çalışabilirsiniz. Sesi denetleyebilir, ses bilgilerinin görsel temsillerini oluşturabilir ve kullanıcı mikrofonundan ses yakalayabilirsiniz.

Daha fazla Yardım konusu

flash.media paketi

flash.events.SampleDataEvent

Sesle çalışmanın temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bilgisayarlar dijital sesi (ses bilgilerinin bilgisayar temsili) yakalayıp kodlayabilir ve bu sesleri depolayıp hoparlörlerden çalmak üzere alabilir. Adobe® Flash® Player veya Adobe® AIR™ ve ActionScript uygulamasını kullanarak ses çalabilirsiniz.

Ses verisi dijital biçime dönüştürüldüğünde, ses düzeyi ve stereo veya stereo olmayan ses gibi çeşitli özellikler içerir. ActionScript'te bir ses çaldığınızda, bu özellikleri de ayarlayabilirsiniz—ses düzeyini artırabilir veya sesi belirli bir yönden geliyormuş gibi ayarlayabilirsiniz.

ActionScript'te bir sesi denetleyebilmeniz için, öncelikle ses bilgilerinin Flash Player veya AIR uygulamasına yüklenmesi gerekir. ActionScript kullanarak ses verileriyle çalışmak için ses verilerini Flash Player veya AIR uygulamasına alabilmenizin beş yolu vardır.

- SWF'ye mp3 gibi harici bir ses dosyası yükleme.
- Oluşturulurken ses bilgisini doğrudan SWF dosyasına gömme.
- Kullanıcının bilgisayarına bağlı mikrofondan ses yakalama.
- Bir sunucudan ses akışı gerçekleştirme.
- Sesi dinamik olarak oluşturma ve oynatma.

Harici bir ses dosyasından ses verisi yüklediğinizde, ses verisinin kalanı yüklenmeye devam ederken, ses dosyasının başını çalmaya başlayabilirsiniz.

Dijital sesi kodlamak için kullanılan çeşitli ses dosyası biçimleri olsa da, ActionScript 3.0, Flash Player ve AIR uygulamaları, mp3 biçiminde saklanan ses dosyalarını destekler. Bu uygulamalar, WAV veya AIFF gibi diğer biçimlerdeki ses dosyalarını doğrudan yükleyemez ya da oynatamaz.

ActionScript'te sesle çalışırken, flash.media paketindeki birçok sınıfla çalışırsınız. Sound sınıfı, bir ses dosyası yükleyerek veya ses verilerini örnekleyen bir olaya işlev atayıp daha sonra çalma işlemini başlatarak ses bilgilerine erişim elde etmek için kullandığınız sınıftır. Ses çalmayı başlattıktan sonra, Flash Player ve AIR uygulaması SoundChannel nesnesine erişmenizi sağlar. Yüklediğiniz ses dosyası, kullanıcı bilgisayarında çalabileceğiniz birçok sesten yalnızca biri olabileceğinden, çalan seslerin her biri kendi SoundChannel nesnesini kullanır; bir araya getirilmiş tüm SoundChannel nesnelerinin birleştirilmiş çıktısı, bilgisayarın hoparlörlerinden gerçekte çalınan şeydir. Sesin özelliklerini denetlemek ve ses çalmasını durdurmak için bu SoundChannel örneğini kullanırsınız. Son olarak, birleştirilmiş sesi denetlemek isterseniz, SoundMixer sınıfı karıştırılmış çıktı üzerinde denetim elde etmenizi sağlar.

ActionScript'te sesle çalışırken daha özel görevler gerçekleştirmek için başka birçok sınıf da kullanabilirsiniz; tüm sesle ilgili sınıflar hakkında daha fazla bilgi almak için, bkz. “Ses mimarisini anlama” sayfa 446.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesinde, karşınıza çıkabilecek olan önemli terimler bulunmaktadır:

Genlik Ses dalga formundaki bir noktanın sıfır veya eşitlik çizgisinden itibaren olan uzaklığı.

Bit hızı Ses dosyasının her bir saniyesi için kodlanan veya akışa alınan veri miktarı. mp3 dosyaları için, bit hızı genellikle saniyede bin bit değeri (kbps) olarak belirtilir. Bit hızının yüksek olması genellikle ses dalgasının daha yüksek kalitede olduğu anlamına gelir.

Arabelleğe alma Ses verilerinin çalınmadan önce alınıp depolanması.

mp3 MPEG-1 Ses Katmanı 3 veya mp3, sık kullanılan bir sıkıştırma biçimidir.

Yatay kaydırma Stereo ses alanında sol ve sağ kanallar arasında ses sinyalinin konumlandırılması.

Zirve Dalga formundaki en yüksek nokta.

Örnekleme hızı Dijital sinyal oluşturmak için analog ses sinyalinden alınan saniyede örnek sayısını tanımlar. Standart kompakt diskin örnekleme hızı is 44,1 kHz veya saniyede 44.100 örnektir.

Akışa alma Dosyanın kalan kısmı sunucudan yüklenmeye devam ederken, ses dosyasının veya video dosyasının ilk kısımlarını oynatma işlemi.

Ses Sesin yüksekliği.

Dalga formu Ses sinyalinin zaman içinde değişen büyüklüğünü gösteren grafik şekli.

Ses mimarisini anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamalarınız beş ana kaynaktan ses verilerini yükleyebilir:

- Çalışma zamanında yüklenen harici ses dosyaları
- Uygulamanın SWF dosyasında gömülü olan ses kaynakları
- Kullanıcının sistemine takılı bir mikrofondan gelen ses verileri
- Flash Media Server gibi uzak bir medya sunucusundan akışa alınan ses verileri
- `sampleData` olay işleyicisi kullanılarak dinamik şekilde oluşturulan ses verileri

Ses verileri oynatılmadan önce tamamen yüklenebilir veya akışa alınabilir, başka bir deyişle, yüklenmeye devam ederken çalınabilir.

ActionScript 3.0 ses sınıfları, mp3 biçiminde saklanan ses dosyalarını destekler. Bu uygulamalar, WAV veya AIFF gibi diğer biçimlerdeki ses dosyalarını doğrudan yükleyemez veya oynatamaz. Ancak, Flash Player 9.0.115.0'dan itibaren, NetStream sınıfı kullanılarak AAC ses dosyaları yüklenip oynatılabilir. Bu, video içeriğinin yüklenip oynatılması için kullanılan teknikle aynıdır. Bu teknikle ilgili daha fazla bilgi almak için, bkz. "Videoyla çalışma" sayfa 478.

Adobe Flash Professional'ı kullanarak, WAV veya AIFF ses dosyalarını içe aktarabilir ve sonra bunları mp3 biçiminde uygulamanızın SWF dosyalarına gömebilirsiniz. Flash Geliştirme aracı ayrıca dosya boyutunu küçültmek için, gömülü ses dosyalarını sıkıştırmanıza da olanak sağlar ancak bu, ses kalitesinde kayba yol açabilir. Daha fazla bilgi için "Sesleri İçe Aktarma", *Flash'ı kullanma*.

ActionScript 3.0 ses mimarisi, flash.media paketindeki şu sınıflardan yararlanır.

Sınıf	Açıklama
flash.media.Sound	Sound sınıfı, ses yüklemesini işler, temel ses özelliklerini yönetir ve ses çalınmasını başlatır.
flash.media.SoundChannel	Bir uygulama bir Sound nesnesini oynattığında, oynatmayı denetlemek için yeni bir Channel nesnesi oluşturulur. SoundChannel nesnesi, sesin sol ve sağ oynatma kanallarının ses düzeyini kontrol eder. Çalınan her ses kendi SoundChannel nesnesine sahiptir.
flash.media.SoundLoaderContext	SoundLoaderContext sınıfı, bir ses yüklenirken kaç saniye arabelleğe alma uygulanacağını ve Flash Player veya AIR uygulamasının bir dosya yüklerken sunucudan ilke dosyası arayıp aramayacağını belirtir. SoundLoaderContext nesnesi, `Sound.load()` yöntemine parametre olarak kullanılır.
flash.media.SoundMixer	SoundMixer sınıfı, oynatmayı ve bir uygulamadaki tüm sesler için geçerli olan güvenlik özelliklerini denetler. Efektte, birden çok ses kanalı ortak bir SoundMixer nesnesi üzerinden karıştırılır, böylece SoundMixer nesnesindeki özellik değerleri, geçerli olarak oynatılan tüm SoundChannel nesnelerini etkiler.
flash.media.SoundTransform	SoundTransform sınıfı, ses düzeyini ve yatay kaydırmayı denetleyen değerleri içerir. SoundTransform nesnesi tek bir SoundChannel nesnesine, genel SoundMixer nesnesine veya bir Microphone nesnesine uygulanabilir.
flash.media.ID3Info	ID3Info nesnesi, daha çok mp3 ses dosyalarında saklanan ID3 meta veri bilgilerini temsil eden özellikleri içerir.
flash.media.Microphone	Microphone sınıfı, kullanıcının bilgisayarına takılı bir mikrofonu veya başka bir ses girdisi aygıtını temsil eder. Mikrofondan gelen ses girdisi, yerel hoparlörlere yönlendirilebilir veya uzak bir sunucuya gönderilebilir. Microphone nesnesi, kazanımı, örnekleme hızını ve kendi ses akışının diğer özelliklerini denetler.
flash.media.AudioPlaybackMode	AudioPlaybackMode sınıfı, SoundMixer sınıfının `audioPlaybackMode` özelliği için sabitleri tanımlar.

Yüklenen ve çalınan her ses için, kendi Sound sınıfı ve SoundChannel sınıfı örneği gerekir. Daha sonra oynatma sırasında genel SoundMixer sınıfı tarafından birden çok SoundChannel örneğinin çıktısı birbiriyle karıştırılır.

Sound, SoundChannel ve SoundMixer sınıfları, bir mikrofondan veya Flash Media Server gibi bir akış medyası sunucusundan alınan ses verileri için kullanılmaz.

Harici ses dosyalarını yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Her Sound sınıfı örneği, belirli bir ses kaynağını yüklemek ve bu ses kaynağının oynatılmasını tetiklemek için bulunur. Bir uygulama, Sound nesnesini birden çok ses yüklemek için yeniden kullanamaz. Yeni bir ses kaynağını yüklemek isterse, yeni bir Sound nesnesi oluşturması gerekir.

Bir düğmeye eklenecek tıklatma sesi gibi küçük bir ses dosyası yüklüyorsanız, aşağıda gösterildiği gibi uygulamanız yeni bir Sound oluşturup bunun otomatik olarak ses dosyasını yüklemesini sağlayabilir:

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

`Sound()` yapıcısı, bir URLRequest nesnesini birinci parametresi olarak kabul eder. URLRequest parametresi için bir değer verilirse, yeni Sound nesnesi, belirtilen ses kaynağını otomatik olarak yüklemeye başlar.

En basit bir durumda uygulamanızın, sesin yükleme ilerlemesine dikkat etmesi ve yükleme sırasında hata olup olmadığını izlemesi gerekir. Örneğin, tıklatma sesinin boyutu büyük olursa, kullanıcı sesi tetikleyen düğmeyi tıklattığında ses tamamen yüklenmeyebilir. Yüklenmeyen bir sesin çalınmaya çalışılması da çalışma zamanı hatasına neden olabilir. Kullanıcıların ses çalmasını başlatacak eylemleri uygulamasına izin vermeden önce, sesin tamamen yüklenmesini beklemek daha güvenli olacaktır.

Sound nesnesi, ses yükleme işlemi sırasında birçok farklı olay dağıtır. Uygulamanız, yükleme ilerlemesini izlemek ve çalınmadan önce sesin tamamen yüklendiğinden emin olmak için bu olayları dinleyebilir. Aşağıdaki tabloda, Sound nesnesi tarafından gönderilebilen olaylar listelenmektedir.

Olay	**Açıklama**
open (`Event.OPEN`)	Ses yükleme işlemi başlamadan hemen önce gönderilir.
progress (`ProgressEvent.PROGRESS`)	Dosyadan veya akıştan veri alındığında ses yükleme işlemi sırasında düzenli aralıklarla gönderilir.
id3 (`Event.ID3`)	Bir mp3 sesi için ID3 verisi kullanılabilir olduğunda gönderilir.
complete (`Event.COMPLETE`)	Tüm ses kaynağının verisi yüklendiğinde gönderilir.
ioError (`IOErrorEvent.IO_ERROR`)	Ses dosyasının yeri bulunamadığında veya tüm ses verisi alınmadan önce yükleme işlemi kesintiye uğradığında gönderilir.

Aşağıdaki kod, ses yüklemesi sona erince sesin nasıl çalındığını gösterir:

```
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();
}
```

İlk olarak, kod örneği, URLRequest parametresi için başlangıç değeri vermeden yeni bir Sound nesnesi oluşturur. Daha sonra, Sound nesnesinden `Event.COMPLETE` olayını dinler ve böylece tüm ses verileri yüklendiğinde `onSoundLoaded()` yönteminin çalıştırılmasını sağlar. Ardından, ses dosyası için yeni bir URLRequest değeriyle `Sound.load()` yöntemini çağırır.

Ses yüklemesi tamamlandığında `onSoundLoaded()` yöntemi çalıştırılır. Event nesnesinin `target` özelliği, Sound nesnesine bir başvurudur. Sound nesnesinin `play()` yönteminin çağrılması daha sonra ses çalınmasını başlatır.

Ses yükleme işlemini izleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Ses dosyaları çok büyük olabilir ve bu dosyaların yüklenmesi uzun zaman alabilir. Flash Player ve AIR, sesler tamamen yüklenmeden uygulamanızın sesleri çalmasına olanak sağlarken, kullanıcılara ses verisinin ne kadarının yüklendiğine ve ne kadarının çalındığına dair gösterge sağlamak isteyebilirsiniz.

Sound sınıfı, ses yüklemesinin ilerlemesinin görüntülenmesini nispeten kolaylaştıran iki olay gönderir: `ProgressEvent.PROGRESS` ve `Event.COMPLETE`. Aşağıdaki örnek, yüklenmekte olan seslerle ilgili ilerleme bilgilerinin görüntülenmesi için bu olayların nasıl kullanıldığını gösterir:

```
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);
}
```

Bu kod ilk olarak bir Sound nesnesi oluşturur ve daha sonra `ProgressEvent.PROGRESS` ve `Event.COMPLETE` olayları için bu nesneye dinleyiciler ekler. `Sound.load()` yöntemi çağrılıp ses dosyasından ilk veriler alındıktan sonra, bir `ProgressEvent.PROGRESS` olayı gerçekleşerek `onSoundLoadProgress()` yöntemini tetikler.

Yüklenen ses verisinin yüzdesi, ProgressEvent nesnesinin `bytesLoaded` özelliğinin `bytesTotal` özelliğine bölümüne eşittir. Aynı `bytesLoaded` ve `bytesTotal` özellikleri, Sound nesnesinde de kullanılabilir. Yukarıdaki örnek, ses yükleme ilerlemesiyle ilgili mesajları gösterir ancak Adobe Flex çerçevesiyle veya Adobe Flash geliştirme aracıyla gelen bileşenler gibi ilerleme çubuğu bileşenlerini güncellemek için `bytesLoaded` ve `bytesTotal` değerlerini kullanabilirsiniz.

Bu örnek ayrıca bir uygulamanın ses dosyaları yüklenirken bir hatayı nasıl tanıyıp yanıtlayabildiğini gösterir. Örneğin, belirli bir ada sahip ses dosyası bulunamazsa, Sound nesnesi tarafından `Event.IO_ERROR` olayı gönderilir. Önceki kodda, `onIOError()` yöntemi çalıştırılır ve hata oluştuğunda kısaca hata mesajını görüntüler.

Gömülü seslerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici bir dosyadan ses yüklemek yerine gömülü seslerin kullanılması, düğme tıklatıldığında çalınan sesler gibi, uygulamanızın kullanıcı arabiriminde gösterge olarak kullanılan küçük sesler için daha kullanışlıdır.

Uygulamanıza bir ses dosyası gömdüğünüzde, sonuçta elde dilen SWF dosyasının boyutu, ses dosyasının boyutu kadar artar. Başka bir deyişle, uygulamanıza büyük ses dosyalarının gömülmesi, SWF dosyanızın boyutunu istenmeyen bir boyuta artırabilir.

Uygulamanızın SWF dosyasına ses dosyası gömme yöntemi, geliştirme ortamınıza göre değişiklik gösterir.

Flash'ta gömülü ses dosyasını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash geliştirme aracı, sesleri birçok ses biçiminde içe aktarmanıza ve Kütüphanenizde semboller olarak saklamanıza olanak sağlar. Daha sonra bunları zaman çizelgesinde karelere veya düğme durumu karelerine atayabilir, Davranışlar ile kullanabilir ya da doğrudan ActionScript kodunda kullanabilirsiniz. Bu bölümde, Flash geliştirme aracıyla ActionScript kodunda gömülü seslerin nasıl kullanıldığı açıklanmaktadır. Flash'ta gömülü sesleri kullanmanın diğer yolları hakkında bilgi almak için, bkz. “Sesleri İçe Aktarma”, *Flash'ı kullanma*.

Flash geliştirme aracını kullanarak bir ses dosyası gömmek için:

1. Dosya > İçe Aktar > Kütüphaneye Aktar seçeneklerini belirleyin ve sonra bir ses dosyasını seçip içe aktarın.
2. Kütüphane panelinde içe aktarılan dosyanın adını sağ tıklatın ve Özellikler seçeneğini belirleyin. ActionScript için Dışa Aktar onay kutusunu tıklatın.
3. Sınıf alanına, ActionScript'te bu gömülü sesi ifade ederken kullanılacak bir ad girin. Bu, varsayılan olarak, bu alandaki ses dosyasının adını kullanır. “DrumSound.mp3” adında olduğu gibi dosya adında bir nokta varsa, bunu “DrumSound” gibi değiştirmeniz gerekir; ActionScript, sınıf adında nokta karakteri olmasına izin vermez. Temel Sınıf alanında flash.media.Sound öğesinin gösterilmesi gerekir.
4. Tamam'ı tıklatın. Bu sınıfın tanımının sınıf yolunda bulunamadığını belirten bir iletişim kutusu görebilirsiniz. Tamam'ı tıklatıp devam edin. Uygulamanızın sınıf yolundaki herhangi bir sınıfın adıyla eşleşmeyen bir sınıf adı girdiyseniz, flash.media.Sound sınıfından miras alan yeni bir sınıf otomatik olarak sizin için oluşturulur.
5. Gömülü sesi kullanmak için, ActionScript'te o sesin sınıf adına başvurursunuz. Örneğin, aşağıdaki kod başlatılarak otomatik olarak oluşturulan DrumSound sınıfının yeni bir örneğini oluşturur:

   ```
   var drum:DrumSound = new DrumSound();
   var channel:SoundChannel = drum.play();
   ```

   DrumSound, flash.media.Sound sınıfının bir alt sınıfıdır, bu nedenle de yukarıda gösterilen `play()` yöntemi de dahil olmak üzere, Sound sınıfının yöntemlerini ve özelliklerini miras alır.

Flex'te gömülü ses dosyasını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flex uygulamasında ses varlıklarını gömmenin birçok yolu vardır, örneğin:

- Bir komut dosyasında `[Embed]` meta veri etiketini kullanma

- Button veya SoundEffect gibi bir bileşenin özelliği olarak gömülü bir varlığı atamak için MXML'de `@Embed` direktifini kullanarak.
- CSS dosyasında `@Embed` direktifini kullanarak

Bu bölümde birinci seçenek açıklanmaktadır: `[Embed]` meta veri etiketini kullanarak Flex uygulamasında ActionScript koduna sesleri gömme.

ActionScript koduna bir varlık gömmek için, `[Embed]` meta veri etiketini kullanın

Ses dosyasını ana kaynak klasörüne veya projenizin oluşturma yolundaki başka bir klasöre yerleştirin. Derleyici bir Embed meta veri etiketiyle karşılaşırsa, sizin için gömülü varlık sınıfını oluşturur. `[Embed]` meta veri etiketinden hemen sonra bildirdiğiniz bir Class veri türündeki değişken üzerinden sınıfa erişebilirsiniz.

Aşağıdaki kod, smallSound.mp3 adındaki bir sesi gömer ve o sesle ilişkilendirilmiş gömülü varlık sınıfının bir başvurusunu saklamak için `soundClass` adındaki bir değişkeni kullanır. Kod daha sonra gömülü varlık sınıfının bir örneğini oluşturur, Sound sınıfının bir örneği olarak çevirir ve o örnekte `play()` yöntemini çağırır:

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

Flex bileşeninin bir özelliğini ayarlamak üzere gömülü sesin kullanılması için, gömülü sesin Sound sınıfının bir örneği olarak değil, mx.core.SoundAsset sınıfının bir örneği olarak çevrilmesi gerekir. SoundAsset sınıfını kullanan benzer bir örnek için, ActionScript'i Öğrenme bölümündeki "Gömülü varlık sınıfları" başlığını inceleyin.

Akış ses dosyalarıyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir ses dosyası veya video dosyası, verileri yüklenirken oynatılıyorsa, *akışa alınıyor* demektir. Uzak bir sunucudan yüklenen harici ses dosyaları genellikle kullanıcı sesi dinlemeden önce tüm ses verilerinin yüklenmesini beklemek zorunda kalmayacak şekilde akışa alınır.

`SoundMixer.bufferTime` özelliği, Flash Player veya AIR uygulamasının ses çalınmasına izin vermeden önce toplaması gereken ses verisinin milisaniye sayısını temsil eder. Başka bir deyişle, `bufferTime` özelliği 5000 olarak ayarlanırsa, Flash Player veya AIR uygulaması ses çalınmaya başlamadan önce ses dosyasından en az 5000 milisaniyelik veriyi yükler. Varsayılan `SoundMixer.bufferTime` değeri 1000'dir.

Uygulamanız, sesi yüklerken yeni bir `bufferTime` değerini açıkça belirterek tek bir ses için genel `SoundMixer.bufferTime` değerini geçersiz kılabilir. Varsayılan arabellek süresini geçersiz kılmak için, aşağıda gösterildiği gibi, ilk önce SoundLoaderContext sınıfının yeni bir örneğini oluşturun, bunun `bufferTime` özelliğini ayarlayın ve sonra bu özelliği `Sound.load()` yöntemine parametre olarak iletin:

```
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();
```

Oynatma devam ettikçe, Flash Player ve AIR uygulaması ses arabelleğini aynı boyutta veya daha yüksek bir boyutta tutmaya çalışır. Ses verisi, oynatma hızından daha çabuk yüklenirse, oynatma kesinti olmadan devam eder. Ancak ağ sınırlamaları nedeniyle veri yükleme hızı düşerse, oynatma kafası ses arabelleğinin sonuna ulaşabilir. Bu durumda oynatma otomatik askıya alınır ancak ses verisi yüklendikten sonra otomatik olarak yeniden başlatılır.

Flash Player veya AIR uygulamasının veri yüklenmesini beklediğinden askıya alınıp alınmadığını öğrenmek için, `Sound.isBuffering` özelliğini kullanın.

Dinamik olarak oluşturulmuş sesle çalışma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Not: *Flash Player 10 ve Adobe AIR 1.5'ten itibaren dinamik olarak ses oluşturma yeteneği kullanılabilir.*

Varolan bir sesi yüklemek veya akışa almak yerine, dinamik olarak ses verisi oluşturabilirsiniz. Sound nesnesinin `sampleData` olayı için bir olay dinleyicisi atadığınızda ses verisi oluşturabilirsiniz. (`sampleData` olayı, flash.events paketindeki SampleDataEvent sınıfında tanımlanır.) Bu ortamda, Sound nesnesi bir dosyadan ses verilerini yüklemez. Onun yerine, bu olaya atadığınız işlevin kullanımı aracılığıyla kendisine akışa alınan ses verileri için bir soket görevi görür.

Bir Sound nesnesine `sampleData` olay dinleyicisi eklediğinizde, nesne düzenli aralıklarla ses arabelleğine eklemek için veriler ister. Bu arabellek, oynatılacak Sound nesnesinin verilerini içerir. Sound nesnesinin `play()` yöntemini çağırdığınızda, bu yöntem yeni ses verileri istenmesi durumunda `sampleData` olayını gönderir. (Bu yalnızca Sound nesnesi bir dosyadan mp3 verilerini yüklemediğinde gerçekleşir.)

SampleDataEvent nesnesi bir `data` özelliğine sahiptir. Olay dinleyicinizde, bu `data` nesnesine ByteArray nesnelerini yazarsınız. Bu nesneye yazdığınız bayt dizileri, Sound nesnesinin oynattığı ses verilerine eklenir. Arabellekteki bayt dizisi, -1 ile 1 arasında kayan nokta değerlerinin bir akışıdır. Her kayan nokta değeri, bir ses örneğinin tek bir kanalının genliğini (sol veya sağ) temsil eder. Ses, saniyede 44.100 örnek hızında örneklenir. Her örnek, bayt dizisinde kayan nokta verisi olarak araya eklenmiş bir sol ve sağ kanal içerir.

İşleyici işlevinizde, `sampleData` olayının `data` özelliğine yazmak için `ByteArray.writeFloat()` yöntemini kullanırsınız. Örneğin, şu kod bir sinüs dalgası oluşturur:

```
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);
    }
}
```

`Sound.play()` öğesini çağırdığınızda, uygulama olay dinleyicinizi çağırmaya başlayarak ses örneği verilerini ister. Siz veri sağlamayı durduruncaya kadar veya `SoundChannel.stop()` öğesini çağırıncaya kadar ses çalındığı sürece uygulama olay göndermeye devam eder.

Olayların gecikmesi platformdan platforma değişiklik gösterir ve Flash Player ve AIR uygulamasının gelecekteki sürümlerinde değişebilir. Belirli bir gecikmeye güvenmeyin; mutlaka gecikmeyi hesaplayın. Gecikmeyi hesaplamak için şu formülü kullanın:

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

SampleDataEvent nesnesinin `data` özelliğine 2048 ile 8192 arasında örnek sağlayın (olay dinleyicisine yapılan her çağrı için). En iyi performans için mümkün olduğu kadar çok örnek sağlayın (en fazla 8192). Ne kadar az örnek sağlarsanız, oynatma sırasında o kadar çok tıklatma öğesi ve açılır pencere ortaya çıkar. Bu davranış çeşitli platformlarda değişiklik gösterebilir ve çeşitli durumlarda ortaya çıkabilir; örneğin, tarayıcıyı yeniden boyutlandırırken. Yalnızca 2048 örneği sağladığınızda tek bir platformda çalışan kod, farklı bir platformda çalıştırıldığında çalışmayabilir. Olası en düşük gecikme gerekiyorsa, veri miktarını kullanıcı tarafından seçilebilir duruma getirin.

2048'den daha az örnek (`sampleData` olay dinleyicisine yapılan her çağrı için) sağlarsanız, kalan örnekler çalıştırıldıktan sonra uygulama durdurulur. Ardından SoundChannel nesnesi bir SoundComplete olayı gönderir.

mp3 verilerinden sesi değiştirme

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Sound nesnesinden verileri ayıklamak için `Sound.extract()` yöntemini kullanırsınız. Oynatma için başka bir Sound nesnesinin dinamik akışına yazmak üzere bu verileri kullanabilirsiniz (ve değiştirebilirsiniz). Örneğin, aşağıdaki kod, yüklenen MP3 dosyasının baytlarını kullanır ve bir filtre işlevi (`upOctave()`) üzerinden bunları iletir:

```
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;
}
```

Oluşturulan seslerdeki sınırlamalar

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`sampleData` olay dinleyicisini bir Sound nesnesiyle kullandığınız zaman etkinleştirilmiş olan diğer Sound yöntemleri yalnızca şunlardır: `Sound.extract()` ve `Sound.play()`. Başka bir yöntemin veya özelliğin çağrılması bir istisnayla sonuçlanır. SoundChannel nesnesinin tüm yöntemleri ve özellikleri etkin kalmaya devam eder.

Sesleri çalma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yüklenmiş bir sesin çalınması, aşağıdaki gibi, Sound nesnesi için `Sound.play()` yönteminin çağrılması kadar basit olabilir:

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

ActionScript 3.0'ı kullanarak sesleri çaldığınızda şu işlemleri gerçekleştirebilirsiniz:

- Belirli bir başlangıç konumundan ses çalma

- Sesi duraklatıp daha sonra aynı konumdan çalmaya devam etme
- Ses çalınmasının ne zaman biteceğini tam olarak bilme
- Ses çalınmasının ilerlemesini izleme
- Ses çalınırken ses düzeyini değiştirme veya yatay kaydırma

Çalma sırasında bu işlemleri gerçekleştirmek için SoundChannel, SoundMixer ve SoundTransform sınıflarını kullanın.

SoundChannel sınıfı, tek bir sesin çalınmasını denetler. `SoundChannel.position` özelliği, çalınan ses verisinde geçerli noktayı belirten bir oynatma kafası olarak düşünülebilir.

Bir uygulama `Sound.play()` yöntemini çağırdığında, çalmayı denetlemek için SoundChannel sınıfının yeni bir örneği oluşturulur.

Uygulamanız, belirli bir başlangıç konumunu milisaniye cinsinden `Sound.play()` yönteminin `startTime` parametresi olarak iletip sesi o başlangıç konumundan itibaren çalabilir. `Sound.play()` yönteminin `loops` parametresinde bir sayısal değer ileterek hızlı şekilde art arda sesi yinelemek için sabit bir tekrarlama sayısı da belirtebilir.

`Sound.play()` yöntemi hem `startTime` parametresiyle hem de `loops` parametresiyle çağrıldığında, aşağıdaki kodda gösterildiği gibi, her defasında aynı başlangıç noktasından art arda ses çalınır:

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

Bu örnekte, ses, ses başlangıcından bir saniye sonraki noktadan itibaren üç defa art arda çalınır.

Sesi duraklatma ve yeniden başlatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız, şarkı veya podcast'ler gibi uzun sesler çalıyorsa, kullanıcıların bu seslerin çalınmasını duraklatıp yeniden başlatmasına olanak sağlamak isteyebilirsiniz. ActionScript'te ses çalınırken duraklatılamaz; yalnızca durdurulabilir. Ancak ses, herhangi bir noktadan başlayarak çalınabilir. Sesin durdurulduğu zamanki konumunu kaydedip daha sonra o konumdan başlayarak sesi yeniden çalabilirsiniz.

Örneğin, kodunuzun şöyle bir ses dosyasını yükleyip oynattığını varsayalım:

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

Ses çalınırken `SoundChannel.position` özelliği, ses dosyasındaki geçerli olarak çalınan noktayı belirtir. Uygulamanız aşağıdaki gibi, ses çalmasını durdurmadan önce konum değerini saklayabilir:

```
var pausePosition:int = channel.position;
channel.stop();
```

Ses çalmasını yeniden başlatmak için, önceden saklanan konum değerini ileterek, sesi önceden durdurulduğu noktadan yeniden başlatın.

```
channel = snd.play(pausePosition);
```

Çalmayı izleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız, başka bir ses çalmaya başlayabilmek veya önceki çalma işlemi sırasında kullanılan bazı kaynakları temizlemek için sesin ne zaman çalmayı durduracağını bilmek isteyebilir. SoundChannel sınıfı, ses çalması bitince bir `Event.SOUND_COMPLETE` olayı gönderir. Uygulamanız, aşağıda gösterildiği gibi bu olayı dinleyip uygun eylemi gerçekleştirebilir:

```
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.");
}
```

SoundChannel sınıfı, çalma sırasında ilerleme olayları göndermez. Çalma ilerlemesiyle ilgili bildirim göndermek için, uygulamanız kendi zamanlama mekanizmasını oluşturup ses oynatma kafasının konumunu izleyebilir.

Sesin yüzde kaçının çalındığını hesaplamak için, `SoundChannel.position` özelliğinin değerini, çalınan ses verisinin uzunluğuna bölebilirsiniz:

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

Ancak bu kod, yalnızca çalma işlemi başlamadan önce ses verisi tamamen yüklenmişse, tam olarak doğru çalma yüzdelerini bildirir. `Sound.length` özelliği, ses dosyasının tamamının son boyutunu değil, geçerli olarak yüklenen ses verisinin boyutunu gösterir. Halen yüklenmekte olan akışa alınan sesin çalma ilerlemesini izlemek için uygulamanızın tam ses dosyasının son boyutunu tahmin edip hesaplamalarında bu değeri kullanması gerekir. Aşağıdaki gibi, Sound nesnesinin `bytesLoaded` ve `bytesTotal` özelliklerini kullanarak ses verisinin son uzunluğunu tahmin edebilirsiniz:

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

Aşağıdaki kod, daha büyük bir ses dosyasını yükler ve çalma ilerlemesini göstermek için zamanlama mekanizması olarak `Event.ENTER_FRAME` olayını kullanır. Geçerli konum değerinin toplam ses verisi uzunluğuna bölünmesiyle elde edilen çalma yüzdesiyle ilgili olarak düzenli aralıklarla bildirim gönderir:

```
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);
}
```

Ses verisi yüklenmeye başladıktan sonra, bu kod `snd.play()` yöntemini çağırır ve sonuçta elde edilen SoundChannel nesnesini `channel` değişkeninde saklar. Bu daha sonra `Event.ENTER_FRAME` olayı için ana uygulamaya bir olay dinleyicisi eklerken, çalma tamamlandığında gerçekleşen `Event.SOUND_COMPLETE` olayı için de SoundChannel nesnesine başka bir olay dinleyicisi ekler.

Uygulama, animasyonunda yeni bir kareye her ulaştığında, `onEnterFrame()` yöntemi çağrılır. `onEnterFrame()` yöntemi, yüklenmiş olan veri miktarını esas alarak ses dosyasının toplam uzunluğunu tahmin eder, daha sonra da geçerli oynatma yüzdesini hesaplayıp görüntüler.

Sesin tamamı çalındığında, `onPlaybackComplete()` yöntemi çalıştırılarak çalma bittikten sonra ilerleme güncellemelerini görüntülememesi için `Event.ENTER_FRAME` olayının olay dinleyicisi kaldırılır.

`Event.ENTER_FRAME` olayı, saniyede birkaç defa gönderilebilir. Bazı durumlarda, çalma ilerlemesini bu kadar sıkça görüntülemek istemezsiniz. Bu durumlarda, uygulamanız flash.util.Timer sınıfını kullanarak kendi zamanlama mekanizmasını ayarlayabilir; bkz. "Tarih ve saatlerle çalışma" sayfa 1.

Akışa alınan sesleri durdurma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Akışa alınan—başka bir deyişle, çalınırken halen yüklenmekte olan seslerin çalınması işleminde garip bir durum vardır. Uygulamanız akışa alınan bir sesi çalan SoundChannel örneğinde `SoundChannel.stop()` yöntemini çağırır, ses çalması bir kare boyunca durdurulur, sonraki karede sesin başından itibaren yeniden çalma işlemi başlatılır. Ses yükleme işlemi devam ettiği için bu durum oluşur. Akışa alınan sesin hem yüklemesini hem de çalınmasını durdurmak için, `Sound.close()` yöntemini çağırın.

Sesler yüklenirken ve çalınırken dikkate alınması gereken güvenlikle ilgili noktalar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanızın ses verilerine erişme yeteneği, Flash Player veya AIR güvenlik modeline göre sınırlandırılabilir. Her ses, iki farklı güvenlik sanal alanının kısıtlamalarına tabidir: içeriğin kendi sanal alanı ("içerik sanal alanı") ve sesi yükleyip çalan uygulama veya nesnenin sanal alanı ("sahip sanal alanı"). Uygulama güvenlik sanal alanında AIR uygulama içeriği için, diğer etki alanlarından yüklenenler de dahil tüm seslere, uygulama güvenlik sanal alanındaki içerik tarafından erişilebilir. Ancak diğer güvenlik sanal alanındaki içerikler, Flash Player'da çalışan içerikle aynı kurallara uyar. Flash Player güvenlik modeli hakkında genel bilgi almak ve sanal alanların tanımlarını öğrenmek için bkz. “Güvenlik” sayfa 1053.

İçerik sanal alanı, `id3` özelliği veya `SoundMixer.computeSpectrum()` yöntemi kullanılarak sesten ses verilerinin ayıklanıp ayıklanamayacağını denetler. Bu, ses dosyasının yüklemesini veya çalınmasını kısıtlamaz.

Ses dosyasının kaynak etki alanı, içerik sanal alanının güvenlik sınırlamalarını tanımlar. Genellikle bir ses dosyası, o dosyayı yükleyen uygulama veya nesnenin SWF dosyasıyla aynı etki alanında ya da klasörde bulunuyorsa, uygulama veya nesne o ses dosyasına tam erişim elde eder. Ses, uygulamanınkinden farklı bir etki alanından geliyorsa, bir ilke dosyası kullanılarak da içerik sanal alanına getirilebilir.

Uygulamanız, `checkPolicyFile` özelliğiyle bir SoundLoaderContext nesnesini parametre olarak `Sound.load()` yöntemine iletebilir. `checkPolicyFile` özelliğinin `true` değerine ayarlanması, Flash Player veya AIR uygulamasına, sesin yüklendiği sunucuda bir ilke dosyası aramasını bildirir. İlke dosyası varsa ve yükleyen SWF dosyasının etki alanına erişim izni veriyorsa, SWF dosyası ses dosyasını yükleyebilir, Sound nesnesinin `id3` özelliğine erişebilir ve yüklenen sesler için `SoundMixer.computeSpectrum()` yöntemini çağırabilir.

Sahip sanal alan, seslerin yerel çalınmasını denetler. Sesin çalınmasını başlatan uygulama veya nesne, sahip sanal alanını tanımlar.

`SoundMixer.stopAll()` yöntemi, şu kriterleri karşıladıkları sürece, tüm SoundChannel nesnelerinde geçerli olarak çalınan seslerin sesini kapatır:

- Sesler, aynı sahip sanal alanındaki nesneler tarafından başlatılmıştır.
- Sesler, `SoundMixer.stopAll()` yöntemini çağıran uygulamanın veya nesnenin etki alanına erişim izni veren bir ilke dosyasının bulunduğu bir kaynaktan gelir.

Ancak AIR uygulamasında, uygulama güvenlik sanal alanındaki içerik (AIR uygulamasıyla yüklenmiş içerik), bu güvenlik sınırlamalarıyla kısıtlandırılmaz.

`SoundMixer.stopAll()` yönteminin tüm seslerin çalınmasını durdurup durdurmayacağını öğrenmek için, uygulamanız `SoundMixer.areSoundsInaccessible()` yöntemini çağırabilir. Bu yöntem `true` değerini döndürürse, çalınan seslerin bazıları, geçerli sahip sanal alanının denetimi dışındadır ve `SoundMixer.stopAll()` yöntemi tarafından durdurulmaz.

`SoundMixer.stopAll()` yöntemi ayrıca harici dosyalardan yüklenmiş tüm sesler için oynatma kafasının devam etmesini durdurur. Ancak, Flash Geliştirme aracı kullanılarak FLA dosyalarına gömülen ve zaman çizelgesindeki karelere eklenen sesler, animasyon yeni bir kareye geçtiğinde tekrar çalınabilir.

Ses düzeyini denetleme ve yatay kaydırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tek bir SoundChannel nesnesi bir sesin hem sol hem de sağ stereo kanallarını denetler. mp3 sesi tek taraftan gelen bir ses ise, SoundChannel nesnesinin sol ve sağ stereo kanalları aynı dalga formlarını içerir.

SoundChannel nesnesinin `leftPeak` ve `rightPeak` özelliklerini kullanarak, çalınan sesin her stereo kanalının genliğini öğrenebilirsiniz. Bu özellikler, ses dalga formunun zirve genliğini gösterir. Geçerli oynatma ses düzeyini temsil etmezler. Gerçek oynatma ses düzeyi, ses dalgası genliğinin ve SoundChannel nesnesinde ve SoundMixer sınıfında ayarlanan ses düzeyi değerlerinin bir işlevidir.

SoundChannel nesnesinin yatay kaydırma özelliği, çalma sırasında sol ve sağ kanalların her biri için farklı bir ses düzeyi belirtmek üzere kullanılabilir. Yatay kaydırma özelliği, -1 ile 1 arasında bir değere sahip olabilir, burada -1, sol kanalın en yüksek ses düzeyinde çalma işlemi yaparken sağ kanalın sesinin kapalı olduğunu, 1 de sağ kanalın en yüksek ses düzeyinde çalma işlemi yaparken sol kanalın sesinin kapalı olduğunu belirtir. -1 ile 1 arasındaki sayısal değerler, sol ve sağ kanal değerleri için orantısal değerler ayarlar ve 0 değeri de her iki kanalın dengeli, orta ses düzeyinde çalma işlemi yaptığını belirtir.

Aşağıdaki kod örneği, 0,6 ses düzeyi değerine ve -1 yatay kaydırma değerine sahip bir SoundTransform nesnesi oluşturur (sol kanalda en yüksek ses düzeyi ve sağ kanal sessiz). Bu, SoundTransform nesnesini parametre olarak `play()` yöntemine iletir ve böylece çalmayı denetlemek için oluşturulan yeni SoundChannel nesnesine bu SoundTransform nesnesi uygulanır.

```
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);
```

Bir SoundTransform nesnesinin `pan` veya `volume` özelliklerini ayarlayıp daha sonra bu nesneyi SoundChannel nesnesinin `soundTransform` özelliği olarak uygulayarak, çalma işlemi sırasında sesin ses düzeyini ve yatay kaydırmasını değiştirebilirsiniz.

Ayrıca aşağıdaki örnekte gösterildiği gibi, SoundMixer sınıfının `soundTransform` özelliğini kullanarak tek seferde tüm seslerin genel ses düzeyini ve yatay kaydırma değerlerini de ayarlayabilirsiniz:

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

Ayrıca Microphone nesnesi (bkz. “Ses girdisini yakalama” sayfa 465), Sprite nesneleri ve SimpleButton nesneleri için ses düzeyi ve yatay kaydırma değerlerini ayarlamak üzere bir SoundTransform nesnesini de kullanabilirsiniz.

Aşağıdaki örnek, ses çalarken sesin yatay kaydırmasını sol kanaldan sağ kanala geçirir ve sonra eski haline getirir.

```
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);
}
```

Bir ses dosyası yüklenip daha sonra ses düzeyi 1 (tam ses) ve yatay kaydırma 0 (sol ve sağ arasında eşit denge) değerine ayarlanmış şekilde yeni bir SoundTransform nesnesinin oluşturulmasıyla bu kod başlatılır. Daha sonra `snd.play()` yöntemini çağırarak SoundTransform nesnesini parametre olarak iletir.

Ses çalınırken, `onEnterFrame()` yöntemi art arda çalıştırılır. `onEnterFrame()` yöntemi, `SoundTransform.pan` özelliğinin kabul edilebilir değerlerine karşılık gelen bir aralık olan -1 ile 1 arasında bir değer oluşturmak için `Math.sin()` işlevini kullanır. SoundTransform nesnesinin `pan` özelliği yeni değere ayarlanır ve sonra değiştirilen SoundTransform nesnesi kullanılmak üzere kanalın `soundTransform` özelliği ayarlanır.

Bu örneği çalıştırmak için, bigSound.mp3 dosya adını yerel bir mp3 dosyasının adıyla değiştirin. Daha sonra örneği çalıştırın. Sağ kanalın ses düzeyi daha yumuşak duruma gelirken sol kanalın ses düzeyinin daha çok yükseldiğini, veya tam tersini fark etmeniz gerekir.

Bu örnekte, SoundMixer sınıfının `soundTransform` özelliği ayarlanarak aynı etki elde edilebilir. Ancak bu yalnızca bu SoundChannel nesnesi tarafından oynatılan tek bir sesi değil, geçerli olarak çalınan tüm seslerin yatay kaydırmasını etkiler.

Ses meta verisiyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

mp3 biçimini kullanan ses dosyaları, sesler hakkında ID3 etiketleri şeklinde ek veriler içerebilir.

Her mp3 dosyası ID3 meta verisini içermez. Bir Sound nesnesi mp3 ses dosyasını yüklediğinde, ses dosyasında ID3 meta verisi bulunsa da bir `Event.ID3` olayını gönderir. Çalışma zamanı hatalarını önlemek için, uygulamanızın yüklenmiş bir sesin `Sound.id3` özelliğine erişmeden önce `Event.ID3` olayını almayı beklemesi gerekir.

Aşağıdaki kod, bir ses dosyası için ID3 meta verisi yüklendiğinde bunun nasıl algılandığını gösterir:

```
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]);
    }
}
```

Bu kod, bir Sound nesnesinin oluşturulup bu nesneye `Event.ID3` olayını dinlemesi bildirilerek başlatılır. Ses dosyasının ID3 meta verileri yüklendiğinde, `onID3InfoReceived()` yöntemi çağrılır. `onID3InfoReceived()` yöntemine iletilen Event nesnesinin hedefi, orijinal Sound nesnesidir, bu nedenle yöntem daha sonra Sound nesnesinin `id3` özelliğini alır ve değerlerini izlemek için adlandırılmış özelliklerinin tümünü yineler.

Ham ses verilerine erişme

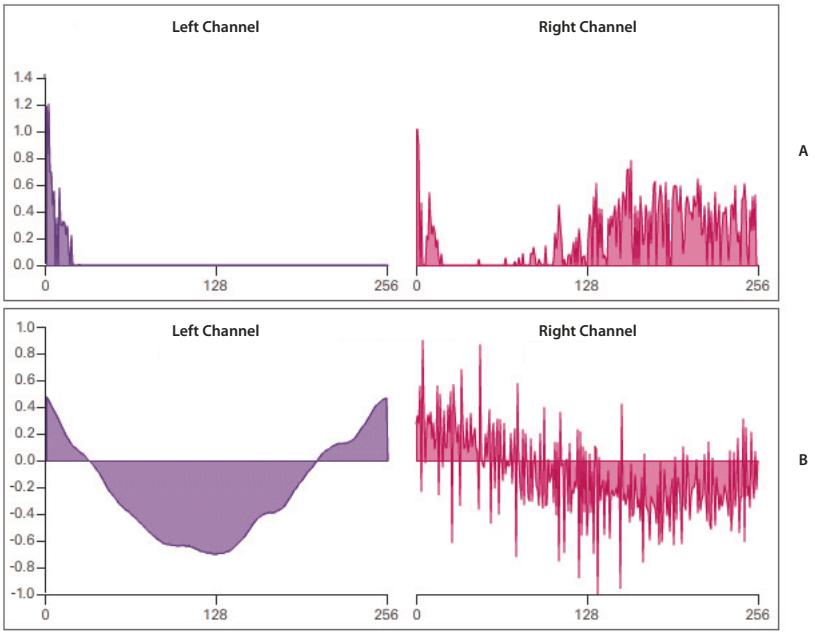
Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`SoundMixer.computeSpectrum()` yöntemi, bir uygulamanın geçerli olarak oynatılan dalga formu için ham ses verilerini okumasına olanak sağlar. Geçerli olarak birden çok SoundChannel nesnesi oynatılıyorsa, `SoundMixer.computeSpectrum()` yöntemi, birbirine karıştırılan her SoundChannel nesnesinin birleştirilmiş ses verisini gösterir.

Ses verileri, her biri -1 ile 1 arasında kayan nokta değeri içeren, 512 bayt verinin bulunduğu bir ByteArray nesnesi olarak döndürülür. Bu değerler, oynatılan ses dalga formunda noktaların genliğini temsil eder. Değerler 256'lık iki grup halinde teslim edilir, birinci grup sol stereo kanalı için ve ikinci grup da sağ stereo kanalı içindir.

`FFTMode` parametresi `true` değerine ayarlandığında, `SoundMixer.computeSpectrum()` yöntemi, dalga formu verisini değil, frekans spektrumu verisini döndürür. Frekans spektrumu, en düşük frekanstan en yüksek frekansa doğru, ses frekansı tarafından düzenlenen genliği gösterir. Dalga formunu frekans spektrumu verilerine dönüştürmek için Fast Fourier Transform (FFT) kullanılır. Sonuçta elde edilen frekans spektrumu değerleri, 0 ile kabaca 1,414 (2'nin kare kökü) arasındadır.

Aşağıdaki diyagram, `FFTMode` parametresi `true` olarak ve `false` olarak ayarlandığında `computeSpectrum()` yönteminden döndürülen verileri karşılaştırır. Bu diyagram için verileri kullanılan ses, sol kanalda yüksek bir bas sesi ve sağ kanalında bir davul vuruşu sesi içerir.



SoundMixer.computeSpectrum() yöntemi tarafından döndürülen değerler
A. *fftMode=true* ***B.*** *fftMode=false*

`computeSpectrum()` yöntemi ayrıca daha düşük bir bit hızında yeniden örneklenmiş verileri de döndürebilir. Genellikle bu, ayrıntı kaybına yol açacak şekilde daha yumuşak bir dalga formu verisiyle veya frekans verisiyle sonuçlanır. `stretchFactor` parametresi, `computeSpectrum()` yöntemi verilerinin örneklendiği hızı denetler. `stretchFactor` parametresi varsayılan 0 değerine ayarlandığında, ses verileri 44,1 kHz hızında örneklenir. stretchFactor parametresinin art arda her değerinde hız yarıya iner, bu nedenle, 1 değeri 22,05 kHz hızı belirtir, 2 değeri 11,025 kHz hızı belirtir ve bu şekilde devam eder. Daha yüksek bir `stretchFactor` değeri kullanıldığında, `computeSpectrum()` yöntemi tek bir stereo kanalı için 256 bayt döndürmeye devam eder.

`SoundMixer.computeSpectrum()` yöntemi bazı sınırlamalara sahiptir:

- Bir mikrofondan veya RTMP akışından gelen ses verileri genel bir SoundMixer nesnesinden geçmediğinden, `SoundMixer.computeSpectrum()` yöntemi bu kaynaklardan veri döndürmez.
- Çalınan seslerden biri veya birkaçı, geçerli içerik sanal alanı dışındaki kaynaklardan geliyorsa, güvenlik kısıtlamaları, `SoundMixer.computeSpectrum()` yönteminin bir hata atmasına neden olur. `SoundMixer.computeSpectrum()` yönteminin güvenlik sınırlamaları hakkında daha fazla ayrıntılı bilgi almak için, bkz. "Sesler yüklenirken ve çalınırken dikkate alınması gereken güvenlikle ilgili noktalar" sayfa 458 ve "Veri olarak yüklenen ortama erişme" sayfa 1077.

Ancak AIR uygulamasında, uygulama güvenlik sanal alanındaki içerik (AIR uygulamasıyla yüklenmiş içerik), bu güvenlik sınırlamalarıyla kısıtlandırılmaz.

Basit ses görselleştiricisi oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

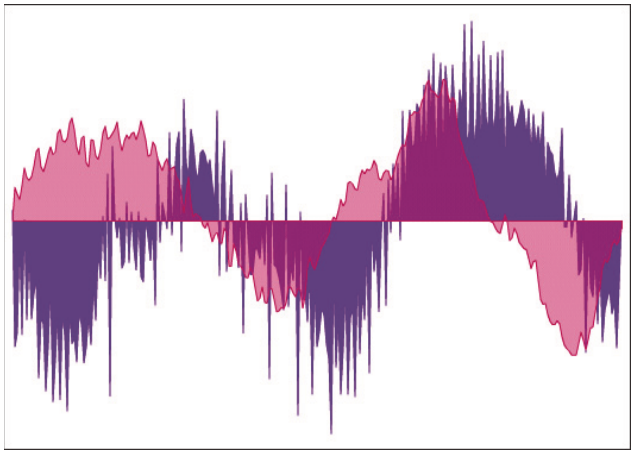
Aşağıdaki örnek, her kareyle animasyon uygulanan bir ses dalga formu grafiğini göstermek için `SoundMixer.computeSpectrum()` yöntemini kullanır:

```
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);
}
```

Bu örnek ilk önce bir ses dosyasını yükler ve çalar daha sonra, ses çalınırken `onEnterFrame()` yöntemini tetikleyen `Event.ENTER_FRAME` olayını dinler. `onEnterFrame()` yöntemi, `bytes` ByteArray nesnesinde ses dalgası verilerini saklayan `SoundMixer.computeSpectrum()` yönteminin çağrılmasıyla başlatılır.

Ses dalga formu, vektör çizim API'si kullanılarak çizilir. `for` döngüsü, sol stereo kanalını temsil eden ilk 256 veri değeri üzerinde döngü gerçekleştirir ve `Graphics.lineTo()` yöntemini kullanarak her noktadan bir sonraki noktaya bir çizgi çizer. İkinci bir `for` döngüsü de bir sonraki 256 değer kümesinde döngü gerçekleştirerek bu defa tam tersi yönde, sağdan sola çizim yapar. Sonuçta elde edilen dalga formu çizimleri, aşağıdaki görüntüde görüldüğü gibi, ilginç bir aynı görüntüsü efekti oluşturabilir.



Ses girdisini yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Microphone sınıfı, uygulamanızın bir mikrofona veya kullanıcının sistemindeki başka bir ses girdisi aygıtına bağlanmasına ve girdi sesini o sistemin hoparlörlerine yayınlamasına ya da Flash Media Server gibi uzak bir sunucuya ses verileri göndermesine olanak sağlar. Mikrofondan işlenmemiş ses verisine erişebilir veya bunu kaydedebilir ya da işleyebilirsiniz. Ayrıca sesi doğrudan sistemin hoparlörlerine gönderebilir veya sıkıştırılmış ses verisini uzak bir sunucuya gönderebilirsiniz. Uzak sunucuya gönderilmiş veriler için Speex veya Nellymoser codec bileşeni kullanabilirsiniz. (Speex codec bileşeni, Flash Player 10 ve Adobe AIR 1.5'ten itibaren desteklenmektedir.)

Daha fazla Yardım konusu

Michael Chaize: AIR, Android, and the Microphone

Christophe Coenraets: Voice Notes for Android

Mikrofona erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Microphone sınıfı bir yapıcı yöntemine sahip değildir. Bunun yerine, aşağıda gösterildiği gibi, yeni bir Microphone örneği almak için statik `Microphone.getMicrophone()` yöntemini kullanırsınız:

```
var mic:Microphone = Microphone.getMicrophone();
```

`Microphone.getMicrophone()` yönteminin bir parametre olmadan çağrılması, kullanıcının sisteminde keşfedilen birinci ses girdisi aygıtını döndürür.

Bir sistemde, bağlı birden çok ses girdisi aygıtı bulunabilir. Uygulamanız, tüm kullanılabilir ses girdisi aygıtlarının adlarının dizisini almak için `Microphone.names` özelliğini kullanabilir. Bu daha sonra dizide aygıtın adının dizin değeriyle eşleşen bir `index` parametresiyle `Microphone.getMicrophone()` yöntemini çağırabilir.

Sistemde bağlı bir mikrofon veya başka bir ses girdisi aygıtı bulunmayabilir. Kullanıcının bir ses girdisi aygıtı takmış olup olmadığını kontrol etmek için `Microphone.names` özelliğini veya `Microphone.getMicrophone()` yöntemini kullanabilirsiniz. Kullanıcı bir ses girdisi aygıtı takmadıysa, `names` dizisi sıfır uzunluğuna sahip olur ve `getMicrophone()` yöntemi `null` değerini döndürür.

Uygulamanız`Microphone.getMicrophone()` yöntemini çağırdığında, Flash Player uygulaması, Flash Player'ın sistemdeki kameraya veya mikrofona erişmesine izin verip vermeyeceğini kullanıcıya soran Flash Player Ayarları iletişim kutusunu görüntüler. Kullanıcı bu iletişim kutusunda İzin Ver veya Reddet düğmesini tıklattıktan sonra bir StatusEvent gönderilir. Bu StatusEvent örneğinin `code` özelliği, bu örnekte gösterildiği gibi, mikrofon erişimine izin verilip verilmediğini belirtir:

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

`StatusEvent.code` özelliği, erişime izin verildiyse “Microphone.Unmuted” öğesini içerirken, erişim reddedildiyse “Microphone.Muted” öğesini içerir.

Kullanıcı mikrofon erişimine izin verdiğinde veya mikrofon erişimini reddettiğinde `Microphone.muted` özelliği sırayla `true` veya `false` değerine ayarlanır. Ancak StatusEvent öğesi gönderilene kadar `muted` özelliği ayarlanmaz. Bu nedenle uygulamanızın `Microphone.muted` özelliğini kontrol etmeden önce `StatusEvent.STATUS` olayının gönderilmesini beklemesi gerekir.

Flash Player'ın ayarlar iletişim kutusunu görüntülemesi için uygulama penceresi bunu görüntüleyecek kadar büyük olmalıdır (en azından 215 x 138 piksel) Aksi takdirde, erişim otomatik olarak reddedilir.

AIR uygulaması sanal alanında çalışan içeriğin mikrofona erişmek için kullanıcı iznine ihtiyacı yoktur. Bu nedenle, mikrofonun sesini kapatmak ve açmak için durum olayları gönderilmez. AIR'de uygulama sanal alanının dışında çalışan içerik kullanıcıdan izin gerektirir. Bu nedenle bu durum olayları gönderilebilir.

Mikrofon sesini yerel hoparlörlere yönlendirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir mikrofondan gelen ses girdisi, `true` parametre değeriyle `Microphone.setLoopback()` yöntemi çağrılarak yerel sistem hoparlörlerine yönlendirilebilir.

Yerel mikrofondan gelen ses yerel hoparlörlere yönlendirildiğinde, ses geribildirimi döngüsü oluşma riski vardır ve bu da cızırtılı seslere neden olarak ses donanımına zarar verebilir. `true` parametre değeriyle `Microphone.setUseEchoSuppression()` yönteminin çağrılması, ses geribildirimi oluşma riskini azaltır ancak tamamen ortadan kaldırmaz. Adobe, kullanıcının kulaklık kullanarak veya hoparlör dışında bir şeyle ses çaldığından emin değilseniz, `Microphone.setLoopback(true)` yöntemini çağırmadan önce her zaman `Microphone.setUseEchoSuppression(true)` yöntemini çağırmanızı önerir.

Aşağıdaki kod, yerel mikrofondan gelen sesin yerel sistem hoparlörlerine nasıl yönlendirildiğini gösterir:

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

Mikrofon sesini değiştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız bir mikrofondan gelen ses verilerini iki şekilde değiştirebilir. İlk olarak, girdi sesinin kazanımını değiştirebilir böylece daha yüksek veya daha alçak bir ses oluşturmak için girdi değerleri belirtilen bir miktarla etkili şekilde çarpılır. `Microphone.gain` özelliği 0 ile 100 arasındaki sayısal değerleri kabul eder. 50 değeri, bir çarpanı gibi hareket eder ve normal ses düzeyini belirtir. Sıfır değeri, sıfır çarpanı gibi hareket eder ve girdi sesini etkili şekilde kapatır. 50'nin üzerindeki değerler, normal ses düzeyinin üzerini belirtir.

Uygulamanız da girdi sesinin örnek hızını değiştirebilir. Yüksek örnek hızları ses kalitesini artırır ancak iletim ve depolama için daha fazla kaynak kullanan daha yoğun veri akışları da oluşturur. `Microphone.rate` özelliği, kilohertz (kHz) cinsinden ölçülen ses örnek hızını temsil eder. Varsayılan örnek hızı 8 kHz'dir. Mikrofonunuz daha yüksek bir değeri destekliyorsa, `Microphone.rate` özelliğini 8 kHz'den yüksek bir değere ayarlayabilirsiniz. Örneğin, `Microphone.rate` özelliğinin 11 değerine ayarlanması, örnek hızını 11 kHz olarak ayarlar; 22 olarak ayarlanması, örnek hızını 22 kHz olarak ayarlar ve bu şekilde gider. Mevcut örnek hızları, seçilen codec bileşenine bağlıdır. Nellymoser codec bileşenini kullandığınızda, örnek hızı olarak 5, 8, 11, 16, 22 ve 44 kHz değerlerini belirtebilirsiniz. Speex codec bileşenini kullanırken (Flash Player 10 ve Adobe AIR 1.5'ten itibaren kullanılabilir), yalnızca 16 kHz kullanabilirsiniz.

Mikrofon etkinliğini algılama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player, bant genişliğini ve işleme kaynaklarını korumak için mikrofon tarafından herhangi bir ses iletilmediğinde bunu algılamaya çalışır. Mikrofonun etkinlik düzeyi, belirli bir süre boyunca sessizlik düzeyi eşiğinin altında kaldığında, Flash Player ses girdisini iletmeyi durdurur ve bunun yerine basit bir ActivityEvent öğesini gönderir. Speex codec bileşenini kullanırsanız (Flash Player 10 veya sonrasında ve Adobe AIR 1.5 veya sonrasında kullanılabilir), uygulamanın ses verilerini sürekli iletmesini sağlamak için sessizlik düzeyini 0'a ayarlayın. Speex ses etkinliği algılama özelliği otomatik olarak bant genişliğini azaltır.

Not: *Uygulamanız mikrofonu izlerken bir Microphone nesnesi yalnızca Activity olayları gönderir. Bu nedenle, `setLoopBack( true )` öğesini çağırmazsanız örnek veri olayları için bir dinleyici ekleyin veya mikrofonu bir NetStream nesnesine bağlayın. Böylece etkinlik olayları gönderilmez.*

Microphone sınıfının üç özelliği, etkinliğin algılanmasını izler ve denetler:

- Salt okunur `activityLevel` özelliği, mikrofonun algıladığı ses miktarını 0 ile 100 arasında bir ölçekte belirtir.
- `silenceLevel` özelliği, mikrofonu etkinleştirmek ve bir `ActivityEvent.ACTIVITY` olayı göndermek için gereken ses miktarını belirtir. `silenceLevel` özelliği ayrıca 0 ile 100 arasında bir ölçek kullanır ve varsayılan değer 10'dur.
- `silenceTimeout` özelliği, mikrofonun sessiz olduğunu belirtmek üzere bir `ActivityEvent.ACTIVITY` olayı gönderilinceye kadar etkinlik düzeyinin sessizlik düzeyinin altında kalması gerektiği milisaniye sayısını açıklar. Varsayılan `silenceTimeout` değeri 2000'dir.

Hem `Microphone.silenceLevel` özelliği hem de `Microphone.silenceTimeout` özelliği salt okunurdur, ancak `Microphone.setSilenceLevel()` yöntemi kullanılarak değerleri değiştirilebilir.

Bazı durumlarda, yeni etkinlik algılandığında mikrofonu etkinleştirme işlemi kısa bir gecikmeye neden olabilir. Mikrofonun her zaman etkin tutulması bu etkinleştirme gecikmelerini ortadan kaldırabilir. Uygulamanız, Flash Player uygulamasına ses algılanmasa da mikrofonu etkin tutmasını ve ses verisi toplamaya devam etmesini bildirmek için `silenceLevel` parametresi sıfıra ayarlı şekilde `Microphone.setSilenceLevel()` yöntemini çağırabilir. Tam tersine, `silenceLevel` parametresinin 100 olarak ayarlanması, mikrofonun tamamen etkinleştirilmesini önler.

Aşağıdaki örnek, mikrofonla ilgili bilgileri görüntüler ve Microphone nesnesi tarafından gönderilen etkinlik olayları ve durum olaylarıyla ilgili bildirimler gönderir:

```
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);
}
```

Yukarıdaki örneği çalıştırdığınızda, sistem mikrofonunuza konuşursanız veya ses çıkarırsanız, sonuçta ortaya çıkan izleme deyimlerinin bir konsolda veya hata ayıklama penceresinde görüntülendiğini görürsünüz.

Medya sunucusuna ve medya sunucusundan ses gönderme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Media Server gibi bir akışa alma medya sunucusuyla ActionScript kullanıldığında ek ses yetenekleri kullanılabilir.

Özellikle uygulamanız bir NetStream nesnesine Microphone nesnesini ekleyip doğrudan kullanıcının mikrofonundan sunucuya veri iletebilir. Ses verileri sunucudan bir uygulamaya akışa alınabilir ve MovieClip öğesinin parçası olarak veya bir Video nesnesi kullanılarak oynatılabilir.

Speex codec bileşeni Flash Player 10 ve Adobe AIR 1.5'ten itibaren kullanılabilir. Medya sunucusuna gönderilen sıkıştırılmış ses için kullanılan codec bileşenini ayarlamak için, `codec` özelliğini Microphone nesnesine ayarlayın. Bu özellik, SoundCodec sınıfında numaralandırılmış iki değere sahiptir. codec özelliğinin `SoundCodec.SPEEX` olarak ayarlanması, sesin sıkıştırılması için Speex codec bileşenini seçer. Özelliğin `SoundCodec.NELLYMOSER` (varsayılan) olarak ayarlanması, sesin sıkıştırılması için Nellymoser codec bileşenini seçer.

Daha fazla bilgi için www.adobe.com/go/learn_fms_docs_tr adresinde çevrimiçi Flash Media Server belgelerine bakın.

Mikrofon ses verilerini yakalama

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Flash Player 10.1 ve AIR 2'de ya da üstünde, mikrofondan kayan nokta değerlerinin bayt dizisi olarak veri yakalayabilirsiniz. Her bir değer, mikrofon ses verilerinin bir örneğini temsil eder.

Mikrofon verilerini almak için, Microphone nesnesinin `sampleData` olayı için bir olay dinleyicisi ayarlayın. Mikrofon arabelleği ses örnekleriyle doldukça, Microphone nesnesi düzenli aralıklarla `sampleData` olayları gönderir. SampleDataEvent nesnesinin, ses örneklerinin bit dizisi olan`data` özelliği vardır. Örneklerin her biri, tek sesli ses örneklerini temsil eden kayan nokta değerleri olarak temsil edilir.

Aşağıdaki kod, mikrofon ses verilerini `soundBytes` adlı bir ByteArray nesnesinde toplar.

```
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);
    }
}
```

Örnek baytları bir Sound nesnesi için oynatma sesi olarak tekrar kullanabilirsiniz. Bunu yaparsanız, Microphone nesnesinin `rate` özelliğini 44'e ayarlamalısınız, bu Sound nesneleri tarafından kullanılan örnek hızıdır. (Sound nesnesinin gerektirdiği 44 kHz hızından daha düşük bir hızda yakalanmış mikrofon örneklerini de dönüştürebilirsiniz.) Ayrıca, Microphone nesnesi tek sesli örnekler yakalarken, Sound nesnesinin stereo sesler kullandığını unutmayın. Yani, Microphone nesnesi tarafından yakalanan her bir biti Sound nesnesine iki kere yazmalısınız. Aşağıdaki örnekte 4 saniyelik mikrofon verisi yakalanmaktadır ve bir Sound nesnesi kullanılarak çalınmaktadır:

```
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.");
}
```

Ses örneği verilerinden ses çalma hakkında daha fazla bilgi için, bkz.“Dinamik olarak oluşturulmuş sesle çalışma” sayfa 452.

Ses örneği: Podcast Oynatıcısı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Podcast, istek üzerine veya abonelik yoluyla Internet üzerinden dağıtılan bir ses dosyasıdır. Podcast'ler genellikle podcast kanalı olarak da bilinen serilerin bir parçası olarak yayınlanır. Podcast bölümleri bir dakika ile birkaç saat arasında değişebildiğinden genellikle oynatılırken akışa alınır. Öğe olarak da adlandırılan podcast bölümleri genellikle mp3 biçiminde sunulur. Video podcast'leri de çok sık kullanılır ancak bu örnek uygulama yalnızca mp3 dosyaları kullanan ses podcast'lerini oynatır.

Bu örnek tam özellikli bir podcast toplayıcı uygulama değildir. Örneğin, bu belirli podcast'lere abonelikleri yönetmez veya uygulama bir sonraki defa çalıştırıldığında kullanıcının hangi podcast'leri dinlediğini hatırlamaz. Bu daha tam özellikli bir podcast toplayıcısı için başlangıç noktası görevi görür.

Podcast Player örneği, şu ActionScript programlama tekniklerini gösterir:

- Harici bir RSS beslemesini okuma ve XML içeriğini ayrıştırma
- Ses dosyalarının yüklenmesini ve oynatılmasını basitleştirmek için SoundFacade sınıfı oluşturma
- Ses çalma ilerlemesini görüntüleme
- Ses çalmasını duraklatma ve yeniden başlatma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Podcast Player uygulama dosyalarını Samples/PodcastPlayer klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
PodcastPlayer.mxml veya PodcastPlayer.fla	Flex (MXML) veya Flash (FLA) uygulaması için kullanıcı arabirimi.
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	Podcast player için kullanıcı arabirimi mantığını içeren belge sınıfı (yalnızca Flash).
SoundPlayer.mxml	Yalnızca Flex için oynatma düğmelerini ve ilerleme çubuklarını görüntüleyen ve ses çalmasını denetleyen bir MXML bileşeni.
main.css	Uygulama kullanıcı arabirimi için stiller (yalnızca Flex).
images/	Düğme stilleri için simgeler (yalnızca Flex).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	Ses oynatıcısı için kullanıcı arabirimi mantığını içeren SoundPlayer film klibi simgesi için sınıf (yalnızca Flash).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	Veri ızgara hücresinde bir oynatma düğmesini görüntülemek için özel hücre oluşturucu (Yalnızca Flash).
com/example/programmingas3/podcastplayer/RSSBase.as	RSSChannel sınıfı ve RSSItem sınıfı için ortak özellikler ve yöntemler sağlayan bir temel sınıf.
com/example/programmingas3/podcastplayer/RSSChannel.as	RSS kanalı hakkında verileri barındıran bir ActionScript sınıfı.

File	Açıklama
com/example/programmingas3/podcastplayer/RSSItem.as	RSS öğesi hakkında verileri barındıran bir ActionScript sınıfı.
com/example/programmingas3/podcastplayer/SoundFacade.as	Uygulamanın ana ActionScript sınıfı Sound sınıfının ve SoundChannel sınıfının yöntemlerini ve olaylarını kapsüller ve oynatmanın duraklatılması ve yeniden başlatılmasına yönelik destek ekler.
com/example/programmingas3/podcastplayer/URLService.as	Uzak bir URL'den gelen verileri alan bir ActionScript sınıfı.
playerconfig.xml	Podcast kanallarını temsil eden RSS beslemelerinin listesini içeren bir XML dosyası.
comp/example/programmingas3/utils/DateUtil.as	Kolay tarih biçimlendirme için kullanılan sınıf (yalnızca Flash).

Podcast kanalı için RSS verilerini okuma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Podcast Player uygulaması, çok sayıda podcast kanalı ve bölümleri hakkında bilgiler okunarak başlatılır:

1. İlk olarak uygulama, podcast kanallarının listesini içeren bir XML konfigürasyon dosyasını okur ve kanalların listesini kullanıcıya görüntüler.

2. Kullanıcı podcast kanallarından birini seçtiğinde, kanalın RSS beslemesini okur ve kanal bölümlerinin listesini görüntüler.

Bu örnek, uzak bir konumdan veya yerel dosyadan metin tabanlı verileri almak için URLLoader yardımcı program sınıfını kullanır. Podcast Player ilk olarak playerconfig.xml dosyasından RSS beslemelerinin listesini XML biçiminde almak için bir URLLoader nesnesi oluşturur. Daha sonra kullanıcı listeden belirli bir beslemeyi seçtiğinde, o beslemenin URL'sinden RSS verilerini okumak için yeni bir URLLoader nesnesi oluşturulur.

SoundFacade sınıfını kullanarak ses yüklemesini ve çalınmasını basitleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0 ses mimarisi güçlü ancak karmaşıktır. Yalnızca temel ses yükleme ve çalma özelliklerine ihtiyaç duyan uygulamalar, daha basit bir yöntem çağrıları ve olayları kümesi sağlayarak karmaşıklığın bir kısmını gizleyen bir sınıfı kullanabilir. Yazılım tasarımı modelleri dünyasında bu sınıf *cephe* olarak adlandırılır.

SoundFacade sınıfı, şu görevlerin gerçekleştirilmesine yönelik tek bir arabirim sunar:

- Sound nesnesini, SoundLoaderContext nesnesini ve SoundMixer sınıfını kullanarak ses dosyaları yükleme
- Sound nesnesini ve SoundChannel nesnesini kullanarak ses dosyalarını çalma
- Çalma ilerlemesi olaylarını gönderme
- Sound nesnesini ve SoundChannel nesnesini kullanarak ses çalınmasını duraklatma ve yeniden başlatma

SoundFacade sınıfı, ActionScript ses sınıflarının en çok sayıda işlevini en az karmaşıklıkta sunmaya çalışır.

Aşağıdaki sınıf, sınıf bildirimini, sınıf özelliklerini ve `SoundFacade()` yapıcı yöntemini gösterir:

```
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();
        }
    }
```

SoundFacade sınıfı, kendi olaylarını gönderebilecek şekilde EventDispatcher sınıfını genişletir. Sınıf kodu ilk olarak bir Sound nesnesi ve bir SoundChannel nesnesi için özellikleri bildirir. Bu sınıf ayrıca ses dosyasının URL'sinin değerini ve ses akışa alınırken kullanılacak `bufferTime` özelliğini içerir. Üstelik yükleme ve çalma davranışını etkileyen bazı Boolean parametrelerini de kabul eder:

- `autoLoad` parametresi, ses yüklemesinin bu nesne oluşturulduğu anda başlaması gerektiğini nesneye bildirir.
- `autoPlay` parametresi, ses çalınmasının yeterli ses verisi yüklendiği anda başlaması gerektiğini belirtir. Bu akışa alınan bir ses ise, `bufferTime` özelliği tarafından belirtildiği şekilde yeterince veri yüklendiği anda çalma işlemi başlar.

- `streaming` parametresi, bu ses dosyasının yükleme tamamlanmadan önce çalınmaya başlayabildiğini belirtir.

`bufferTime` parametresinin varsayılan değeri -1'dir. Yapıcı yöntemi `bufferTime` parametresinde negatif bir değer algılarsa, `bufferTime` özelliğini `SoundMixer.bufferTime` değerine ayarlar. Bu, istendiği zaman uygulamanın genel `SoundMixer.bufferTime` değerini varsayılan olarak uygulamasına olanak sağlar.

`autoLoad` parametresi `true` değerine ayarlanırsa, ses dosyasının yüklemesini başlatmak için yapıcı yöntemi hemen şu `load()` yöntemini çağırır:

```
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);
}
```

`load()` yöntemi yeni bir Sound nesnesi oluşturur ve sonra tüm önemli ses olayları için dinleyiciler ekler. Daha sonra Sound nesnesine, `bufferTime` değerinde iletmek üzere SoundLoaderContext nesnesini kullanarak ses dosyasını yüklemesini bildirir.

`url` özelliği değiştirilebildiğinden, art arda farklı ses dosyalarını çalmak için SoundFacade örneği kullanılabilir: yalnızca `url` özelliğinin değiştirilip `load()` yönteminin çağrılması sonucunda yeni ses dosyası yüklenmiş olur.

Aşağıdaki olay dinleyicisi yöntemleri, SoundFacade nesnesinin yükleme ilerlemesini nasıl izlediğini ve ses çalmasını başlatmaya nasıl karar verdiğini gösterir:

```
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();
    }
}
```

Ses yüklemesi başladığında `onLoadOpen()` yöntemi çalıştırılır. Akışa alma modunda ses çalınabiliyorsa, `onLoadComplete()` yöntemi hemen `isReadyToPlay` bayrağını `true` değerine ayarlar. `isReadyToPlay` bayrağı, uygulamanın, belki de Oynat/Çal düğmesinin tıklatılması gibi bir kullanıcı eylemine yanıt olarak ses çalmasını başlatıp başlatamayacağını belirler. SoundChannel sınıfı, ses verilerinin arabelleğe alınmasını yönetir, bu nedenle `play()` yöntemi çağrılmadan önce yeterince veri yüklenmiş olup olmadığını açıkça kontrol etmeye gerek yoktur.

`onLoadProgress()` yöntemi, yükleme işlemi sırasında düzenli aralıklarla çalıştırılır. Bu SoundFacade nesnesini kullanan kod tarafından kullanılmak üzere ProgressEvent nesnesinin bir klonunu gönderir.

Ses verileri tamamen yüklendiğinde `onLoadComplete()` yöntemi çalıştırılarak gerektiğinde akışa alınmayan sesler için `play()` yöntemini çağırır. `play()` yöntemi aşağıda gösterilmiştir.

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

`play()` yöntemi, ses çalınmaya hazır olduğunda `Sound.play()` yöntemini çağırır. Sonuçta elde edilen SoundChannel nesnesi, `sc` özelliğinde saklanır. Daha sonra `play()` yöntemi, düzenli aralıklarla çalma ilerlemesi olaylarını göndermek için kullanılacak bir Timer nesnesini oluşturur.

Çalma ilerlemesini görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çalmayı izlemek için bir Timer nesnesinin oluşturulması, yalnızca bir defa kodlamanız gereken karmaşık bir işlemdir. Bu Timer mantığının SoundFacade sınıfı gibi yeniden kullanılabilir bir sınıfta kapsüllenmesi, bir ses yüklenirken ve çalınırken uygulamaların aynı türdeki ilerleme olaylarını dinlemesine olanak sağlar.

`SoundFacade.play()` yöntemi tarafından oluşturulan Timer nesnesi, her saniye bir TimerEvent örneği gönderir. Aşağıdaki `onPlayTimer()` yöntemi, her yeni bir TimerEvent geldiğinde çalıştırılır:

```
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);
}
```

`onPlayTimer()` yöntemi, "Çalmayı izleme" sayfa 456 bölümünde açıklanan boyut tahmini tekniğini uygular. Daha sonra `bytesLoaded` özelliği, SoundChannel nesnesinin geçerli konumuna ve `bytesTotal` özelliği de ses verisinin tahmini uzunluğuna ayarlanmış şekilde, `SoundFacade.PLAY_PROGRESS` olay türünde yeni bir ProgressEvent örneği oluşturur.

Çalmayı duraklatma ve yeniden başlatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Önceden gösterilen `SoundFacade.play()` yöntemi, ses verisindeki bir başlangıç konumuna karşılık gelen `pos` parametresini kabul eder. `pos` değeri sıfır olursa, ses baştan çalınmaya başlar.

`SoundFacade.stop()` yöntemi burada gösterildiği gibi bir `pos` parametresini de kabul eder:

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

`SoundFacade.stop()` yöntemi her çağrıldığında `pausePosition` özelliğini ayarlar, böylece kullanıcı aynı sesin çalınmasını yeniden başlatmak istediğinde uygulama da oynatma kafasının nerede konumlandırıldığını bilir.

Aşağıda gösterilen `SoundFacade.pause()` ve `SoundFacade.resume()` yöntemleri sırayla `SoundFacade.stop()` ve `SoundFacade.play()` yöntemlerini çağırarak her seferinde bir `pos` parametre değerini iletir.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

`pause()` yöntemi geçerli `SoundChannel.position` değerini `play()` yöntemine iletir ve böylece bu değer `pausePosition` özelliğinde saklanır. `resume()` yöntemi, `pausePosition` değerini başlangıç noktası olarak kullanarak aynı sesin tekrar çalınmasını sağlar.

Podcast Player örneğini genişletme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnek, yeniden kullanılabilir SoundFacade sınıfının kullanılmasını gösteren temel bir Podcast Player uygulamasını sunmaktadır. Bu uygulamanın kullanışlılığını artırmak için aşağıdaki gibi başka ek özellikler de ekleyebilirsiniz:

- Kullanıcı bir dahaki sefer uygulamayı çalıştırdığında kullanılabilen bir SharedObject örneğinde beslemelerin listesini ve her bölümle ilgili kullanım bilgilerini saklama.
- Kullanıcının podcast kanallarının listesine kendi RSS beslemelerini eklemesine olanak sağlama.
- Kullanıcı bir bölümü durdurduğunda veya bir bölümden çıktığında oynatma kafasının konumunu hatırlama ve böylece kullanıcı uygulamayı bir dahaki sefer çalıştırdığında o noktadan başlanabilmesini sağlama.
- Internet bağlantısı olmadığında çevrimdışı da dinlenmesi için bölümlerin mp3 dosyalarını indirme.
- Podcast kanalında yeni bölümleri düzenli aralıklarla kontrol edip bölüm listesini otomatik olarak güncelleyen ek abonelik özellikleri ekleme.
- API kullanılarak Odeo.com gibi bir podcast barındırma hizmetinden podcast arama ve göz atma işlevselliği ekleme.

Bölüm 25: Videoyla çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash, Internet'teki üstünlüğüyle göze çarpan teknolojilerden biridir. Ancak videonun geleneksel sunumu (alt kısmında bir ilerleme çubuğu ve denetim düğmeleri bulunan diktörgen bir ekran), videonun olası kullanım şekillerinden yalnızca biridir. ActionScript yoluyla, video yüklemesi, sunum ve oynatma gibi işlemlere erişimin ince ayarını yapar ve bu işlemler üzerinde denetim elde edersiniz.

Video temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe® Flash® Player ve Adobe® AIR™ uygulamalarının önemli bir özelliği, ActionScript'te tıpkı görüntü, animasyon, metin vb. gibi diğer görsel içerikleri işlediğiniz şekilde video bilgilerini görüntüleme ve işleme yeteneğidir. Adobe Flash CS4 Professional'da Flash Video (FLV) dosyası oluşturduğunuzda, ortak oynatma denetimlerini içeren bir kaplama tercih etme seçeneğiniz olur. Ancak kendinizi sunulan seçeneklerle sınırlamanıza gerek yoktur. ActionScript'i kullanarak video yükleme, görüntüleme ve oynatma işleminin ince ayarını yaparsınız, başka bir deyişle, kendi video oynatıcı kaplamanızı oluşturabilir veya istediğiniz daha az geleneksel bir şekilde de videonuzu kullanabilirsiniz. ActionScript'te video ile çalışılması, birçok sınıfın birleşimiyle çalışılmasını kapsar:

- Video sınıfı: Sahne Alanı'ndaki klasik video içeriği kutusu Video sınıfının bir örneğidir. Video sınıfı bir görüntüleme nesnesidir, bu nedenle diğer görüntüleme nesnelerine uygulanabilen konumlandırma, dönüştürmeler uygulama, filtreler ve harmanlama modları uygulama vb. gibi tekniklerin aynısı kullanılarak işlenebilir.
- StageVideo sınıfı: Video sınıfı genellikle yazılım kodu çözme ve oluşturma kullanır. Bir aygıtta GPU donanım hızlandırma mevcut olduğunda, uygulamanız StageVideo sınıfına geçerek donanım hızlandırmalı gösterimden en iyi şekilde faydalanabilir. StageVideo API'si, kodunuza StageVideo ve Video nesneleri arasında geçiş yapmasını söyleyen bir olaylar kümesi içerir. Sahne alanı videosu video oynatmaya bazı küçük sınırlamalar getirir. Uygulamanız bu sınırlamaları kabul ediyorsa StageVideo API'sini uygulayın. Bkz. "Yönergeler ve sınırlamalar" sayfa 519.
- NetStream sınıfı: ActionScript tarafından kontrol edilecek bir video dosyası yüklerken bir NetStream örneği video içeriğini; bu durumda video verileri akışını gösterir. NetStream örneğinin kullanılması, video verilerinin üzerinden beslendiği tünel niteliğindeki video dosyası bağlantısı olan bir NetConnection nesnesinin kullanılmasını kapsar.
- Camera sınıfı: Kullanıcının bilgisayarına bağlanmış bir kameradan gelen video verileriyle çalışıyorken, bir Camera örneği video içeriğinin kaynağını gösterir; kullanıcı kamerası ve video verilerinin kullanılmasını sağlar. Flash Player 11.4 ve AIR 3.4'te yeni bir özellik olarak StageVideo nesnesine besleme sağlamak için kamerayı da kullanabilirsiniz.

Harici video yüklerken, aşamalı indirme için standart bir web sunucusundan dosyayı yükleyebilir veya Adobe Flash® Media Server gibi özelleştirilmiş bir sunucu tarafından sunulan akış videosuyla çalışabilirsiniz.

Önemli kavramlar ve terimler

İşaret noktası Örneğin, zaman içindeki söz konusu anın bulunması amacıyla yer imi olarak hareket etmesi veya o anla ilişkilendirilmiş ek veriler sağlaması için video dosyasında zaman içindeki belirli bir ana yerleştirilebilen bir işaretçi.

Kodlama Bir biçimdeki video verilerinin başka bir video verisi biçimine dönüştürülmesi işlemi; örneğin, yüksek çözünürlüklü kaynak videonun alınıp İnternet'te sunulması için uygun bir biçime dönüştürülmesi.

Kare Video bilgilerinin tek bir parçası; her kare, zaman içindeki bir anlık görüntüyü temsil eden hareketsiz görüntüdür. Karelerin sırayla yüksek hızda oynatılmasıyla hareket görünümü oluşturulur.

Anahtar kare Karenin tüm bilgilerini içeren bir video karesi. Anahtar kareden sonra gelen diğer kareler, karenin tüm bilgilerini değil, yalnızca anahtar kareden nasıl farklılık gösterdiği hakkındaki bilgileri içerir.

Meta Veriler Video dosyası içine gömülmüş ve video yüklendiğinde alınan bir video dosyası hakkındaki bilgiler.

Aşamalı indirme Standart bir web sunucusundan video teslim edildiğinde, aşamalı indirme kullanılarak video verileri yüklenir, başka bir deyişle, sırayla video bilgileri yüklenir. Bunun avantajı, dosyanın tamamı indirilmeden önce videonun oynatılmaya başlayabilmesidir; ancak bu, videonun yüklenmemiş olan bölümüne atlanmasını önler.

Akışa alma Aşamalı indirmeye alternatif olarak, akışa alma olarak bilenen (bazen "gerçek akışa alma" da denir) bir teknik kullanılarak İnternet üzerinden video sunmak için özel bir video sunucusu kullanılabilir. Akışa alma sayesinde, izleyicinin bilgisayarı asla videonun tamamını bir defada indirmez. İndirme işlemini hızlandırmak için, herhangi bir anda bilgisayarın toplam video bilgilerinin yalnızca bir kısmına ihtiyacı vardır. Video içeriğinin dağıtımını özel bir sunucu denetlediğinden, videoya erişmek için videonun indirilmesini beklemeye gerek kalmadan videonun herhangi bir bölümüne her an erişilebilir.

Video biçimlerini anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe FLV video formatına ek olarak, Flash Player ve Adobe AIR uygulaması, MPEG-4 standart dosya formatlarında H.264 ve HE-AAC'de kodlanmış video ve sesi destekler. Bu formatlar, yüksek kaliteli videoyu daha düşük bit hızlarında akışa alır. Geliştiriciler, etkileyici video içeriği oluşturup sunmak için Adobe Premiere Pro ve Adobe After Effects gibi endüstri standardı araçlarını kullanabilir.

Tür	Biçim	Kap
Video	H.264	MPEG-4: MP4, M4V, F4V, 3GPP
Video	Sorenson Spark	FLV dosyası
Video	ON2 VP6	FLV dosyası
Ses	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4:MP4, M4V, F4V, 3GPP
Ses	Mp3	Mp3
Ses	Nellymoser	FLV dosyası
Ses	Speex	FLV dosyası

Daha fazla Yardım konusu

Flash Media Server: Desteklenen codec'ler

Adobe HTTP Dinamik Akış

Mobil aygıtlar için video kodlama

Android'de AIR birçok H.264 videosunun kodunu çözebilir. Ancak, H.264 videolarının yalnızca küçük bir alt kümesi mobil aygıtlarda kusursuz oynatmaya uygundur. Bunun nedeni bir çok cep telefonunun işleme gücü için sınırlandırılmasıdır. Mobil aygıtlar için Adobe Flash Player yerleşik donanım hızlandırmayı kullanarak H.264 videolarının kodunu çözebilir. Bu kod çözme işlemi daha az güç kullanımıyla daha iyi kalite sunar.

H.264 standardı çeşitli kodlama tekniklerini destekler. Karmaşık profillere ve düzeylere sahip videoları yalnızca en yüksek kalitedeki aygıtlar düzgün biçimde oynatabilir. Ancak, aygıtların büyük çoğunluğu temel profille kodlanmış videoyu oynatabilir. Mobil aygıtlarda, donanım hızlandırma bu tekniklerin bir alt kümesi için mevcuttur. Bu kodlama teknikleri alt kümesini ve kodlayıcı tarafından kullanılan ayarları profil ve düzey parametreleri tanımlar. Geliştiriciler için, birçok aygıtta sorunsuz şekilde oynatılabilen seçili çözünürlükteki videoyu kodlama anlamına gelir.

Donanım hızlandırmadan faydalanan çözünürlükler aygıttan aygıta değişse de, çoğu aygıt aşağıdaki standart çözünürlükleri destekler.

En boy oranı	Önerilen çözünürlükler		
4:3	640 × 480	512 × 384	480 × 360
16:9	640 × 360	512 x 288	480 × 272

Not: *Flash Player H.264 standardının her düzeyini ve profilini destekler. Bu önerilere uymak birçok aygıtta donanım hızlandırmayı ve daha iyi kullanıcı deneyimini sağlar. Bu önerilere uymak zorunlu değildir.*

Ayrıntılı bir açıklama ve Adobe Media Encoder CS5'teki kodlama ayarları için bkz. Mobil aygıtlarda Flash Player 10.1 için H.264 videosunu kodlama.

Not: *iOS'de, Video sınıfını kullanarak yalnızca Sorenson Spark ve On2 VP6 codec'leri kullanılarak kodlanmış videolar oynatılabilir. H.264 kodlu videoyu aygıtın video oynatıcısında, videonun URL'sini* `flash.net.navigateToURL()` *işlevini kullanıp başlatarak oynatabilirsiniz. Ayrıca, H.264 videoyu StageWebView nesnesinde görüntülenen bir html sayfasındaki <video> etiketini kullanarak da oynatabilirsiniz.*

Kodlanmış video dosyalarıyla Flash Player ve AIR uyumluluğu

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player 7, Sorenson™ Spark™ video codec bileşeniyle kodlanmış FLV dosyalarını destekler. Flash Player 8, Flash Professional 8'de Sorenson Spark veya On2 VP6 kodlayıcısı ile kodlanmış FLV dosyalarını destekler. On2 VP6 video codec bileşeni bir alfa kanalını destekler.

Flash Player 9.0.115.0 ve sonraki sürümler, standart MPEG-4 konteyner formatından türeyen dosyaları destekler. Bu dosyalar arasında, F4V, MP4, M4A, MOV, MP4V, 3GP ve 3G2 (H.264 videoyu veya HE-AAC v2 kodlanmış sesi ya da her ikisini içeriyorsa) yer alır. H.264, Sorenson veya On2'deki aynı kodlama profiliyle karşılaştırıldığında daha düşük bit hızlarında daha yüksek kaliteli video sağlar. HE-AAC v2, MPEG-4 video standardında tanımlanmış standart bir ses formatı olan AAC'nin uzantısıdır. HE-AAC v2, düşük bit hızlarında kodlama verimliliğini artırmak için dalga boyu yenilemesi (SBR) ve parametric stereo (PS) tekniklerini kullanır.

Aşağıdaki tabloda, desteklenen codec bileşenleri listelenmektedir. Bu tabloda ayrıca karşılık gelen SWF formatı ve bu dosyaları oynatmak için gerekli olan Flash Player ve AIR sürümleri gösterilmektedir:

Codec	SWF dosya formatı sürümü (en eski yayınlama sürümü)	Flash Player ve AIR (oynatma için gerekli en eski sürüm)
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3. Yalnızca Flash Player 8 ve sonraki sürümler, On2 VP6 videosunun yayınlanmasını ve oynatılmasını destekler.
H.264 (MPEG-4 Bölüm 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 Bölüm 3)	9	Flash Player 9 Update 3, AIR 1.0
Speex (ses)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

Adobe F4V ve FLV video dosyası formatlarını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe, Flash Player ve AIR uygulamasına içeriği akışa almak için F4V ve FLV video dosyası formatlarını sağlar. Bu video dosyası formatlarının tam açıklaması için, bkz. www.adobe.com/go/video_file_format_tr.

F4V video dosyası formatı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player Update 3 (9.0.115.0) ve AIR 1.0'dan itibaren, Flash Player ve AIR, ISO MP4 formatını esas alan Adobe F4V video formatını destekler. Bu formatın alt kümeleri, farklı özellikleri destekler. Flash Player, şu üst düzey kutulardan biriyle başlamak için geçerli bir F4V dosyası bekler:

- ftyp

  ftyp kutusu, belirli bir dosya formatını oynatmak için bir programın desteklemesi gereken özellikleri tanımlar.

- moov

  moov kutusu, F4V dosyasının etkin şekilde başlığıdır. F4V verilerinin yapısını tanımlayan diğer dosyaları içeren bir veya daha fazla kutu içerir. F4V dosyası yalnızca bir moov kutusu içermelidir.

- mdat

  mdat kutusu, F4V dosyasının veri yükünü içerir. FV dosyası yalnızca bir mdat kutusu içerir. mdat kutusu tek başına anlaşılamadığından, dosyada moov kutusunun da bulunması gerekir.

F4V dosyaları, en düşük adreste en önemli baytların en başta geldiği big-endian bayt sırasında çok baytlı tam sayıları destekler.

FLV video dosyası formatı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe FLV dosya formatı, Flash Player tarafından teslim edilmesi için kodlanmış ses ve video verileri içerir. QuickTime veya Windows Media video dosyasını bir FLV dosyasına dönüştürmek için Adobe Media Encoder ya da Sorenson™ Squeeze gibi bir kodlayıcı kullanabilirsiniz.

Not: *Flash uygulamasına içe aktarıp FLV dosyası olarak dışa aktararak FLV dosyaları oluşturabilirsiniz. Desteklenen video düzenleme uygulamalarından FLV dosyalarını dışa aktarmak için FLV Export eklentisini kullanabilirsiniz. Web sunucusundan FLV dosyalarını yüklemek için dosya adı uzantısını ve MIME türünü web sunucunuzla kaydedin. Web sunucusu belgelerinizi kontrol edin. FLV dosyalarının MIME türü* `video/x-flv` *şeklindedir. Daha fazla bilgi için, bkz. "Sunucuda barındırmak üzere FLV dosyalarını konfigüre etme hakkında" sayfa 510.*

FLV dosyaları hakkında daha fazla bilgi için bkz. "Video dosyaları için gelişmiş başlıklar" sayfa 509.

Harici ve gömülü video

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici video dosyalarının kullanılması, içe aktarılmış video kullandığınızda mevcut olmayan belirli yetenekleri sağlar:

- Uygulamanızda oynatma yavaşlamadan daha uzun video klipler kullanılabilir. Harici video dosyaları, önbellek kullanır, başka bir deyişle büyük dosyalar küçük parçalar halinde saklanır ve bu parçalara dinamik olarak erişilir. Bu nedenle, harici F4V ve FLV dosyaları için gömülü video dosyalarından daha az bellek gerekir.
- Harici bir video dosyası, içinde oynatıldığı SWF dosyasından farklı bir kare hızına sahip olabilir. Örneğin, SWF dosyasının kare hızını saniyede 30 kareye (fps) ve video kare hızını 21 fps'ye ayarlayabilirsiniz. Bu ayar, akıcı video oynatımı sağlamak için gömülü videodan daha iyi video denetimi sağlar. Ayrıca varolan SWF dosya içeriklerinin değiştirilmesine gerek kalmadan farklı kare hızlarında video dosyalarının oynatılmasını sağlar.
- Harici video dosyalarıyla, video yüklenirken SWF içeriğinin oynatılması kesintiye uğramaz. İçe aktarılan video dosyaları bazen belirli işlevleri gerçekleştirmek için (örneğin, bir CD-ROM sürücüsüne erişmek için) belgenin oynatılmasını kesintiye uğratabilir. Video dosyaları, oynatmayı kesintiye uğratmadan SWF içeriğinden bağımsız şekilde işlevleri gerçekleştirebilir.
- Olay işleyicilerini kullanarak video meta verilerine erişebildiğinizden, video içeriğine resim yazısı eklenmesi, harici FLV dosyalarında daha kolaydır.

Video sınıfını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Video sınıfı canlı akış videosunu, SWF dosyanıza gömmeden bir uygulamada görüntüleyebilmenizi sağlar. `Camera.getCamera()` yöntemini kullanarak canlı video yakalayıp oynatabilirsiniz. Ayrıca HTTP üzerinden veya yerel dosya sisteminden video dosyalarını oynatmak için Video sınıfını da kullanabilirsiniz. Projelerinizde Video sınıfını kullanmanın birçok farklı yolu vardır:

- NetConnection ve NetStream sınıflarını kullanarak dinamik olarak video dosyasını yükleyin ve videoyu bir Video nesnesinde görüntüleyin.
- Kullanıcının kamerasından girdi yakalayın. Daha fazla bilgi için bkz. "Kameralar ile çalışma" sayfa 526.
- FLVPlayback bileşenini kullanın.

- VideoDisplay denetimini kullanma.

Not: *Sahne Alanı'ndaki Video nesnesi örnekleri, Video sınıfının örnekleridir.*

Video sınıfı flash.media paketinde bulunsa da, flash.display.DisplayObject sınıfından miras alır. Bu nedenle, matris dönüştürmesi ve filtreler gibi tüm görüntüleme nesnesi işlevleri de Video örnekleri için geçerli olur.

Daha fazla bilgi için, bkz. "Görüntüleme nesnelerini işleme" sayfa 176, "Geometriyle çalışma" sayfa 213 ve "Görüntüleme nesnelerine filtre uygulama" sayfa 270.

Video dosyalarını yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

NetStream ve NetConnection sınıfları kullanılarak video yüklenmesi çok adımlı bir işlemdir. En iyi uygulama olarak, görüntüleme listesine Video nesnesi ekleme, Video örneğine NetStream nesnesini ekleme ve NetStream nesnesinin `play()` yöntemini çağırma adımları belirtilen sırada gerçekleştirilmelidir:

1. Bir NetConnection nesnesi oluşturun. Yerel bir video sunucusuna veya Adobe Flash Media Server 2 gibi sunucu kullanmayan bir video dosyasına bağlanıyorsanız, HTTP adresinden ya da yerel bir sürücüden video dosyalarını oynatmak için `null` öğesini `connect()` yöntemine iletin. Bir sunucuya bağlanıyorsanız, bu parametreyi sunucu üzerinde video dosyasını içeren uygulama URI'sine ayarlayın.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
```

2. Videoyu görüntüleyen yeni bir Video nesnesi oluşturun ve söz konusu nesneyi aşağıdaki parçacıkta gösterildiği şekilde sahne alanı görüntüleme listesine ekleyin:

```
var vid:Video = new Video();
addChild(vid);
```

3. NetConnection nesnesini yapıcıya bir argüman olarak ileterek bir NetStream nesnesi oluşturun. Aşağıdaki parçacık bir NetStream nesnesini NetConnection örneğine bağlar ve akış için olay işleyicilerini ayarlar:

```
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. Aşağıdaki parçacıkta görüldüğü şekilde Video nesnesinin `attachNetStream()` yöntemini kullanarak NetStream nesnesini Video nesnesine ekleyin:

```
vid.attachNetStream(ns);
```

5. Video yürütmesini başlatacak argüman olarak video dosyası url'si ile NetStream nesnesinin `play()` yöntemini çağırın. Aşağıdaki parçacık, "video.mp4" adlı bir video dosyasını SWF dosyasıyla aynı dizinde yükler ve yürütür:

```
ns.play("video.mp4");
```

Daha fazla Yardım konusu

Flex: Spark VideoPlayer denetimi

spark.components.VideoDisplay

Video oynatımını denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

NetStream sınıfı, video oynatımının denetlenmesine yönelik dört ana yöntem sunar:

`pause()`: Bir video akışının oynatılmasını duraklatır. Video zaten duraklatılmışsa, bu yöntemin çağrılması herhangi bir işlem gerçekleştirmez.

`resume()`: Duraklatılan bir video akışının oynatılmasını kaldığı yerden devam ettirir. Video zaten oynatılıyorsa, bu yöntemin çağrılması herhangi bir işlem gerçekleştirmez.

`seek()`: Belirtilen konuma (akışın başından itibaren saniye cinsinden uzaklık) en yakın olan anahtar kareyi arar.

`togglePause()`: Akışın oynatılmasını duraklatır veya kaldığı yerden devam ettirir.

Not: *Herhangi bir `stop()` yöntemi yoktur. Bir akışı durdurmak için, oynatmayı duraklatmanız ve video akışının başını aramanız gerekir.*

Not: *`play()` yöntemi oynatmayı yeniden başlatmaz, video dosyalarının yüklenmesi için kullanılır.*

Aşağıdaki örnek, birçok farklı düğme kullanılarak videonun nasıl denetlendiğini gösterir. Şu örneği çalıştırmak için, yeni bir belge oluşturun ve çalışma alanına dört düğme örneği (`pauseBtn`, `playBtn`, `stopBtn` ve `togglePauseBtn`) ekleyin:

```
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();
}
```

Video oynatılırken `pauseBtn` düğme örneğinin tıklatılması, video dosyasının duraklatılmasına neden olur. Video zaten duraklatılmışsa, bu düğme herhangi bir etki oluşturmaz. `playBtn` düğme örneğinin tıklatılması, oynatma önceden duraklatılmışsa, video oynatımını yeniden başlatır; aksi takdirde video zaten oynatılıyorsa herhangi bir etki oluşturmaz.

Video akışının sonunu algılama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Video akışının başını ve sonunu dinlemek için, NetStream örneğine bir olay dinleyicisi ekleyerek `netStatus` olayını dinlemeniz gerekir. Aşağıdaki kod, video oynatımı sırasında çeşitli kodların nasıl dinleneceğini gösterir:

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

Önceki kod, şu çıktıyı oluşturur:

```
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
```

Özellikle dinlemek istediğiniz iki kod, video oynatımının başını ve sonunu bildiren "NetStream.Play.Start" ve "NetStream.Play.Stop" kodlarıdır. Aşağıdaki kod parçası, bu iki koda filtre uygulamak ve bir mesajı izlemek için bir anahtar deyimini kullanır:

```
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;
    }
}
```

`netStatus` olayını (`NetStatusEvent.NET_STATUS`) dinleyerek, geçerli video oynatımı bittikten sonra bir sonraki videoyu oynatma listesine yükleyen bir video oynatıcısı oluşturabilirsiniz.

Tam ekran modunda video oynatma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve AIR, video oynatmanız için tam ekran uygulama oluşturmayı sağlar ve videoyu tam ekrana ölçeklemeyi destekler.

Masaüstünde tam ekran modunda çalışan AIR içeriği için video girişi durana kadar veya kullanıcı tam ekran modundan çıkana kadar oynatma sırasında sistem ekran koruyucusu ve güç tasarrufu seçenekleri devre dışı bırakılır.

Tam ekran modunu kullanma konusunda ayrıntılar için, bkz. "Tam ekran modunda çalışma" sayfa 169.

Bir tarayıcıda Flash Player için tam ekran modunu etkinleştirme

Bir tarayıcıda Flash Player için tam ekran modunu uygulayabilmeniz için öncelikle uygulamanızın Yayınla şablonu üzerinden bu modu etkinleştirin. Tam ekrana izin veren şablonlar, bir `allowFullScreen` parametresi içeren `<object>` ve `<embed>` etiketlerini barındırır. Şu örnek, bir `<embed>` etiketinde `allowFullScreen` parametresini gösterir.

```
<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>
```

Flash'ta, Dosya -> Yayınlama Ayarları seçeneklerini belirleyin ve Yayınlama Ayarları iletişim kutusunun HTML sekmesinde Yalnızca Flash - Tam Ekran Moduna İzin Ver şablonunu seçin.

Flex'te, HTML şablonunun tam ekranı destekleyen `<object>` ve `<embed>` etiketlerini içerdiğinden emin olun.

Tam ekran modunu başlatma

Bir tarayıcıda çalışan Flash Player içeriği için, fare tıklatma veya tuş basışına yanıt olarak video için tam ekran modunu başlatırsınız. Örneğin, kullanıcı Tam Ekran etiketli bir düğmeyi tıklattığında veya bağlam menüsünden Tam Ekran komutunu seçtiğinde,tam ekran modunu başlatabilirsiniz. Kullanıcıya yanıt vermek için, üzerinde eylemin gerçekleştiği nesneye bir olay dinleyicisi ekleyin. Şu kod, kullanıcının tam ekran moduna girmek için tıklattığı bir düğmeye olay dinleyicisi ekler:

```
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;

}
```

Kod, `Stage.displayState` özelliğini `StageDisplayState.FULL_SCREEN` olarak ayarlayarak tam ekran modunu başlatır. Bu kod, video ölçekleme, sahne alanında kapladığı alanla orantılı olacak şekilde sahne alanının tamamını tam ekrana ölçekler.

`fullScreenSourceRect` özelliği, tam ekrana ölçeklemek için sahne alanının belirli bir alanını belirtmenize olanak sağlar. İlk olarak, tam ekrana ölçeklemek istediğiniz dikdörtgeni tanımlayın. Daha sonra bunu `Stage.fullScreenSourceRect` özelliğine atayın. `fullScreenButtonHandler()` işlevinin bu sürümü, yalnızca videoyu tam ekrana ölçekleyen iki ek kod satırı ekler.

```
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;
}
```

Bu örnek fare tıklatmasına yanıt olarak bir olay işleyicisi çağırsa da, tam ekran moduna gitme tekniği, Flash Player ve AIR uygulamaları için aynıdır. Ölçeklemek istediğiniz dikdörtgeni tanımlayın ve sonra `Stage.displayState` özelliğini ayarlayın. Daha fazla bilgi için bkz. Adobe Flash Platform için ActionScript 3.0 Başvurusu.

Daha sonra gelen örneğin tamamı, video için bağlantı ve NetStream nesnesi oluşturan videoyu oynatmaya başlayan kodu ekler.

```
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
        }
    }
}
```

`onMetaData()` işlevi, varsa, video meta verilerinin işlenmesine yönelik geri çağrı işlevidir. Geri çağrı işlevi, çalışma zamanının herhangi türde bir oluşum veya olaya yanıt olarak çağırdığı işlevdir. Bu örnekte, `onMetaData()` işlevi, işlev sağlama gereksinimini karşılayan bir kalıntıdır. Daha fazla bilgi için, bkz. "Meta veri ve işaret noktaları için geri çağrı yöntemleri yazma" sayfa 491

Tam ekran modundan çıkma

Kullanıcı, Escape tuşu gibi klavye kısayollarından birini girerek tam ekran modundan çıkabilir. ActionScript'te `Stage.displayState` özelliğini `StageDisplayState.NORMAL` olarak ayarlayarak tam ekran modunu sonlandırabilirsiniz. Şu örnekteki kod, NetStream.Play.Stop `netStatus` olayı oluştuğunda tam ekran modunu sonlandırır.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

Tam ekran donanım ivmesi

Sahne Alanı'nın dikdörtgen bir alanını tam ekran moduna yeniden ölçeklediğinizde, Flash Player veya AIR mevcutsa veya etkinde donanım hızlandırmayı kullanır. Çalışma zamanı, videonun veya sahne alanının bir kısmının tam ekran boyutuna ölçeklenmesini hızlandırmak için bilgisayardaki video adaptörünü kullanır. Bu koşullar altında, Flash Player uygulamaları genellikle bir Video sınıfından (veya Flash Player 11.4/AIR 3.4 ve üstünde Camera sınıfından) StageVideo sınıfına geçilmesinden avantaj sağlayabilir.

Tam ekran modunda donanım hızlandırmasıyla ilgili daha fazla bilgi için bkz. "Tam ekran modunda çalışma" sayfa 169. StageVideo hakkında daha fazla bilgi için bkz. "Donanım hızlandırmalı gösterim için StageVideo sınıfını kullanma" sayfa 517.

Video dosyalarını akışa alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Media Server'dan dosyaları akışa almak için, NetConnection ve NetStream sınıflarını kullanarak uzak bir sunucu örneğine bağlanabilir ve belirtilen bir akışı oynatabilirsiniz. Bir Real-Time Messaging Protocol (RTMP) sunucusu belirtmek için, `null` değeri iletmek yerine, "rtmp://localhost/appName/appInstance" gibi istediğiniz RTMP URL'sini `NetConnection.connect()` yöntemine iletirsiniz. Belirtilen Flash Media Server'dan belirli bir canlı veya kayıtlı akışı oynatmak üzere, `NetStream.publish()` tarafından yayınlanan canlı veriler için tanımlayıcı bir ad ya da `NetStream.play()` yöntemine oynatma için kayıtlı bir dosya adı iletirsiniz.

Sunucuya video gönderme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Video veya kamera nesneleri içeren daha karmaşık uygulamalar oluşturmak istiyorsanız, Flash Media Server akış ortamı yeteneklerinin birleşimini ve ortam uygulamalarının oluşturulup geniş kitlelere dağıtılmasına yönelik bir geliştirme ortamını sunar. Bu birleşim, geliştiricilerin İstek Üzerine Video, canlı web olayı yayınları ve MP3 akışı ile video web günlüğü, video mesajlaşma ve multimedya sohbet ortamları gibi uygulamalar oluşturmasını sağlar. Daha fazla bilgi için www.adobe.com/go/learn_fms_docs_tr adresinde çevrimiçi Flash Media Server belgelerine bakın.

İşaret noktalarını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İşaret noktalarını kodlama sırasında bir Adobe F4V veya FLV video dosyasına gömebilirsiniz. Eskiden işaret noktaları, projeksiyonculara film makarasının bitmek üzere olduğunu belirten görsel bir sinyal vermek için filmlere gömülürdü. Adobe F4V ve FLV video formatlarında işaret noktası, video akışında gerçekleştiği anda uygulamanızda bir veya daha çok eylemi tetiklemenize olanak sağlar.

Flash videosu ile birçok farklı türde işaret noktası kullanabilirsiniz. Oluşturduğunuzda video dosyasına gömdüğünüz işaret noktalarıyla etkileşim kurmak için ActionScript'i kullanabilirsiniz.

- Gezinme işaret noktaları: Video dosyasını kodlarken gezinme işaret noktalarını video akışına ve meta veri paketine gömebilirsiniz. Kullanıcıların dosyanın belirtilen bir bölümünü bulmasını sağlamak için gezinme işaret noktalarını kullanırsınız.
- Olay işaret noktaları: Video dosyasını kodlarken olay işaret noktalarını video akışına ve meta veri paketine gömebilirsiniz. Video oynatımı sırasında belirtilen noktalarda tetiklenen olayları işlemek için kod yazabilirsiniz.
- ActionScript işaret noktaları: ActionScript işaret noktaları yalnızca Flash FLVPlayback bileşeni için kullanılabilir. ActionScript işaret noktaları, ActionScript koduyla oluşturup eriştiğiniz harici işaret noktalarıdır. Video oynatımına göre bu işaret noktalarını tetiklemek için kod yazabilirsiniz. Video oynatıcısı bu işaret noktalarını ayrı ayrı izlediğinden, bu işaret noktaları, gömülü işaret noktalarından daha az (saniyede onu kadar) kesinliğe sahiptir. Kullanıcıların bir işaret noktasına gitmesini istediğiniz bir uygulama oluşturmayı planlıyorsanız, ActionScript işaret noktalarını kullanmak yerine dosyayı kodlarken işaret noktaları oluşturmanız ve gömmeniz gerekir. Daha kesin oldukları için işaret noktalarını FLV dosyasına gömmeniz gerekir.

Gezinme işaret noktaları belirtilen işaret noktası konumunda bir anahtar kare oluşturur, böylece video oynatıcının oynatma kafasını o konuma taşımak için kod kullanabilirsiniz. Video dosyasında kullanıcıların bulabileceği belirli noktalar belirleyebilirsiniz. Örneğin, videonuzda birden çok bölüm veya parça olabilir ve siz video dosyasına gezinme işaret noktaları gömerek videoyu denetleyebilirsiniz.

Adobe video dosyalarını işaret noktalarıyla kodlama hakkında daha fazla bilgi almak için, bkz. "İşaret noktalarını gömme", *Flash'ı Kullanma*.

ActionScript yazarak işaret noktası parametrelerine erişebilirsiniz. İşaret noktası parametreleri, geri çağrı işleyicisi tarafından alınan olay nesnesinin bir bölümüdür.

FLV dosyası belirli bir işaret noktasına ulaştığında kodunuzda belirli eylemleri tetiklemek için, `NetStream.onCuePoint` olay işleyicisini kullanın.

F4V video dosyasında bir işaret noktası için eylem senkronize etmek üzere, `onMetaData()` veya `onXMPData()` geri çağrı işlevlerinden işaret noktası verilerini almanız ve ActionScript 3.0'da Timer sınıfını kullanarak işaret noktasını tetiklemeniz gerekir. F4V işaret noktaları hakkında daha fazla bilgi almak için bkz. "onXMPData() öğesini kullanma" sayfa 502.

İşaret noktalarını ve meta verileri işleme hakkında daha fazla bilgi için, bkz. "Meta veri ve işaret noktaları için geri çağrı yöntemleri yazma" sayfa 491.

Meta veri ve işaret noktaları için geri çağrı yöntemleri yazma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Oynatıcı tarafından belirli meta veriler alındığında veya belirli işaret noktalarına ulaşıldığında uygulamanızda eylemleri tetikleyebilirsiniz. Bu olaylar gerçekleştiğinde, belirli geri çağrı yöntemlerini olay işleyicileri olarak kullanmanız gerekir. NetStream sınıfı, oynatma sırasında oluşabilecek şu meta veri olaylarını belirtir: `onCuePoint` (yalnızca FLV dosyaları), `onImageData`, `onMetaData`, `onPlayStatus`, `onTextData` ve `onXMPData`.

Bu işleyiciler için geri çağrı yöntemleri yazmanız gerekir, aksi takdirde Flash çalışma zamanı hata atabilir. Örneğin, şu kod SWF dosyasının bulunduğu aynı klasörde video.flv adında bir FLV dosyasını oynatır:

```
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);
```

Önceki kod, video.flv adında yerel bir video yükler ve gönderilecek `asyncError` (`AsyncErrorEvent.ASYNC_ERROR`) öğesini dinler. Yerel eşzamansız koddan bir istisna atıldığında bu olay gönderilir. Bu durumda, video dosyası meta veri veya işaret noktası bilgileri içerdiğinde ve uygun dinleyiciler tanımlanmamış olduğunda gönderilir. Önceki kod, `asyncError` olayını işler ve siz video dosyasının meta veri veya işaret noktası bilgileriyle ilgilenmiyorsanız hatayı yok sayar. Meta veri ve birçok işaret noktası içeren bir FLV dosyanız varsa, `trace()` işlevi şu hata mesajlarını görüntüler:

```
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.
```

NetStream nesnesi, `onMetaData` veya `onCuePoint` geri çağrı yöntemini bulamadığından hatalar gerçekleşir. Uygulamalarınızda bu geri çağrı yöntemlerini tanımlamanın birçok yolu vardır.

Daha fazla Yardım konusu

Flash Media Server: Akışlarda meta veriyi işleme

NetStream nesnesinin istemci özelliğini bir Object öğesine ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`client` özelliğini bir Object öğesine veya NetStream öğesinin alt sınıfına ayarlayarak, `onMetaData` ve `onCuePoint` geri çağrı yöntemlerini yeniden yönlendirebilir ya da tamamen yoksayabilirsiniz. Şu örnek, `asyncError` olayını dinlemeden geri çağrı yöntemlerini yoksaymak için boş bir Object öğesini nasıl kullanabileceğinizi gösterir:

```
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);
```

`onMetaData` veya `onCuePoint` geri çağrı yöntemlerini dinlemek isterseniz, aşağıdaki kod parçasında gösterildiği gibi, bu geri çağrı yöntemlerini işlemek için yöntemler tanımlamanız gerekir:

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

Önceki kod, `onMetaData` geri çağrı yöntemini dinler ve bir dizeyi izleyen `metaDataHandler()` yöntemini çağırır. Flash çalışma zamanı bir işaret noktasıyla karşılaşırsa, herhangi bir `onCuePoint` geri çağrı yöntemi tanımlanmamış olsa da, hata oluşturulmaz.

Özel bir sınıf oluşturma ve geri çağrı yöntemlerini işlemek için yöntemleri tanımlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Şu kod, NetStream nesnesinin `client` özelliğini özel bir sınıf olan CustomClient sınıfına ayarlar ve böylece geri çağrı yöntemleri için işleyiciler tanımlanır:

```
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);
```

CustomClient sınıfı şöyledir:

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

CustomClient sınıfı, `onMetaData` geri çağrı işleyicisi için bir işleyici tanımlar. Bir işaret noktasıyla karşılaşıldıysa ve `onCuePoint` geri çağrı işleyicisi çağrıldıysa, bir `asyncError` olayı (`AsyncErrorEvent.ASYNC_ERROR`) gönderilerek "flash.net.NetStream, onCuePoint geri çağrısını çağıramadı." mesajını bildirir. Bu hatayı önlemek için, CustomClient sınıfınızda bir `onCuePoint` geri çağrı yöntemini tanımlamanız veya `asyncError` olayı için bir olay işleyicisi tanımlamanız gerekir.

NetStream sınıfını genişletme ve geri çağrı yöntemlerini işlemek için yöntemler ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Şu kod, daha sonra bir kod listesinde tanımlanan, CustomNetStream sınıfının bir örneğini oluşturur:

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

Şu kod listesi, NetStream sınıfını genişleten, gerekli NetConnection nesnesinin oluşumunu işleyen ve `onMetaData` ve `onCuePoint` geri çağrı işleyici yöntemlerini işleyen CustomNetStream sınıfını tanımlar:

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

CustomNetStream sınıfında `onMetaData()` ve `onCuePoint()` yöntemlerini yeniden adlandırmak isterseniz, şu kodu kullanabilirsiniz:

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

NetStream sınıfını genişletme ve dinamik hale getirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

NetStream sınıfını genişletebilir ve `onCuePoint` ve `onMetaData` geri çağrı işleyicileri dinamik olarak eklenebilecek şekilde alt sınıfı dinamik duruma getirebilirsiniz. Aşağıdaki listede bu gösterilmektedir:

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

DynamicCustomNetStream sınıfı şöyledir:

```
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);
        }
    }
}
```

`onMetaData` ve `onCuePoint` geri çağrı işleyicileri için herhangi bir işleyici olmadığında da, DynamicCustomNetStream sınıfı dinamik olduğundan hata atılmaz. `onMetaData` ve `onCuePoint` geri çağrı işleyicileri için yöntemleri tanımlamak istiyorsanız, şu kodu kullanabilirsiniz:

```
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");
}
```

NetStream nesnesinin istemci özelliğini this olarak ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`client` özelliği `this` olarak ayarlandığında, uygulama `onMetaData()` ve `onCuePoint()` yöntemlerinin geçerli kapsamında arama yapar. Aşağıdaki örnekte bunu görebilirsiniz:

```
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);
```

`onMetaData` veya `onCuePoint` geri çağrı işleyicileri çağrılırsa ve geri çağrıyı işleyecek bir yöntem yoksa herhangi bir hata oluşturulmaz. Bu geri çağrı işleyicilerini işlemek için, aşağıdaki kod parçasında görüldüğü gibi bir `onMetaData()` ve `onCuePoint()` yöntemi oluşturun:

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

İşaret noktalarını ve meta verileri kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Video oynatılırken işaret noktası ve meta veri olaylarını yakalamak ve işlemek için NetStream geri çağrı yöntemlerini kullanın.

İşaret noktalarını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki tablo, Flash Player uygulamasında F4V ve FLV işaret noktalarını yakalamak için kullanabileceğiniz geri çağrı yöntemlerini açıklar.

Çalışma zamanı	F4V	FLV
Flash Player 9/ AIR1.0		OnCuePoint
		OnMetaData

Çalışma zamanı	F4V	FLV
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

Aşağıdaki örnek, basit bir `for..in` döngüsünü kullanarak, `onCuePoint()` işlevinin aldığı `infoObject` parametresindeki özelliklerin her birini yineler. İşaret noktası verileri aldığında bir mesaj görüntülemek için trace() işlevini çağırır:

```
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]);
    }
}
```

Şu çıktı görüntülenir:

```
parameters:
name: point1
time: 0.418
type: navigation
```

Bu kod, geri çağrı yönteminin üzerinde çalıştığı nesneyi ayarlamak için çok sayıdaki teknikten birini kullanır. Başka teknikler de kullanabilirsiniz; daha fazla bilgi için, bkz. "Meta veri ve işaret noktaları için geri çağrı yöntemleri yazma" sayfa 491.

Video meta verilerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İşaret noktaları da dahil olmak üzere, video dosyanızdaki meta verilere erişmek için `OnMetaData()` ve `OnXMPData()` işlevlerini kullanabilirsiniz.

OnMetaData() öğesini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Meta veriler arasında, süre, genişlik, yükseklik ve kare hızı gibi video dosyanız hakkında bilgiler yer alır. Video dosyanıza eklenen meta veri bilgileri, video dosyasını kodlamak için kullandığınız yazılıma bağlıdır.

```
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]);
    }
}
```

Önceki kod, şunun gibi bir çıktı oluşturur:

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

Videonuzda ses yoksa, kodlama sırasında meta verilere ses bilgisi eklenmediğinden, sesle ilgili meta veri bilgileri (örn. `audiodatarate`) `undefined` değerini döndürür.

Önceki kodda, işaret noktası bilgileri görüntülenmiyordu. İşaret noktası meta verilerini görüntülemek için, Object içindeki öğeleri art arda görüntüleyen şu işlevi kullanabilirsiniz:

```
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);
        }
    }
}
```

`onMetaData()` yönteminde `infoObject` parametresini izlemek için önceki kod parçasının kullanılması şu çıktıyı oluşturur:

```
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
```

Şu örnek, MP4 video için meta verileri görüntüler. Ayrıca meta verileri yazdığı, `metaDataOut` adında bir TextArea nesnesinin olduğunu varsayar.

```
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;
        }
    }
}
```

`onMetaData()` işlevi bu video için şu çıktıyı üretmiştir:

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

Bilgi nesnesini kullanma

Aşağıdaki tablo, aldığı Objecy öğesinde `onMetaData()` geri çağrı işlevine iletilen video meta verileri için olası değerleri gösterir:

Parametre	Açıklama
aacaot	AAC ses nesnesi türü; 0, 1 veya 2 desteklenir.
avclevel	10, 11, 20, 21, vb. gibi AVC IDC düzey sayısı.
avcprofile	55, 77, 100, vb. gibi AVC profil sayısı.
audiocodecid	Kullanılmış olan ses codec bileşenini (kodlama/kod çözme tekniği) belirten bir dize; örneğin, ".Mp3" veya "mp4a"
audiodatarate	Saniyede kilobayt cinsinden, sesin kodlandığı hızı belirten bir sayı.
audiodelay	Orijinal FLV dosyasının "0 zamanının", FLV dosyasında ne zaman bulunduğunu belirten bir sayı. Video içeriğinin sesle düzgün şekilde senkronize edilmesi için küçük bir miktar ertelenmesi gerekir.
canSeekToEnd	FLV dosyası, aşamalı indirme video dosyasının sonuna kadar arama yapılmasını sağlayan son karedeki anahtar kareyle kodlanmışsa `true` olan bir Boolean değeri. FLV dosyası son karedeki bir anahtar kareyle kodlanmamışsa bu `false` olur.
cuePoints	FLV dosyasında gömülü her işaret noktası için birer tane olmak üzere nesneler dizisi. FLV dosyası herhangi bir işaret noktası içermezse, değer tanımsız olur. Her nesne şu özelliklere sahiptir: • `type`: "gezinme" veya "olay" olarak işaret noktasının türünü belirten bir dize. • `name`: işaret noktasının adı olan bir dize. • `time`: işaret noktasının zamanı olan üç ondalık basamak kesinliğinde (milisaniye) saniye cinsinden bir sayı. • `parameters`: işaret noktaları oluşturulurken kullanıcı tarafından atanan ad-değer çiftlerine sahip isteğe bağlı bir nesne.
duration	Video dosyasının saniye cinsinden süresini belirten bir sayı.
framerate	FLV dosyasının kare hızını belirten bir sayı.
height	FLV dosyasının piksel cinsinden yüksekliğini belirten bir sayı.
seekpoints	Milisaniye cinsinden zaman damgası olarak kullanılabilir anahtar kareleri listeleyen bir dizi. İsteğe bağlı.
etiketler	MP4 dosyalarının ID3 etiketlerine eşdeğer olan "ilst" atomundaki bilgileri temsil eden anahtar-değer çiftleri dizisi. iTunes bu etiketleri kullanır. Varsa, resim görüntülemek için kullanılabilir.
trackinfo	Örnek açıklama kimliği gibi, MP4 dosyasındaki tüm parçalar hakkında bilgi sağlayan nesne.
videocodecid	Videoyu kodlamak için kullanılmış olan codec sürümünü belirten bir dize. - örneğin, "avc1" veya "VP6F"

Parametre	Açıklama
videodatarate	FLV dosyasının video verisi hızını belirten bir sayı.
videoframerate	MP4 videosunun kare hızı.
width	FLV dosyasının piksel cinsinden genişliğini belirten bir sayı.

Aşağıdaki tablo, `videocodecid` parametresi için olası değerleri gösterir:

videocodecid	Codec adı
2	Sorenson H.263
3	Ekran videosu (yalnızca SWF sürüm 7 ve sonrası)
4	VP6 (yalnızca SWF sürüm 8 ve sonrası)
5	Alfa kanalı ile VP6 video (yalnızca SWF sürüm 8 ve sonrası)

Aşağıdaki tablo, `audiocodecid` parametresi için olası değerleri gösterir:

audiocodecid	Codec Adı
0	sıkıştırılmamış
1	ADPCM
2	Mp3
4	Nellymoser @ 16 kHz mono
5	Nellymoser, 8kHz mono
6	Nellymoser
10	AAC
11	Speex

onXMPData() öğesini kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

`onXMPData()` geri çağrı işlevi, Adobe F4V veya FLV video dosyasında gömülü olan, Adobe Genişletilebilir Meta Veriler Platformu'na (XMP) özgü bilgileri alır. XMP meta verileri, diğer video meta verilerinin yanı sıra işaret noktaları da içerir. XMP meta verileri desteği, Flash Player 10 ve Adobe AIR 1.5 ile sunulmuş olup sonraki sürümler tarafından desteklenir.

Şu örnek, XMP meta verilerinde işaret noktası verilerini işler:

```
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);
        }
    }
}
```

startrekintro.f4v adında kısa bir video dosyası için, bu örnek aşağıdaki izleme satırlarını üretir. Satırlar, XMP meta verilerinde gezinme ve olay işaret noktaları için işaret noktası verilerini gösterir:

```
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
```

Not: *XMP verilerinde, zaman saniye olarak değil, DVA Onay İşaretleri olarak saklanır. İşaret noktası zamanını hesaplamak için, başlangıç zamanını kare hızına bölün. Örneğin, 254016000000 değerine bölünmüş 7695905817600 başlangıç zamanı, 30:30 değerine eşittir.*

Kare hızını da içeren ham XMP meta verilerinin tamamını görmek için, `onXMPData()` işlevinin başında ikinci ve üçüncü `trace()` deyimlerinin başında bulunan yorum tanımlayıcılarını (//) kaldırın.

XMP hakkında daha fazla bilgi için bkz:

- partners.adobe.com/public/developer/xmp/topic.html
- www.adobe.com/devnet/xmp/

Görüntü meta verilerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`onImageData` olayı, görüntü verilerini AMF0 veri kanalı üzerinden bir bayt dizisi olarak gönderir. Veriler JPEG, PNG veya GIF formatları. Bu bilgileri işlemek için, `onCuePoint` ve `onMetaData` öğelerinin geri çağrı yöntemlerini tanımladığınız aynı şekilde `onImageData()` geri çağrı yöntemini de tanımlayın. Aşağıdaki örnek, bir `onImageData()` geri çağrı yöntemini kullanarak görüntü verilerine erişir ve bu verileri görüntüler:

```
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);
}
```

Metin meta verilerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`onTextData` olayı, AMF0 veri kanalı üzerinden metin verileri gönderir. Metin verisi UTF-8 formatındadır ve 3GP zamanlanmış metin belirtimini esas alarak formatlama hakkında ek bilgiler içerir. Belirtim, standartlaştırılmış bir altyazı formatını tanımlar. Bu bilgileri işlemek için, `onCuePoint` veya `onMetaData` öğesinin geri çağrı yöntemlerini tanımladığınız aynı şekilde `onTextData()` geri çağrı yöntemini de tanımlayın. Aşağıdaki örnekte, `onTextData()` yöntemi parça kimliğini ve karşılık gelen parça metnini görüntüler.

```
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);
}
```

NetStream etkinliğini izleme

Flash Player 10.3 ve üstü, Adobe AIR 2.7 ve üstü

Medya kullanımı analizini ve raporlamasını desteklemek için gereken bilgileri toplamak üzere NetStream etkinliğini izleyebilirsiniz. Bu bölümde geçen izleme özellikleri, medyayı görüntüleyen belirli video oynatıcıyla kapalı eşleme yapmadan veri toplayan medya ölçüm kitaplıkları oluşturmanıza izin verir. Bu, istemci geliştiricilerinizin kitaplığınızı kullanırken sık kullandıkları video oynatıcılarını seçmelerine izin verir. Bir uygulamadaki NetStream nesnelerinin oluşturulmasını ve etkinliklerini izlemek için NetMonitor sınıfını kullanın. NetMonitor sınıfı belirli bir zamanda etkin olan NetStream öğelerinin listesini sağlar ve ayrıca her NetStream nesnesi oluşturulduğunda bir olay gönderir.

Bir NetStream nesnesi oynatılan medya türüne bağlı olarak aşağıdaki tabloda listelenen olayları gönderir:

Olay	Aşamalı indirme	RTMP akışı	HTTP akışı
NetStream.Play.Start	Evet	Evet	Hayır
NetStream.Play.Stop	Evet	Evet	Hayır
NetStream.Play.Complete	Evet	Evet	Hayır
NetStream.SeekStart.Notify	Evet	Evet	Evet
NetStream.Seek.Notify	Evet	Evet	Evet
NetStream.Unpause.Notify	Evet	Evet	Evet
NetStream.Unpause.Notify	Evet	Evet	Evet
NetStream.Play.Transition	Uygulanamaz	Evet	Uygulanamaz
NetStream.Play.TransitionComplete	Uygulanamaz	Evet	Uygulanamaz
NetStream.Buffer.Full	Evet	Evet	Evet
NetStream.Buffer.Flush	Evet	Evet	Evet
NetStream.Buffer.Empty	Evet	Evet	Evet

Bir NetStream örneğiyle ilişkili NetStreamInfo nesnesi ayrıca medyada karşılaşılan en son meta verileri ve XMP veri nesnelerini saklar.

Medya HTTP akışıyla oynatıldığında, uygulama medya akışının tam kontrolüne sahip olduğundan NetStream.Play.Start, NetStream.Play.Stop ve NetStream.Play.Complete öğeleri gönderilmez. Bir video oynatıcısı HTTP akışları için bu olayları sentezlemeli ve göndermelidir.

Benzer şekilde, NetStream.Play.Transition ve NetStream.Play.TransitionComplete öğeleri aşamalı indirme veya HTTP medyası için de gönderilmez. Dinamik bit hızı değiştirme bir RTMP özelliğidir. HTTP akışını kullanan bir video oynatıcı benzer bir özelliği destekliyorsa, oynatıcı geçiş olaylarını sentezleyebilir ve gönderebilir.

Daha fazla Yardım konusu

Adobe Developer Connection: Measuring video consumption in Flash (Adobe Geliştirici Bağlantısı: Flash uygulamasında video tüketimini ölçme)

NetStream olaylarını izleme

İki tür olay değerli kullanım verileri sağlar: `netStatus` ve `mediaTypeData`. Ek olarak, NetStream oynatma kafasının konumunu düzenli bir şekilde günlüğe kaydetmek için bir zamanlayıcı kullanılabilir.

`netStatus` olayları, bir kullanıcının akışın ne kadarını görüntülediğini belirlemek için kullanabileceğiniz bilgiler sağlar. Arabellek ve RTMFP akışı geçiş olayları ayrıca `netStatus` olayına neden olur.

`mediaTypeData` olayları meta veri ve XMP veri bilgileri sağlar. Netstream.Play.Complete olayı bir `mediaTypeData` olayı olarak gönderilir. İşaret noktaları, metin ve görüntüler dahil olmak üzere akış içinde gömülü olan diğer veriler de `mediaTypeData` olayları aracılığıyla kullanılabilir durumdadır.

Aşağıdaki örnek bir uygulamadaki herhangi bir etkin NetStream öğesindeki durum ve veri olaylarını izleyen bir sınıfın nasıl oluşturulacağını gösterir. Normalde, bu tür bir sınıf analiz etmeyi düşündüğü verileri bir koleksiyon sunucusuna yükler.

```
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
            }
        }

    }
}
```

Oynatıcı etki alanını belirleme

Bir kullanıcının medya içeriğini görüntülediği web sayfasının URL'sine ve etki alanına her zaman kolaylıkla ulaşılamaz. Barındıran web sitesi tarafından izin verilirse tam URL'yi almak için ExternalInterface sınıfını kullanabilirsiniz. Ancak, üçüncü taraf video oynatıcılarına izin veren bazı web siteleri ExternalInterface öğesinin kullanılmasına izin vermez. Bu gibi durumlarda, Security sınıfının `pageDomain` özelliğinden geçerli web sayfasının etki alanını alabilirsiniz. Tam URL, güvenlik ve gizlilik nedenleriyle açıkça gösterilmez.

Sayfa etki alanı Security sınıfının `pageDomain` özelliğinde bulunur:

```
var domain:String = Security.pageDomain;
```

Video dosyaları için gelişmiş başlıklar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki başlıklar, FLV dosyalarıyla çalışılmasına yönelik bazı özel konuları ele alır.

Sunucuda barındırmak üzere FLV dosyalarını konfigüre etme hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

FLV dosyalarıyla çalıştığınızda, FLV dosya formatıyla çalışmak için sunucunuzu konfigüre etmeniz gerekebilir. Multipurpose Internet Mail Extensions (MIME), Internet bağlantıları üzerinden ASCII olmayan dosyalar göndermenizi sağlayan standartlaştırılmış bir veri belirtimidir. Web tarayıcıları ve e-posta istemcileri, video, ses, grafik ve formatlanmış metinler gönderip alabilmesi için birçok MIME türünü yorumlamak üzere konfigüre edilmiştir. Web sunucusundan FLV dosyaları yüklemek için, dosya uzantısını ve MIME türünü web sunucunuza kaydetmeniz gerekebilir, böylece web sunucusu belgelerinizi kontrol etmeniz gerekir. FLV dosyalarının MIME türü `video/x-flv` şeklindedir. FLV dosyası türüyle ilgili tam bilgiler şu şekildedir:

- Mime Türü: video/x-flv
- Dosya uzantısı: .flv
- Gerekli parametreler: yok
- İsteğe bağlı parametreler: yok
- Kodlama ile ilgili dikkate alınması gereken noktalar: FLV dosyaları ikili dosyalardır; bazı uygulamalar için uygulama/sekizli akış alt türünün ayarlanması gerekebilir
- Güvenlik ile ilgili sorunlar: yok
- Yayınlanan belirtim: www.adobe.com/go/video_file_format_tr

Microsoft, önceki sürümlerden Microsoft Internet Information Services (IIS) 6.0 web sunucusunda akış ortamının işlenme şeklini değiştirmiştir. Önceki IIS sürümlerinde, Flash Videosunu akışa almak için herhangi bir değişiklik gerekmez. Windows 2003 ile gelen varsayılan web sunucusu IIS 6.0'da ise, sunucu, FLV dosyalarının ortamı akışa aldığını xxxxxxx

Harici dosyaları akışa alan SWF dosyaları, Microsoft Windows Server® 2003'e yerleştirilip bir tarayıcıda görüntülendiğinde, SWF dosyası düzgün şekilde oynatılır ancak FLV videosu akışa alınmaz. Bu sorun, önceki Flash geliştirme aracı sürümleriyle ve Adobe Dreamweaver MX 2004 için Macromedia Flash Video Kit ile yaptığınız dosyalar da dahil olmak üzere, Windows Server 2003'e yerleştirilen tüm FLV dosyalarını etkiler. Bu dosyaları başka işletim sistemlerinde test ederseniz, söz konusu dosyalar düzgün şekilde çalışır.

FLV videosunu akışa almak üzere Microsoft Windows 2003 ve Microsoft IIS Server 6.0'ı konfigüre etme hakkında bilgi almak için, bkz. www.adobe.com/go/tn_19439_tr.

Macintosh'ta yerel FLV dosyalarını hedefleme hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Eğik çizgi bulunan bir yolu kullanarak Apple® Macintosh® bilgisayarda sistem dışı bir sürücüden yerel FLV oynatmayı denerseniz, video oynatılmaz. Sistem dışı sürücüler arasında, CD-ROM'lar, bölümlenmiş sabit diskler, çıkarılabilir depolama ortamları ve bağlı depolama aygıtları, vb. yer alır.

Not: *Bu arızanın nedeni, Flash Player veya AIR uygulamasındaki bir sınırlama değil, işletim sisteminin bir sınırlamasıdır.*

FLV dosyasının Macintosh'ta sistem dışı bir sürücüde oynatılması için, eğik çizgiyi esas alan notasyon (/) değil, iki noktayı esas alan notasyon (:) kullanarak mutlak bir yolla dosyayı ifade edin. Aşağıdaki listede, iki notasyon türü arasındaki fark gösterilmektedir:

- Eğik çizgiyi esas alan notasyon**:** myDrive/myFolder/myFLV.flv

- İki noktayı esas alan notasyon**:** (Mac OS®) myDrive:myFolder:myFLV.flv

Video örneği: Video Jukebox

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, bir video listesini sırayla oynatmak için dinamik olarak yükleyen basit bir video jukebox oluşturur. Bu da, kullanıcının eğitim videoları dizisi içinde gezinmesini sağlayan veya belki de kullanıcının istediği videoyu sunmadan önce hangi reklamların oynatılması gerektiğini belirten bir uygulama oluşturmanıza olanak verir. Bu örnek, ActionScript 3.0'ın şu özelliklerini gösterir:

- Video dosyasının oynatma ilerlemesini esas alarak oynatma kafasını güncelleme
- Video dosyasının meta verilerini dinleme ve ayrıştırma
- Net akışındaki belirli codec bileşenlerini işleme
- Dinamik olarak yüklenmiş bir FLV'yi yükleme, oynatma, duraklatma ve durdurma
- Net akışının meta verilerini esas alarak görüntüleme listesinde video nesnesini yeniden boyutlandırma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Video Jukebox uygulama dosyalarını Samples/VideoJukebox klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
VideoJukebox.fla veya VideoJukebox.mxml	Flex (MXML) veya Flash (FLA) için ana uygulama dosyası.
VideoJukebox.as	Uygulamanın temel işlevini sağlayan sınıf.
playlist.xml	Video jukebox'a hangi video dosyalarının yükleneceğini listeleyen bir dosya.

Harici bir video oynatma listesi dosyası yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici playlist.xml dosyası, hangi videoların yükleneceğini ve bu videoların hangi sırayla oynatılacağını belirtir. XML dosyasını yüklemek için, aşağıdaki kodda görüldüğü gibi bir URLLoader nesnesi ve bir URLRequest nesnesi kullanmanız gerekir:

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

Başka bir kod çalıştırılmadan önce dosyanın yüklenmesi için bu kod VideoJukebox sınıfının yapıcısına yerleştirilir. XML dosyasının yüklemesi bittiğinde, aşağıdaki kodda görüldüğü gibi, harici dosyayı bir XML dosyasına ayrıştıran `xmlCompleteHandler()` yöntemi çağrılır:

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

Videonun XML'i yalnızca video düğümlerini içeren bir XMLList nesnesiyken, `playlist` XML nesnesi harici dosyanın işlenmemiş XML'ini içerir. Aşağıdaki kod parçasında basit bir playlist.xml dosyası görülebilir:

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

Son olarak, `xmlCompleteHandler()` yöntemi, harici FLV dosyalarını yüklemek için kullanılan NetConnection ve NetStream nesnelerini ve görüntüleme listesinde çeşitli bileşen örneklerini ayarlayan `main()` yöntemini çağırır.

Kullanıcı arabirimini oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı arabirimini oluşturmak için, beş Button örneğini görüntüleme listesine sürüklemeniz ve şu örnek adlarıyla adlandırmanız gerekir: `playButton`, `pauseButton`, `stopButton`, `backButton` ve `forwardButton`.

Bu Button örneklerinin her biri için, aşağıdaki kod parçasında görüldüğü gibi, `click` olayına yönelik bir işleyici atamanız gerekir:

```
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);
```

`buttonClickHandler()` yöntemi, aşağıdaki kodda görüldüğü gibi hangi düğme örneğinin tıklatılmış olduğunu belirlemek için bir anahtar deyimi kullanır:

```
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;
    }
}
```

Daha sonra, görüntüleme listesine bir Slider örneği ekleyin ve bu örneğe `volumeSlider` örnek adını verin. Aşağıdaki kod, kaydırıcı örneğinin `liveDragging` özelliğini `true` değerine ayarlar ve kaydırıcı örneğinin `change` olayı için bir olay dinleyicisini tanımlar:

```
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);
```

Görüntüleme listesine bir ProgressBar örneği ekleyin ve bu örneğe `positionBar` örnek adını verin. Aşağıdaki kod parçasında görüldüğü gibi, örneğin `mode` özelliğini manüel olarak ayarlayın:

```
positionBar.mode = ProgressBarMode.MANUAL;
```

Daha sonra, görüntüleme listesine bir Label örneği ekleyin ve bu örneğe `positionLabel` örnek adını verin. Bu Label örneğinin değeri, zamanlayıcı örneğiyle ayarlanır

Video nesnesinin meta verilerini dinleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player, yüklenen her video için meta verilerle karşılaştığında, NetStream nesnesinin `client` özelliğinde `onMetaData()` geri çağrı işleyicisi çağrılır. Aşağıdaki kod bir Object öğesini başlatır ve belirtilen geri çağrı işleyicisini ayarlar:

```
client = new Object();
client.onMetaData = metadataHandler;
```

`metadataHandler()` yöntemi, kodun önceki kısımlarında tanımlanan meta özelliğine verilerini kopyalar. Bu da uygulamanın tamamı boyunca istediğiniz zaman geçerli videonun meta verilerine erişmenizi sağlar. Daha sonra, Sahne Alanı'ndaki video nesnesi, meta verilerden döndürülen boyutlarla eşleşecek şekilde yeniden boyutlandırılır. Son olarak, positionBar ilerleme çubuğu örneği taşınır ve geçerli olarak oynatılan videonun boyutu esas alınarak yeniden boyutlandırılır. Aşağıdaki kod, `metadataHandler()` yönteminin tamamını içerir:

```
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;
}
```

Dinamik olarak bir video yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Videoların her birini dinamik olarak yüklemek için, uygulama bir NetConnection ve NetStream nesnesi kullanır. Aşağıdaki kod bir NetConnection nesnesi oluşturur ve `connect()` yöntemine `null` değerini iletir. `null` değeri belirtildiğinde, Flash Player, Flash Media Server gibi bir sunucuya bağlanmak yerine, yerel sunucuda bir videoya bağlanır.

Aşağıdaki kod, NetConnection ve NetStream örneklerini oluşturur, `netStatus` olayı için bir olay dinleyicisi tanımlar ve `client` Object öğesini `client` özelliğine atar:

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

Videonun durumu her değiştiğinde, `netStatusHandler()` yöntemi çağrılır. Video durumu değişikliği kapsamında, video oynatımının başlatılması veya durdurulması, videonun arabelleğe alınması ya da video akışının bulunamaması yer alır. Aşağıdaki kod, `netStatusHandler()` olayını listeler:

```
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.
    }
}
```

Önceki kod, bilgi nesnesinin kod özelliğini değerlendirir ve kodun “NetStream.Play.Start”, “NetStream.Play.StreamNotFound” veya “NetStream.Play.Stop” öğesi mi olduğunu filtreler. Diğer tüm kodlar yoksayılır. Net akışı başlıyorsa, kod, oynatma kafasını güncelleyen Timer örneğini başlatır. Net akışı bulunamazsa veya durdurulursa, Timer örneği durdurulur ve uygulama oynatma listesindeki bir sonraki videoyu oynatmaya çalşır.

Timer her çalıştırıldığında, `positionBar` ilerleme çubuğu örneği, ProgressBar sınıfının `setProgress()` yöntemini çağırarak geçerli konumunu günceller ve `positionLabel` Label örneği, geçen süre ve geçerli videonun toplam süresiyle güncellenir.

```
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.
    }
}
```

Videonun ses düzeyini denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

NetStream nesnesinde `soundTransform` özelliğini ayarlayarak dinamik şekilde yüklenmiş videonun ses düzeyini denetleyebilirsiniz. Video jukebox uygulaması, `volumeSlider` Slider örneğinin değerini değiştirerek ses düzeyini değiştirmenize olanak sağlar. Aşağıdaki kod, NetStream nesnesinde `soundTransform` özelliğine ayarlanan SoundTransform nesnesine Slider bileşeninin değerini atayarak ses düzeyini nasış değiştirebileceğinizi gösterir:

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

Video oynatımını denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Video, video akışının sonuna ulaştığında, uygulamanın geri kalanı video oynatımını denetler ve kullanıcı önceki veya sonraki videoya atlar.

Aşağıdaki yöntem, geçerli olarak seçili dizin için XMLList öğesinden video URL'sini alır:

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

`playVideo()` yöntemi, geçerli olarak seçili videoyu yüklemek için, NetStream nesnesinde `play()` yöntemini çağırır:

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

`playPreviousVideo()` yöntemi, geçerli video dizinini azaltır, yeni video dosyasını yüklemek için `playVideo()` yöntemini çağırır ve ilerleme çubuğunu görünür olacak şekilde ayarlar:

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

Son yöntem olan `playNextVideo()` ise, video dizinini artırır ve `playVideo()` yöntemini çağırır. Geçerli video, oynatma listesindeki son videoysa, Video nesnesinde `clear()` yöntemi çağrılır ve ilerleme çubuğu örneğinin `visible` özelliği `false` değerine ayarlanır:

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

Donanım hızlandırmalı gösterim için StageVideo sınıfını kullanma

Flash Player 10.2 ve üstü

Flash Player, H.264 kod çözme işlemi için donanım hızlandırmayı kullanarak video performansını en iyi duruma getirir. StageVideo API'sini kullanarak performansı daha da geliştirebilirsiniz. Sahne alanı videosu uygulamanızın donanım hızlandırmalı gösterimden faydalanmasına izin verir.

StageVideo API'sini destekleyen çalışma zamanları şunları içerir:

- Flash Player 10.2 ve üstü

Not: *Flash Player 11.4/AIR 3.4 ve üstünde StageVideo ile kamera girdisini kullanabilirsiniz.*

İndirilebilir kaynak kodu ve sahne alanı videosu özelliğinin ek ayrıntıları Sahne Alanı Videosuna Başlarken bölümünde mevcuttur.

StageVideo hızlı başlangıç eğitimi için bkz. Sahne Alanı Videosu ile çalışma.

StageVideo öğesini kullanarak donanım hızlandırma hakkında

Video ölçeklemeyi, renk dönüştürmeyi ve ara doldurmayı içeren donanım hızlandırmalı gösterim, donanım hızlandırmalı kod çözme işleminin performans faydalarını geliştirir. GPU (donanım) hızlandırması sunan aygıtlarda videoyu doğrudan aygıt donanımında işlemek için bir flash.media.StageVideo nesnesi kullanabilirsiniz. Doğrudan işleme GPU videoyu işlerken işlemcinin diğer görevleri gerçekleştirmek için serbest kalmasını sağlar. Diğer taraftan, eski Video sınıfı genellikle yazılım gösterimini kullanır. Yazılım gösterimi işlemcide gerçekleşir ve sistem kaynaklarının önemli bir bölümünü tüketebilir.

Şu anda tam GPU hızlandırmasını birkaç aygıt desteklemektedir. Ancak, sahne alanı videosu uygulamaların mevcut olan donanım hızlandırmadan maksimum faydayı sağlamasına izin verir.

StageVideo sınıfı Video sınıfını eski yapmaz. Birlikte çalışarak bu iki sınıf belirli bir zamanda aygıt kaynaklarının izin verdiği ölçüde en iyi video görüntüleme deneyimini sunarlar. Uygulamanız doğru olayları dinleyerek ve gerektiği şekilde StageVideo ve Video sınıfları arasında geçiş yaparak donanım hızlandırmadan faydalanır.

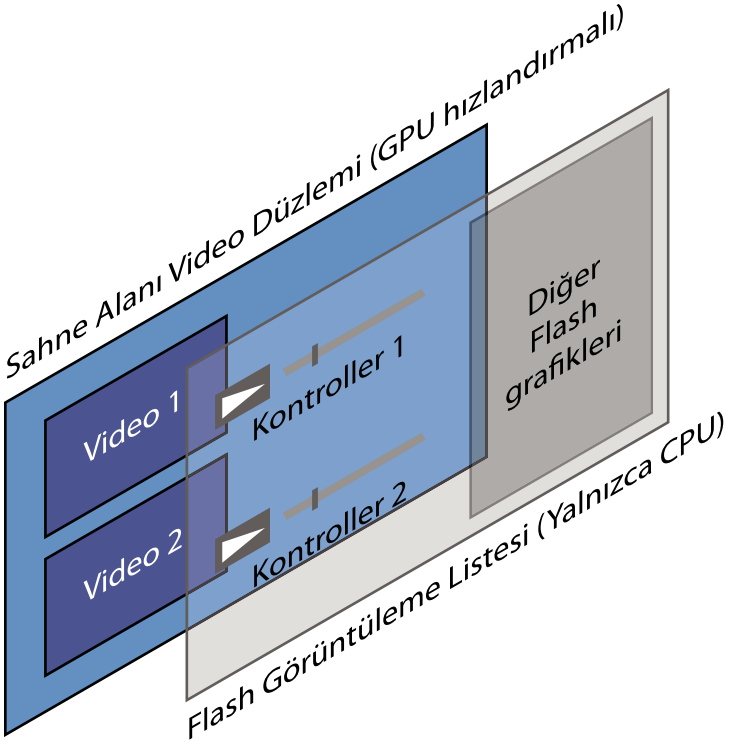
StageVideo sınıfı, video kullanımına bazı sınırlamalar getirir. StageVideo uygulamadan önce yönergeleri gözden geçirin ve uygulamanızın bunları kabul edeceğinden emin olun. Bu sınırlamaları kabul ediyorsanız, Flash Player donanım hızlandırmalı gösterimin mevcut olduğunu algıladığında StageVideo sınıfını kullanın. Bkz. "Yönergeler ve sınırlamalar" sayfa 519.

Paralel düzlemler: Sahne alanı videosu ve Flash görüntüleme listesi

Sahne alanı videosu modeli ile Flash Player videoyu görüntüleme listesinden ayırabilir. Flash Player, bileşik ekranı iki Z sıralı düzlem arasında böler:

Sahne Alanı videosu düzlemi Sahne Alanı videosu düzlemi arka planda kalır. Yalnızca donanım hızlandırmalı videoyu görüntüler. Bu tasarımdan dolayı, aygıtta donanım hızlandırma desteklenmiyorsa veya mevcut değilse bu düzlem de kullanılamaz. ActionScript'te, StageVideo nesneleri sahne alanı videosu düzleminde oynatılan videoları işler.

Flash görüntüleme listesi düzlemi Flash görüntüleme listesi varlıkları sahne alanı videosu düzleminin önünde bulunan bir düzlemde birleştirilir. Görüntüleme listesi varlıkları oynatma kontrolleri de dahil olmak üzere çalışma zamanının oluşturduğu her şeyi içerir. Donanım hızlandırma mevcut olmadığında videolar yalnızca Video sınıfı nesnesini kullanarak bu düzlem üzerinde oynatılabilir. Sahne alanı videosu her zaman Flash görüntüleme listesi grafiklerinin arkasında görüntülenir.



Video ekranı düzlemleri

StageVideo nesnesi ekranın döndürülmemiş, pencereye göre hizalanmış dikdörtgen bir alanında görüntülenir. Sahne alanı videosu düzleminin arkasında nesneleri katmanlar halinde yerleştiremezsiniz. Ancak, sahne alanı videosu düzleminin üzerine diğer grafikleri katman olarak yerleştirmek için Flash görüntüleme listesi düzlemini kullanabilirsiniz. Sahne alanı videosu görüntüleme listesiyle eş zamanlı çalışır. Bu nedenle iki gizli düzlemi kullanan birleştirilmiş bir görsel efekt oluşturmak için iki mekanizmayı birlikte kullanabilirsiniz. Örneğin, ön düzlemi arka planda çalışan sahne alanı videosu üzerinde işleyen oynatma kontrolleri için kullanabilirsiniz.

Sahne alanı videosu ve H.264 codec'i

Flash Player uygulamalarında video hızlandırmayı uygulama işlemi iki adımdan oluşur:

1 Videoyu H.264 olarak kodlama

2 StageVideo API'sini uygulama

En iyi sonuçları elde etmek için her iki adımı da gerçekleştirin. H.264 codec'i, video kodu çözme işleminden gösterime kadar, donanım hızlandırmadan maksimum faydayı sağlamanıza izin verir.

Sahne alanı videosu GPU'dan İşlemciye yapılan geri okumayı ortadan kaldırır. Başka bir deyişle, GPU kodu çözülmüş kareleri görüntüleme listesi nesneleriyle birleştirmek için işlemciye geri göndermez. Bunun yerine GPU kod çözme işleminden geçirilmiş ve oluşturulmuş kareleri doğrudan sahnede, görüntüleme listesi nesnelerinin arkasında doldurarak oluşturur. Bu teknik, işlemci ve bellek kullanımını azaltır ve piksellerin aslına daha uygun olmasını sağlar.

Daha fazla Yardım konusu

"Video biçimlerini anlama" sayfa 479

Yönergeler ve sınırlamalar

Videoyu tam ekran modunda çalışırken, aygıt donanım hızlandırmayı destekliyorsa sahne alanı videosu her zaman kullanılabilir durumdadır. Ancak, Flash Player tarayıcı içinde de çalışır. Tarayıcı bağlamında sahne alanı videosu kullanılabilirliğini `wmode` ayarı etkiler. Sahne alanı videosunu kullanmak istiyorsanız her zaman `wmode="direct"` öğesini kullanmayı deneyin. Sahne alanı videosu tam ekran modunda değilken diğer `wmode` ayarları ile uyumlu değildir. Bu kısıtlama çalışma zamanında sahne alanının önceden tahmin edilemeyecek bir şekilde kullanılabilir ve kullanılamaz durumları arasında kalabileceği anlamına gelir. Örneğin, kullanıcı sahne alanı videosu çalışırken tam ekran modundan çıkarsa video bağlamı tarayıcıya geri döner. Tarayıcının `wmode` parametresi `"direct"` olarak ayarlanmazsa sahne alanı videosu aniden kullanılamaz duruma gelebilir. Flash Player oynatma bağlamı değişikliklerini uygulamalara olay kümeleri aracılığıyla iletir. StageVideo API'si uygularsanız, sahne alanı videosu kullanılamaz duruma geldiğinde yedek olması için bir Video nesnesi tutun.

Donanımla doğrudan bir ilişkisi olduğundan sahne alanı videosu bazı video özelliklerini kısıtlar. Sahne alanı videosu şu kısıtlamaları uygular:

- Her SWF dosyası için Flash Player aynı anda video görüntüleyebilecek StageVideo nesnesi sayısını dört ile sınırlar. Ancak, aygıtın donanım kaynaklarına bağlı olarak gerçek sınır daha düşük olabilir.
- Video zamanlaması, çalışma zamanının görüntülediği içeriğin zamanlamasıyla senkronize değildir.
- Video görüntüleme alanı yalnızca bir dikdörtgen olabilir. Oval veya düzensiz şekiller gibi daha gelişmiş görüntüleme alanları kullanamazsınız.
- Videoyu döndüremezsiniz.
- Videoyu bitmap önbelleğine alamazsınız veya videoya erişmek için BitmapData nesnesini kullanamazsınız.
- Videoya filtre uygulayamazsınız.
- Videoya renk dönüştürmeleri uygulayamazsınız.
- Videoya bir alfa değeri uygulayamazsınız.
- Görüntüleme listesi düzleminde nesnelere uyguladığınız harmanlama modları, sahne alanı videosuna uygulanmaz.
- Videoyu yalnızca tam piksel sınırları üzerine yerleştirebilirsiniz.
- GPU görüntü oluşturma belirli aygıt için mevcut olanın en iyisi olsa da tüm aygıtlarda %100 "aynı piksellere" sahip değildir. Sürücü ve platform farklılıklarından dolayı ufak farklılıklar görülür.

- Bazı aygıtlar gerekli tüm renk alanlarını desteklemez. Örneğin, bazı aygıtlar H.264 standardı olan BT.709'u desteklemez. Bu tür durumlarda hızlı görüntüleme için BT.601'i kullanabilirsiniz.
- Sahne alanı videosunu normal, opak veya saydam gibi WMODE ayarlarıyla kullanamazsınız. Sahne alanı videosu tam ekran modunda değilken yalnızca `WMODE=direct` öğesini destekler. WMODE, Safari 4 veya üstü ve IE 9 veya üst sürümlerinde etkili değildir.

Birçok durumda bu sınırlamalar video oynatıcı uygulamalarını etkilemez. Bu sınırlamaları kabul edebiliyorsanız mümkün olduğunda sahne alanı videosunu kullanın.

Daha fazla Yardım konusu

"Tam ekran modunda çalışma" sayfa 169

StageVideo API'lerini kullanma

Sahne alanı videosu çalışma zamanı içindeki video oynatmayı ve aygıt performansını geliştiren bir mekanizmadır. Çalışma zamanı bu mekanizmayı oluşturur ve sürdürür. Geliştirici olarak sizin rolünüz bundan faydalanmak üzere uygulamanızı konfigüre etmektir.

Sahne alanı videosunu kullanmak için sahne alanı videosunun kullanılabilir olduğu ve kullanılamaz olduğu zamanları belirleyen olay işleyicileri çerçevesi uygularsınız. Sahne alanı videosunun kullanılabilir olduğu bildirimini aldığınızda `Stage.stageVideos` özelliğinden bir StageVideo nesnesi alırsınız. Çalışma zamanı bu Vector nesnesini bir veya daha fazla StageVideo nesnesiyle doldurur. Bundan sonra akış halindeki videoları görüntülemek için Video nesnesi yerine sağlanan StageVideo nesnelerinden birini kullanabilirsiniz.

Flash Player'da, sahne alanı videosunun artık kullanılabilir durumda olmadığına ilişkin bir bildirim aldığınızda video akışınızı Video nesnesine geri döndürün.

Not: *StageVideo nesneleri oluşturamazsınız.*

Stage.stageVideos özelliği

`Stage.stageVideos` özelliği size StageVideo örneklerine erişim sağlayan bir Vector nesnesidir. Bu vektör donanım ve sistem kaynaklarına bağlı olarak dört adete kadar StageVideo nesnesi içerebilir. Mobil aygıtlarda bir tane bulunabilir veya hiç bulunmayabilir.

Sahne alanı videosu kullanılabilir olmadığında bu vektör herhangi bir nesne içermez. Çalışma zamanı hatalarından kaçınmak için bu vektörün üyelerine yalnızca sahne alanı videosunun kullanılabilir olduğunu belirten en güncel `StageVideoAvailability` olayını aldığınızda eriştiğinizden emin olun.

StageVideo olayları

StageVideo API çerçevesi aşağıdaki olayları sağlar:

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY `Stage.stageVideos` özelliği değiştiğinde gönderilir. `StageVideoAvailabilityEvent.availability` özelliği `AVAILABLE` veya `UNAVAILABLE` öğesini gösterir. Bu olayı `stageVideos` özelliğinin doğrudan `Stage.stageVideos` vektörünün uzunluğunu kontrol etmek yerine herhangi bir StageVideo nesnesi içerip içermeyeceğini belirlemek için kullanın.

StageVideoEvent.RENDER_STATE Bir NetStream veya Camera nesnesi bir StageVideo nesnesine eklendiğinde ve oynatıldığında gönderilir. Geçerli olarak kullanımda olan kod çözme türünü gösterir: donanım, yazılım veya yok (hiçbir şey görüntülenmez). Olay hedefi video görünüm penceresini yeniden boyutlandırmak üzere kullanmak için güvenli olan `videoWidth` ve `videoHeight` özelliklerini içerir.

Önemli: *StageVideo hedef nesnesinden alınan koordinatlar, standart görüntüleme listesinin bir parçası olmadıklarından Sahne Alanı koordinatlarını kullanır.*

VideoEvent.RENDER_STATE Bir Video nesnesi kullanılırken gönderilir. Yazılım veya donanım hızlandırmalı kod çözmenin kullanımda olup olmadığını belirtir. Bu olay donanım hızlandırmalı kod çözmeyi gösteriyorsa, mümkünse StageVideo nesnesine geçin. Video olay hedefi video görünüm penceresini yeniden boyutlandırmak üzere kullanmak için güvenli olan `videoWidth` ve `videoHeight` özelliklerini içerir.

StageVideo özelliğini uygulamak için iş akışı

StageVideo özelliğini uygulamak için şu üst düzey adımları izleyin:

1. `Stage.stageVideos` vektörünün ne zaman değiştiğini öğrenmek için `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` olayını dinleyin. Bkz. "StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY olayını kullanma" sayfa 522.
2. `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` olayı sahne alanı videosunun kullanılabilir olduğunu bildirirse, StageVideo nesnesine erişmek için olay işleyicisinin içindeki `Stage.stageVideos` Vector nesnesini kullanın.
3. `StageVideo.attachNetStream()` kullanarak bir NetStream nesnesi veya `StageVideo.attachCamera()` kullanarak bir Camera nesnesi ekleyin.
4. `NetStream.play()` öğesini kullanarak videoyu oynatın.
5. Video oynatma işleminin durumunu belirlemek için StageVideo nesnesi üzerinde `StageVideoEvent.RENDER_STATE` olayını dinleyin. Bu olayın alınması, videonun genişlik ve yükseklik özelliklerinin başlatıldığını veya değiştirildiğini de belirtir. Bkz. "StageVideoEvent.RENDER_STATE ve VideoEvent.RENDER_STATE olaylarını kullanma" sayfa 524.
6. Video nesnesi üzerinde `VideoEvent.RENDER_STATE` olayını dinleyin. Bu olay, `StageVideoEvent.RENDER_STATE` ile aynı durumları sağlar, bu nedenle GPU hızlandırmasının kullanılabilir olup olmadığını belirlemek için de bunu kullanabilirsiniz. Bu olayın alınması, videonun genişlik ve yükseklik özelliklerinin başlatıldığını veya değiştirildiğini de belirtir. Bkz. "StageVideoEvent.RENDER_STATE ve VideoEvent.RENDER_STATE olaylarını kullanma" sayfa 524.

StageVideo olay dinleyicilerini başlatma

Uygulama başlatılırken StageVideoAvailabilityEvent ve VideoEvent dinleyicilerinizi ayarlayın. Örneğin, bu dinleyicileri `flash.events.Event.ADDED_TO_STAGE` olay işleyicisinde başlatabilirsiniz. Bu olay uygulamanızın sahne alanında görünür olacağını garantiler:

```
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);
        //...
    }
```

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY olayını kullanma

`StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` işleyicisinde StageVideo'nun kullanılabilirlik durumuna göre Video nesnesi mi yoksa StageVideo nesnesi mi kullanacağınıza karar verin. `StageVideoAvailabilityEvent.availability` özelliği `StageVideoAvailability.AVAILABLE` olarak ayarlıysa StageVideo nesnesini kullanın. Bu durumda, Stage.stageVideos vektörünün bir veya daha fazla StageVideo nesnesi içereceğine güvenebilirsiniz. `Stage.stageVideos` özelliğinden bir StageVideo nesnesi alın ve NetStream nesnesini buna ekleyin. StageVideo nesneleri her zaman arka planda görüntülendiğinden, varolan Video nesnelerini (her zaman ön plandadır) kaldırın. Ayrıca bu olay işleyicisini `StageVideoEvent.RENDER_STATE` olayı için bir dinleyici eklemek üzere kullanabilirsiniz.

`StageVideoAvailabilityEvent.availability` özelliği `StageVideoAvailability.UNAVAILABLE` olarak ayarlıysa StageVideo nesnesini kullanmayın veya `Stage.stageVideos` vektörüne erişmeyin. Bu durumda, NetStream nesnesini bir Video nesnesi ekleyin. Son olarak, StageVideo veya Video nesnesini sahne alanına ekleyin ve `NetStream.play()` öğesini çağırın.

Aşağıdaki kod `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` olayının nasıl işleneceğini gösterir:

```
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);
    }
}
```

Önemli: *Uygulama Stage.stageVideos[0] öğesinde vektör öğesine ilk eriştiğinde, varsayılan dikdörtgen 0,0,0,0 olarak ayarlıdır ve kaydırma ve yakınlaştırma özellikleri varsayılan değerleri kullanır. Bu değerleri her zaman tercih ettiğiniz ayarlara sıfırlayın. Video görünüm penceresinin boyutlarını hesaplamak için* `StageVideoEvent.RENDER_STATE` *veya* `VideoEvent.RENDER_STATE` *öğesinin* `videoWidth` *ve* `videoHeight` *özelliklerini kullanabilirsiniz.*

Sahne Alanı Videosuna Başlarken bağlantısından bu örnek uygulamanın tam kaynak kodunu indirin.

StageVideoEvent.RENDER_STATE ve VideoEvent.RENDER_STATE olaylarını kullanma

StageVideo ve Video nesneleri görüntüleme ortamı değiştiğinde uygulamaları bilgilendiren olaylar gönderir. Bu olaylar `StageVideoEvent.RENDER_STATE` ve `VideoEvent.RENDER_STATE` olaylarıdır.

Bir StageVideo veya Video nesnesi NetStream nesnesi eklendiğinde ve oynatılmaya başladığında oluşturma durumu olayı gönderir. Bu olayı ayrıca görüntüleme ortamı değiştiğinde gönderir. Örneğin, video görünüm penceresi yeniden boyutlandırıldığında. Bu bildirimleri, görünüm pencerenizi olay hedef nesnesinin geçerli `videoHeight` ve `videoWidth` değerlerine sıfırlamak için kullanın.

Bildirilen oluşturma durumları şunları içerir:

- `RENDER_STATUS_UNAVAILABLE`
- `RENDER_STATUS_SOFTWARE`
- `RENDER_STATUS_ACCELERATED`

Oluşturma durumları, hangi sınıfın o anda videoyu oynattığına bakılmaksızın, donanım hızlandırmalı kod çözme işleminin ne zaman kullanıldığını gösterir. Gerekli kod çözme işleminin mevcut olup olmadığını kontrol etmek için `StageVideoEvent.status` özelliğini kullanın. Özellik "unavailable" olarak ayarlanmışsa, StageVideo nesnesi videoyu oynatamaz. Bu durum NetStream nesnesini hemen bir Video nesnesine yeniden eklemenizi gerektirir. Diğer durumlar uygulamanıza geçerli oluşturma koşullarıyla ilgili bilgi verir.

Aşağıdaki tablo Flash Player'daki StageVideoEvent ve VideoEvent nesneleri için tüm oluşturma durumu değerlerinin etkilerini açıklar:

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	Kod çözme ve gösterim donanımda gerçekleşir. (En iyi performans.)	Donanımda gösterim, yazılımda kod çözme. (Kabul edilebilir performans.)	Videoyu işlemek için mevcut GPU kaynağı yok ve hiçbir şey görüntülenmiyor. **Video nesnesine geri döner.**
VideoEvent	Yazılımda gösterim, donanımda kod çözme. (Yalnızca modern bir masaüstü sisteminde kabul edilebilir performans. Düşük düzeyli tam ekran performansı.)	Yazılımda gösterim, donanımda gösterim. (En kötü performans durumu. Düşük düzeyli tam ekran performansı.)	(Yok)

Renk alanları

Sahne alanı videosu renk alanlarını desteklemek için temelde bulunan donanım özelliklerini kullanır. SWF içeriği tercih ettiği renk alanını gösteren bir meta veri sağlayabilir. Ancak, renk alanının kullanılıp kullanılamayacağını aygıtın grafik donanımı belirler. Bir aygıt birkaç renk alanını desteklerken başka bir aygıt hiçbirini desteklemeyebilir. Donanım istenilen renk alanını desteklemiyorsa, Flash Player desteklenen renk alanları arasından en yakın eşleşmeyi bulmaya çalışır.

Donanımın hangi renk alanlarını desteklediğini sorgulamak için `StageVideo.colorSpaces` özelliğini kullanın. Bu özellik bir String vektöründe desteklenen renk alanlarının listesini döndürür:

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

O anda oynatılan videonun hangi renk alanını kullandığını öğrenmek için `StageVideoEvent.colorSpace` özelliğini kontrol edin. Bu özelliği `StageVideoEvent.RENDER_STATE` olayı için olay işleyicinizde kontrol edin:

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Flash Player desteklenmeyen bir renk alanı için yerine geçecek bir alan bulamazsa, sahne alanı videosu varsayılan renk alanı BT.601'i kullanır. Örneğin, H.264 kodlamalı video akışları genellikle BT.601 renk alanını kullanır. Aygıt donanımı BT.709'u desteklemiyorsa `colorSpace` özelliği `"BT601"` öğesini döndürür. `"unknown"` öğesinin `StageVideoEvent.colorSpace` değeri donanımın renk alanını sorgulamak için araç sağlamadığını gösterir.

Uygulamanız geçerli renk alanının kabul edilemez olduğunu varsayarsa, StageVideo nesnesinden Video nesnesine geçiş yapmayı seçebilirsiniz. Video sınıfı yazılım birleştirme aracılığıyla tüm renk alanlarını destekler.

Bölüm 26: Kameralar ile çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcının bilgisayarına takılı bir kamera, ActionScript kullanarak görüntüleyip işleyebileceğiniz bir video verisi kaynağı görevi görebilir. Camera sınıfı, bir bilgisayar veya aygıt kamerasıyla çalışmak için kullanabileceğiniz, ActionScript'te tümleşik olan mekanizmadır.

Mobil aygıtlarda CameraUI sınıfını da kullanabilirsiniz. CameraUI sınıfı, kullanıcının durağan görüntü veya video yakalamasına izin vermek için ayrı bir kamera uygulaması başlatır. Kullanıcı işlemi tamamladığında, uygulamanız bir MediaPromise nesnesi aracılığıyla görüntüye veya videoya erişebilir.

Daha fazla Yardım konusu

Christian Cantrell: How to use CameraUI in a Cross-platform Way (CameraUI öğesi platformlar arası bir şekilde nasıl kullanılır?)

Michaël CHAIZE: Android, AIR and the Camera (Android, AIR ve Kamera)

Camera sınıfını anlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Camera nesnesi, kullanıcının yerel kamerasına bağlanmanıza ve yerel olarak (geri kullanıcıya) veya uzaktan bir sunucuya (örn. Flash Media Server) videoyu yayınlamanıza olanak sağlar.

Camera sınıfını kullanarak, kullanıcının kamerası hakkında şu tür bilgilere erişebilirsiniz:

- Kullanıcının bilgisayarında takılı olan hangi kameraların veya aygıtların kullanılabilir olduğu
- Bir kameranın takılmış olup olmadığı
- Flash Player uygulamasının kullanıcının kamerasına erişime izin verdiği veya erişimi reddettiği
- Hangi kameranın geçerli olarak etkin olduğu
- Yakalanan videonun genişliği ve yüksekliği

Camera sınıfı, kamera nesneleriyle çalışılmasına yönelik çok sayıda yararlı yöntem ve özellik içerir. Örneğin, statik `Camera.names` özelliği, kullanıcının bilgisayarında geçerli olarak takılı olan kamera adlarının dizisini içerir. Geçerli olarak etkin olan kameranın adını görüntülemek için `name` özelliğini de kullanabilirsiniz.

Not: *Ağ içinde kamera videosu akışı gerçekleştirirken, her zaman ağ kesintileriyle ilgilenmeniz gerekir. Ağ kesintileri özellikle mobil aygıtlarda çeşitli nedenlerden dolayı meydana gelebilir.*

Ekranda kamera içeriğini görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir kameraya bağlanılması, video yüklemek için NetConnection ve NetStream sınıflarının kullanılmasından daha az kod gerektirebilir. Flash Player ile bir kullanıcının kamerasına erişmek için öncelikle kullanıcıdan kameraya bağlanma izni almanız gerektiğinden, kamera sınıfı hemen incelikli de olabilir.

Şu kod, kullanıcının yerel kamerasına bağlanmak için, Camera sınıfını nasıl kullanabileceğinizi gösterir:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Not: *Camera sınıfı bir yapıcı yöntemine sahip değildir. Yeni bir Camera örneği oluşturmak için, statik* `Camera.getCamera()` *yöntemini kullanırsınız.*

Kamera uygulamanızı tasarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir kullanıcının kamerasına bağlanan uygulama yazarken kodunuzda şunları açıklamanız gerekir:

- Kullanıcının geçerli olarak takılı bir kameraya sahip olup olmadığını kontrol edin. Durumu kameranın mevcut olmadığı durumlarda işleyin.
- Yalnızca Flash Player için, kullanıcının açıkça kameraya erişilmesine izin verip vermediğini kontrol edin. Güvenlik nedenleriyle oynatıcı, kullanıcının kamerasına erişilmesine izin vermesine veya erişilmesini reddetmesini sağlayan Flash Player Ayarları iletişim kutusunu görüntüler. Bu da Flash Player uygulamasının bir kullanıcının kamerasına bağlanıp kullanıcının izni olmadan bir video akışını yayınlamasını önler. Kullanıcı izin verme seçeneğini tıklatırsa, uygulamanız kullanıcının kamerasına bağlanabilir. Kullanıcı reddetme seçeneğini tıklatırsa, uygulamanız kullanıcının kamerasına erişemez. Uygulamalarınızın her zaman her iki durumu da düzgün biçimde işlemesi gerekir.
- Yalnızca AIR için, Camera sınıfının uygulamanız tarafından desteklenen aygıt profilleri tarafından desteklenip desteklenmediğini kontrol edin.
- Camera sınıfı mobil tarayıcılarda desteklenmez.
- Camera sınıfı GPU oluşturma modunu kullanan mobil AIR uygulamalarında desteklenmez.
- Mobil aygıtlarda aynı anda yalnızca bir kamera etkin olabilir.

Daha fazla Yardım konusu

Christophe Coenraets: Multi-User Video Tic-Tac-Toe

Mark Doherty: Android Radar app (source)

Bir kullanıcının kamerasına bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir kullanıcının kamerasına bağlanırken birinci adım, Camera türünde bir değişken oluşturup bu değişkeni statik `Camera.getCamera()` yönteminin döndürme değerine başlatarak yeni bir kamera örneği oluşturulmasıdır.

Sonraki adım ise, yeni bir video nesnesi oluşturulup buna Camera nesnesinin eklenmesidir.

Üçüncü adım, görüntüleme listesine video nesnesinin eklenmesidir. Camera sınıfı DisplayObject sınıfını genişletmeyip söz konusu sınıf doğrudan görüntüleme listesine eklenemediğinden, adım 2 ve 3'ü uygulamanız gerekir. Kameranın yakalanan videosunu görüntülemek için yeni bir video nesnesi oluşturup `attachCamera()` yöntemini çağırırsınız.

Aşağıdaki kod bu üç adımı gösterir:

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

Bir kullanıcının kamerası takılı değilse, uygulamanın herhangi bir şey görüntülemeyeceğini unutmayın.

Gerçek yaşamda, uygulamanız için ek adımlar uygulamanız gerekir. Daha fazla bilgi için, bkz. "Kameraların takılı olduğunu doğrulama" sayfa 528 ve "Kamera erişimi için izinleri algılama" sayfa 529.

Daha fazla Yardım konusu

Lee Brimelow: How to access the camera on Android devices

Kameraların takılı olduğunu doğrulama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir kamera örneğinde herhangi bir yöntemi veya özelliği kullanmaya çalışmadan önce, kullanıcının bir kamera taktığını doğrulamak istersiniz. Kullanıcının takılı bir kameraya sahip olup olmadığını kontrol etmenin iki yolu vardır:

- Kullanılabilir kamera adlarının bir dizisini içeren statik `Camera.names` özelliğini kontrol edin. Çoğu kullanıcı aynı anda birden çok kamera takmayacağından, genellikle bu dizi bir veya daha az dize içerir. Şu kod, kullanıcının kullanılabilir kameraya sahip olup olmadığını görmek için `Camera.names` özelliğini nasıl kontrol edebileceğinizi gösterir:

  ```
  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.");
  }
  ```

- Statik `Camera.getCamera()` yönteminin döndürme değerini kontrol edin. Herhangi bir kamera kullanılabilir veya takılı değilse, bu yöntem `null` değerini, aksi takdirde Camera nesnesinin bir başvurusunu döndürür. Şu kod, kullanıcının kullanılabilir kameraya sahip olup olmadığını görmek için `Camera.getCamera()` yöntemini nasıl kontrol edebileceğinizi gösterir:

```
var cam:Camera = Camera.getCamera();
if (cam == null)
{
    trace("User has no cameras installed.");
}
else
{
    trace("User has at least 1 camera installed.");
}
```

Camera sınıfı, DisplayObject sınıfını genişletmediğinden, `addChild()` yöntemi kullanılarak doğrudan görüntüleme listesine eklenemez. Kameranın yakalanan videosunu görüntülemek için yeni bir Video nesnesi oluşturmanız ve Video örneğinde `attachCamera()` yöntemini çağırmanız gerekir.

Bu kod parçası, varsa, kamerayı nasıl ekleyebileceğinizi gösterir; kamera yoksa, uygulama herhangi bir şey görüntülemez:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

Mobil aygıt kameraları

Camera sınıfı mobil tarayıcılarındaki Flash Player çalışma zamanında desteklenmez.

Mobil aygıtlardaki AIR uygulamalarında, aygıttaki kameraya veya kameralara erişebilirsiniz. Mobil aygıtlarda hem ön tarafa hem de arka tarafa bakan kamerayı kullanabilirsiniz. Ancak, aynı anda yalnızca bir kamera çıktısı görüntülenebilir. (İkinci bir kamera takmak ilk kameranın bağlantısını keser.) Ön kamera, iOS'de yatay olarak yansıtılmıştır, Android'de yatay olarak yansıtılmamıştır.

Kamera erişimi için izinleri algılama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

AIR uygulama sanal alanında, uygulama kullanıcının izni olmadan herhangi bir kameraya erişebilir. Ancak, Android'de uygulama uygulama açıklayıcısında Android CAMERA iznini belirtmelidir.

Flash Player uygulamasının kamera çıktısını görüntüleyebilmesi için öncelikle kullanıcının açıkça Flash Player'a kameraya erişme izni vermesi gerekir. `attachCamera()` yöntemi çağrıldığında, Flash Player uygulaması, Flash Player'ın kameraya veya mikrofona erişmesine izin verip vermeyeceğini kullanıcıya soran Flash Player Ayarları iletişim kutusunu görüntüler. Kullanıcı İzin Ver düğmesini tıklatırsa, Flash Player uygulaması Sahne Alanı'nda Video örneğinde kameranın çıktısını görüntüler. Kullanıcı Reddet düğmesini tıklatırsa, Flash Player kameraya bağlanamaz ve Video nesnesi herhangi bir şey görüntülemez.

Kullanıcının, Flash Player uygulamasının kameraya erişmesine izin verip vermediğini algılamak istiyorsanız, aşağıdaki kodda görüldüğü gibi kameranın `status` olayını (`StatusEvent.STATUS`) dinleyebilirsiniz:

```
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"
}
```

Kullanıcı İzin Ver veya Reddet öğesini tıklattığı anda `statusHandler()` işlevi çağrılır. İki yöntemden birini kullanarak, kullanıcının hangi düğmeyi tıklattığını algılayabilirsiniz:

- `statusHandler()` işlevinin `event` parametresi, “Camera.Muted” veya “Camera.Unmuted” dizesini içeren bir kod özelliği içerir. Değer “Camera.Muted” ise, kullanıcı Reddet düğmesini tıklatmış olduğundan Flash Player kameraya erişemiyordur. Aşağıdaki kod parçasında buna bir örnek görebilirsiniz:

  ```
  function statusHandler(event:StatusEvent):void
  {
      switch (event.code)
      {
          case "Camera.Muted":
              trace("User clicked Deny.");
              break;
          case "Camera.Unmuted":
              trace("User clicked Accept.");
              break;
      }
  }
  ```

- Camera sınıfı, kullanıcının Flash Player Gizlilik panelinde kameraya erişilmesini reddettiğini (`true`) veya erişilmesine izin verdiğini (`false`) belirten `muted` adında salt okunur bir özellik içerir. Aşağıdaki kod parçasında buna bir örnek görebilirsiniz:

  ```
  function statusHandler(event:StatusEvent):void
  {
      if (cam.muted)
      {
          trace("User clicked Deny.");
      }
      else
      {
          trace("User clicked Accept.");
      }
  }
  ```

Gönderilecek durum olayını kontrol ederek, kameraya erişilmesini kabul eden veya reddeden kullanıcıyı işleyen kod yazabilir ve uygun şekilde kamerayı silebilirsiniz. Örneğin, kullanıcı Reddet düğmesini tıklatırsa, kullanıcıya video sohbetine katılmak istiyorsa İzin Ver'i tıklatması gerektiğini belirten bir mesaj görüntüleyebilir veya sistem kaynaklarında yer açmak için görüntüleme listesinde Video nesnesinin silindiğinden emin olabilirsiniz.

AIR'de, bir Camera nesnesi kamera kullanma izni dinamik olmadığı için durum olayları göndermez.

Kameranın video kalitesini en üst düzeye çıkarma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varsayılan olarak, yeni Video sınıfı örnekleri 320 piksel genişlikte ve 240 piksel yüksekliktedir. Video kalitesini en üst düzeye çıkarmak için, her zaman video nesnenizin kamera nesnesiyle döndürülen videoyla aynı boyutlarda olduğundan emin olmanız gerekir. Aşağıdaki kod parçasında görüldüğü gibi, Camera sınıfının `width` ve `height` özelliklerini kullanarak kamera nesnesinin genişlik ve yüksekliğini alabilir, video nesnesinin `width` ve `height` özelliklerini kamera nesnelerinin boyutlarıyla eşleşecek şekilde ayarlayabilir veya kameranın genişlik ve yüksekliğini Video sınıfının yapıcı yöntemine iletebilirsiniz:

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

`getCamera()` yöntemi, kamera nesnesinin başvurusunu (veya kullanılabilir kamera yoksa `null` değerini) döndürdüğünden, kullanıcı kamerasına erişilmesini reddetse de kameranın yöntemlerine ve özelliklerine erişebilirsiniz. Böylece kameranın yerel yükseklik ve genişliğini kullanarak video örneğinin boyutunu ayarlayabilirsiniz.

```
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);
}
```

Tam ekran modu hakkında bilgi almak için, “Sahne Alanı özelliklerini ayarlama” sayfa 167 bölümündeki tam ekran modu konusuna bakın.

Kamera durumunu izleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kamera sınıfı, Camera nesnesinin geçerli durumunu izlemenize olanak sağlayan birçok özellik içerir. Örneğin, şu kod, bir Timer nesnesini ve görüntüleme listesindeki metin alanı örneğini kullanarak çok sayıda kamera özelliğini görüntüler:

```
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");
}
```

Saniyenin her 1/10'unda (100 milisaniyede) Timer nesnesinin `timer` olayı gönderilir ve `timerHandler()` işlevi görüntüleme listesinde metin alanını günceller.

Bölüm 27: Dijital haklar yönetimini kullanma

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

Adobe® Access™ bir içerik koruma çözümüdür. Değerli içeriğe kesintisiz erişim sağlayarak içerik sahiplerinin, dağıtıcıların ve reklamcıların yeni gelir kaynakları keşfetmesine imkan verir. Yayıncılar içeriği şifrelemek, ilke oluşturmak ve lisans vermek için Adobe Access'i kullanır. Adobe Flash Player ve Adobe AIR dahilinde, Adobe Access modülü olan bir DRM kütüphanesi bulunur. Bu modül, çalışma zamanının Adobe Access lisans sunucusu ve oynatma korumalı içerikle iletişim kurmasını sağlar. Böylece, çalışma zamanı Adobe Access tarafından korunup, Flash Media Server tarafından dağıtılan içeriğin yaşam döngüsünü tamamlar.

Adobe Access ile, içerik yayıncıları hem ücretsiz hem de premium içerik sağlayabilir. Örneğin, bir tüketici bir televizyon programını reklamsız izlemek istiyor. Tüketici, içerik yayıncısına kaydolur ve ödeme yapar. Bundan sonra erişim sağlamak ve programı reklamsız oynatmak için kullanıcı bilgilerini girebilir.

Başka bir örnekte, bir tüketici seyahat sırasında İnternet erişimi yokken içeriği çevrimdışı olarak görüntülemek istiyor. Bu çevrimdışı iş akışı AIR uygulamalarında desteklenmektedir. Kullanıcı, yayıncıya kaydolup içerik için ödeme yaptıktan sonra yayıncının web sitesine erişip içeriği ve ilgili AIR uygulamasını buradan indirebilir. AIR uygulamasını kullanarak, izin verilen süre içerisinde içeriği çevrimdışı olarak görüntüleyebilir. İçerik, etki alanlarını kullanan aynı aygıt grubundaki diğer aygıtlarla da paylaşılabilir (**3.0'da yenidir**).

Adobe Access, ayrıca kullanıcı denetimi gerektirmeyen anonim erişimi de destekler. Örneğin, bir yayıncı reklamla desteklenen içeriği dağıtmak için anonim erişim kullanabilir. Ayrıca anonim erişim, yayıncıya geçerli içeriğe belirli sayıda gün boyunca ücretsiz olarak erişme izni verir. Ayrıca, içerik sağlayıcı, içeriği için gerekli olan oynatıcının türünü ve sürümünü belirtebilir ve sınırlayabilir.

Bir kullanıcı Flash Player veya Adobe AIR'deki korumalı içeriği oynatmak istediğinde, uygulamanız DRM API'lerini çağırmalıdır. DRM API'leri korumalı içeriğin oynatılması için iş akışını başlatır. Çalışma zamanı, Adobe Access modülü üzerinden lisans sunucusuyla iletişim kurar. Lisans sunucusu, gerekiyorsa kullanıcıyı doğrular ve korumalı içeriğin oynatılmasına izin verecek bir lisans yayınlar. Çalışma zamanı lisansı alır ve içeriği oynatmak üzere şifresini çözer.

Adobe Access ile korunmakta olan içeriği oynatmak için uygulamanızı nasıl etkinleştireceğiniz burada anlatılmaktadır. Adobe Access kullanarak içeriğin nasıl şifreleneceğini veya ilkelerin korunacağını anlamak gerekli değildir. Ancak, kullanıcıyı doğrulamak ve lisans almak için Adobe Access lisans sunucusu ile iletişim kurduğunuz varsayılır. Ayrıca, özellikle Adobe Access tarafından korunan içeriği oynatmak için bir uygulama tasarladığınız varsayılmaktadır.

Oluşturma ilkeleri de dahil Adobe Access uygulamasına genel bakış için Adobe Access paketine dahil edilen belgelere göz atın.

Daha fazla Yardım konusu

flash.net.drm paketi

flash.net.NetConnection

flash.net.NetStream

Korumalı içerik iş akışını anlama

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Önemli: Flash Player 11.5 ve üzeri, Adobe Access modülü ile entegre olur, böylece güncelleme adımına (`SystemUpdater.update(SystemUpdaterType.DRM)` öğesinin çağrılmasına) gerek kalmaz. Bu aşağıdaki tarayıcıları ve platformları içerir:

- Intel işlemcilerde çalışan Windows 8'deki Internet Explorer haricindeki tüm platformlar için Flash Player 11.5 ActiveX denetimi
- Tüm tarayıcılar için Flash Player 11.5 eklentisi
- Adobe AIR (masaüstü ve mobil)

Bu, aşağıdaki durumlar için güncelleme adımının *hala gerekli olduğu* anlamına gelir:

- Intel işlemcilerde çalışan Windows 8'de Internet Explorer
- Flash Player 11.4 ve altı, Google Chrome 22 ve üstü (tüm platformlar) veya 21 ve üstü (Windows) hariç

Not: *Flash Player 11.5 veya üstüne sahip bir sistemde yine de güvenli şekilde* `SystemUpdater.update(SystemUpdaterType.DRM)` *öğesini çağırabilirsiniz, ancak hiçbir öğe indirilmez.*

Aşağıdaki yüksek seviyeli iş akışı, bir uygulamanın korumalı içeriği nasıl alacağını ve oynatacağını gösterir. İş akışı, uygulamanın özellikle Adobe Access tarafından korunan içeriği oynatmak için tasarlandığını varsayar:

1. İçerik meta verilerini alın.
2. Gerekirse Flash Player güncellemelerini işleyin.
3. Yerel olarak bir lisansın mevcut olup olmadığını kontrol edin. Mevcutsa, yükleyin ve adım 7'ye gidin. Mevcut değilse, adım 4'e gidin.
4. Kimlik doğrulamanın gerekli olup olmadığını kontrol edin. Gerekmiyorsa, adım 7'ye gidebilirsiniz.
5. Kimlik doğrulama gerekiyorsa, kullanıcıdan kimlik doğrulama bilgilerini alın ve lisans sunucusuna aktarın.
6. Etki alanı kaydı gerekirse etki alanına girin (AIR 3.0 ve daha üstü).
7. Kimlik doğrulama başarılı olduğunda, sunucudan lisansı indirin.
8. İçeriği oynatın.

Hata oluşmadıysa ve kullanıcı içeriği görüntülemek için başarılı bir şekilde yetkilendirildiyse, NetStream nesnesi bir DRMStatusEvent nesnesi gönderir. Ardından uygulama içeriği oynatmaya başlar. DRMStatusEvent nesnesi ilgili makbuz bilgilerini saklar, bu bilgiler kullanıcı politika ve izinlerini tanımlar. Örneğin, içeriğin çevrimdışı kullanılabilir hale getirilip getirilmediğini veya lisansın ne zaman sona ereceği ile ilgili bilgileri tutar. Uygulama, kullanıcıya politikasının durumunu bildirmek için bu bilgileri kullanabilir. Örneğin uygulama, kullanıcının içeriği görebileceği kaç günü kaldığını bir durum çubuğunda gösterebilir.

Kullanıcının çevrimdışı erişime yetkisi varsa, makbuz ön belleğe alınır ve şifrelenmiş içerik kullanıcının makinesine indirilir. İçerik, lisansı önbelleğe alma süresinde belirlenen süre boyunca erişilebilir hale getirilir. Olaydaki `detail özelli i "DRM.voucherObtained"` içerir. Uygulama, çevrimdışıyken de erişebilmesi için içeriğin yerel olarak nerede saklanacağına karar verir. DRMManager sınıfını kullanarak da makbuzları önceden yükleyebilirsiniz.

Not: *Makbuzların önbelleğe alınması ve ön yüklemesi AIR ve Flash Player tarafından desteklenir. Bununla birlikte, şifrelenmiş içeriğin indirilmesi ve depolanması yalnızca AIR'de desteklenir.*

Hata olaylarını açıkça işlemek uygulamanın sorumluluğundadır. Bu olaylar arasında, kullanıcının geçerli kimlik bilgilerini girdiği, ancak makbuzun şifreli içeriği koruyarak içeriğe erişimi kısıtladığı durumlar da bulunmaktadır. Örneğin, erişim haklarının ödemesi yapılmadıysa kimlik doğrulaması yapılmış bir kullanıcı içeriğe erişemez. Bu durum ayrıca aynı yayıncının iki kayıtlı üyesi, yalnızca bir tanesinin ödeme yaptığı bir içeriği paylaşmak istediklerinde de gerçekleşebilir. Uygulama, kullanıcıyı hata konusunda bilgilendirmeli ve alternatif bir öneri sunmalıdır. Tipik bir alternatif öneri, hakları görmek için nasıl kayıt olunacağını ve ödeme yapılacağını belirten talimatlardır.

Ayrıntılı API iş akışı

Flash Player 10.1 ve üstü, AIR 2.0 ve üstü

Bu iş akışı, korumalı içerik iş akışına yönelik daha ayrıntılı bir görünüm sağlamaktadır. Bu iş akışı, Adobe Access tarafından korunan içeriğin oynatılmasında kullanılan özel API'leri anlatmaktadır.

1. Bir URLLoader nesnesini kullanarak, korumalı içeriğin meta veri dosyasının baytlarını yükleyin. Bu nesneyi `metadata_bytes` gibi bir değişkene ayarlayın.

   Adobe Access tarafından kontrol edilen tüm içerik Adobe Access meta verilerine sahiptir. İçerik paketlendiğinde, bu meta veriler içerikle birlikte ayrı bir meta veri dosyası (.metadata) olarak kaydedilebilir. Daha fazla bilgi için Adobe Access belgelerine bakın.

2. Bir DRMContentData örneği oluşturun. Bu kodu bir try-catch bloğuna yerleştirin:

   ```
   new DRMContentData(metadata_bytes)
   ```

   burada *`metadata_bytes`* adım 1'de elde edilen URLLoader nesnesidir.

3. (Yalnızca Flash Player) Çalışma zamanı Adobe Access modülünün olup olmadığını kontrol eder. Bulunmazsa, DRMErrorEvent hata kodu 3344 veya DRMErrorEvent hata kodu 3343 ile bir IllegalOperationError öğesi gönderilir.

   Bu hatayı düzeltmek için SystemUpdater API'sini kullanarak Adobe Access modülünü indirin. Bu modül indirildikten sonra, SystemUpdater nesnesi bir COMPLETE olayı gönderir. Bu olay için, bu olay gönderildiğinde adım 2'ye dönen bir olay dinleyici ekleyin. Aşağıdaki kod bu adımları gösterir:

   ```
   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);
   }
   ```

   Oynatıcının kendisinin güncellenmesi gerekiyorsa, bir durum olayı gönderilir. Bu olayın işlenmesi hakkında daha fazla bilgi edinmek için bkz. "Bir güncelleme olayını dinleme" sayfa 550.

   Not: *AIR uygulamalarında, AIR yükleyicisi Adobe Access modülünün güncellemesini ve gerekli çalışma zamanı güncellemelerini yapar.*

4. DRMManager nesnesinden gönderilen DRMStatusEvent ve DRMErrorEvent öğesini dinlemek için dinleyiciler oluşturun:

   ```
   DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
   DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
   ```

   DRMStatusEvent dinleyicisinde, makbuzun geçerli olup olmadığını (null değerinde olmadığını) kontrol edin. DRMErrorEvent dinleyicisinde, DRMErrorEvents öğesini ele alın. Bkz. "DRMStatusEvent sınıfını kullanma" sayfa 543 ve "DRMErrorEvent sınıfını kullanma" sayfa 547.

5 İçeriği oynatmak için gereken makbuzu (lisansı) yükleyin.

İlk olarak, içeriği oynatmak için yerel olarak depolanmış bir lisans yüklemeyi deneyin:

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

Yükleme tamamlandıktan sonra, DRMManager nesnesi `DRMStatusEvent.DRM_Status` öğesini gönderir.

6 DRMVoucher nesnesi null değerinde değilse, makbuz geçerlidir. Adım 13'e atlayın.

7 DRMVoucher null değerindeyse, bu içerik için ilkelerin gerektirdiği kimlik doğrulama yöntemini kontrol edin. `DRMContentData.authenticationMethod` özelliğini kullanın.

8 Kimlik doğrulama yöntemi `ANONYMOUS` ise adım 13'e gidin.

9 Kimlik doğrulama yöntemi `USERNAME_AND_PASSWORD` ise, uygulamanız, kullanıcının bilgilerini girebilmesine izin verecek bir mekanizmaya sahip olmalıdır. Kullanıcının kimlik doğrulamasını yapmak için lisans sunucusuna kimlik bilgilerini aktarın:

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

DRMManager, kimlik doğrulama başarısız olursa `DRMAuthenticationErrorEvent` olayını, kimlik doğrulama başarılı olursa `DRMAuthenticationCompleteEvent` olayını gönderir. Bu olaylar için dinleyiciler oluşturun.

10 Doğrulama yöntemi `UNKNOWN` ise, özel bir doğrulama yöntemi kullanılmalıdır. Bu durumda içerik sağlayıcı, doğrulamanın ActionScript 3.0 API'leri kullanmadan bant dışı bir yolla yapılmasını sağlar. Özel doğrulama yordamı, `DRMManager.setAuthenticationToken()` yöntemine geçirilebilen bir doğrulama simgesi oluşturmalıdır.

11 Kimlik doğrulama başarısız olursa, uygulamanız adım 9'a dönmelidir. Uygulamanızın kimlik doğrulama hatalarını işleyecek ve tekrarını sınırlayacak bir mekanizmaya sahip olduğundan emin olun. Örneğin, üç girişimin ardından kullanıcıya kimlik doğrulamanın başarısız olduğunu ve içeriğin gösterilemeyeceğini belirten bir mesaj görüntülenir.

12 Kullanıcının bilgilerini girmesini istemek yerine saklanmış işareti kullanmak için, işareti `DRMManager.setAuthenticationToken()` yöntemiyle ayarlayın. Ardından lisans sunucusundan lisansı indirin ve içeriği Adım 8'deki gibi oynatın.

13 (isteğe bağlı) Kimlik doğrulama başarılı olursa, önbelleğe alınmış bir bayt dizisi olan kimlik doğrulama işaretini yakalayabilirsiniz. Bu işareti `DRMAuthenticationCompleteEvent.token` özelliğiyle alın. Kullanıcının bu içeriğe ait kimlik bilgilerini tekrar tekrar girmesine gerek kalmaması için kimlik doğrulama işaretini saklayıp kullanabilirsiniz. Bu lisans sunucusu, kimlik doğrulama işaretinin geçerlilik süresini belirler.

14 Kimlik doğrulama başarılı olduğunda, lisans sunucusundan lisansı indirin:

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

Yükleme tamamlandıktan sonra, DRMManager nesnesi DRMStatusEvent.DRM_STATUS olayını gönderir. Bu olayı dinleyin; gönderildiğinde, içeriği oynatabilirsiniz.

15 Bir NetStream nesnesi oluşturup ardından `play()` yöntemini çağırarak videoyu oynatın:

```
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 ve session nesneleri

`DRMContentData` oluşturulduğunda Flash Player DRM modülüne ilişkin bir session nesnesi kullanılır. Bu `DRMContentData` öğesini alan tüm `DRMManager` API'leri bu belirli DRM modülünü kullanır. Bununla birlikte, `DRMContentData` öğesini kullanmayan 2 `DRMManager` API'si vardır. Bu kurallar şunlardır:

1. `authenticate()`
2. `setAuthenticationToken()`

İlişkili `DRMContentData` olmadığından, bu `DRMManager` API'leri çağırıldığında diskteki en yeni DRM modülü kullanılır. DRM modülünün güncellenmesi, uygulamanın DRM iş akışının ortasında gerçekleşirse bu bir problem haline gelebilir. Aşağıdaki senaryoyu göz önünde bulundurun:

1. Uygulama, *AdobeCP1* öğesini DRM modülü olarak kullanan bir `DRMContentData` nesnesi `contentData1` oluşturur.
2. Uygulama, `DRMManager.authenticate(contentData1.serverURL,...)` yöntemini çağırır.
3. Uygulama, `DRMManager.loadVoucher(contentData1, ...)` yöntemini çağırır.

Uygulama 2. adıma geçmeden önce DRM modülü güncellemesi olursa, bu durumda `DRMManager.authenticate()` yöntemi, kimlik doğrulamayı *AdobeCP2* öğesini DRM modülü olarak kullanarak gerçekleştirir. 3. adımda `loadVoucher()` yöntemi, DRM modülü olarak *AdobeCP1* öğesini kullanmaya devam ettiğinden hata verir. Güncelleme, DRM modülü güncellemesini çağıran başka bir uygulama nedeniyle gerçekleşmiş olabilir. Uygulama başlangıcında DRM modülünü çağırarak bu senaryodan kaçınabilirsiniz.

DRM ile ilgili olaylar

Bir uygulama korumalı içeriği oynatmaya çalıştığında, çalışma zamanı çok sayıda olay gönderir:

- DRMManager tarafından gönderilen DRMDeviceGroupErrorEvent (yalnızca AIR)
- NetStream tarafından gönderilen DRMAuthenticateEvent (sadece AIR)
- DRMManager tarafından gönderilen DRMAuthenticationCompleteEvent
- DRMManager tarafından gönderilen DRMAuthenticationErrorEvent
- NetStream ve DRMManager tarafından gönderilen DRMErrorEvent
- NetStream ve DRMManager tarafından gönderilen DRMStatusEvent
- StatusEvent
- NetStatusEvent. Bkz. “Bir güncelleme olayını dinleme” sayfa 550

Adobe Access tarafından korunan içeriği desteklemek üzere DRM olaylarını işleyecek olay dinleyicilerini ekleyin.

Çevrimdışı oynatma için makbuzları önceden yükleme

Adobe AIR 1.5 ve üstü

Adobe Access tarafından korunan içeriği oynatmak için gereken makbuzları (lisansları) önceden yükleyebilirsiniz. Önceden yüklenmiş makbuzlar kullanıcıların etkin İnternet bağlantıları olmasa da içeriği görüntüleyebilmelerine izin verir. (Ön yükleme işleminin kendisi de bir İnternet bağlantısı gerektirir.) Makbuzları önceden yüklemek için NetStream sınıfı `preloadEmbeddedMetadata()` yöntemini ve DRMManager sınıfını kullanabilirsiniz. AIR 2.0 ve üstünde, makbuzları önceden doğrudan yüklemek için bir DRMContentData nesnesi kullanabilirsiniz. Bu teknik, DRMContentData nesnesini içerikten bağımsız olarak güncellemenize imkan verdiği için tercih edilir. (`preloadEmbeddedData()` yöntemi, içerikten DRMContentData’yı getirir.)

DRMContentData'yı kullanma

Adobe AIR 2.0 ve üstü

Aşağıdaki adımlar korumalı bir medya dosyası makbuzunu, bir DRMContentData nesnesi kullanarak önceden yüklemeye yönelik iş akışını açıklar.

1. Paketlenmiş içeriğe ait ikili meta verileri alın. Adobe Access Java Referans Paketleyicisi kullanılıyorsa, bu meta veri dosyası otomatik olarak *.metadata* uzantısıyla oluşturulur. Örneğin bu meta verileri URLLoader sınıfını kullanarak indirebilirsiniz.
2. Yapıcı işlevine meta verileri geçirerek bir DRMContentData nesnesi oluşturun:

   ```
   var drmData:DRMContentData = new DRMContentData( metadata );
   ```

3. Geri kalan adımlar, "Korumalı içerik iş akışını anlama" sayfa 535 bölümünde anlatılan iş akışıyla aynıdır.

preloadEmbeddedMetadata() kullanma

Adobe AIR 1.5 ve üstü

Aşağıdaki adımlar, `preloadEmbeddedMetadata()` öğesini kullanan DRM korumalı bir medya dosyası makbuzunu önceden yüklemeye ilişkin iş akışını açıklar:

1. Medya dosyasını indirin ve saklayın. (DRM meta verileri yalnızca yerel olarak depolanmış dosyalardan önceden yüklenebilir.)
2. NetStream istemci nesnesinin `onDRMContentData()` ve `onPlayStatus()` geri çağırma işlevleri için uygulamaları sağlayarak NetConnection ve NetStream nesnelerini oluşturun.
3. Bir NetStreamPlayOptions nesnesi oluşturun ve `stream` özelliğini yerel medya dosyasının URL'sine ayarlayın.
4. Ayrıştırılacak medya dosyasını tanımlayan NetStreamPlayOptions nesnesini aktaran NetStream `preloadEmbeddedMetadata()` yöntemini çağırın.
5. Medya dosyası DRM meta verileri içeriyorsa, `onDRMContentData()` geri çağırma işlevi çağrılır. Meta veriler bu işleve bir DRMContentData nesnesi olarak aktarılır.
6. DRMManager `loadVoucher()` yöntemini kullanarak makbuzu elde etmek için DRMContentData nesnesini kullanın.

   `DRMContentData` nesnesine ait `authenticationMethod` özelliğinin değeri `flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD` ise, kullanıcıyı makbuzu yüklemeden önce medya hakları sunucusundan doğrulayın. DRMContentData nesnesinin `serverURL` ve `domain` özellikleri, kullanıcının kimlik bilgileriyle birlikte DRMManager `authenticate()` yöntemine aktarılabilir.
7. Dosya ayrıştırma tamamlandığında `onPlayStatus()` geri çağırma işlevi çağrılır. `onDRMContentData()` işlevi çağrılmadıysa, dosya makbuzu almak için gerekli olan meta verileri içermez. Bu eksik çağrı büyük ihtimalle Adobe Access'in bu dosyayı korumadığı anlamına da gelir.

Aşağıdaki AIR kod örneği, yerel bir medya dosyası için makbuzunun nasıl önceden yükleneceğini gösterir:

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

NetStream sınıfının DRM ilişkili üyeleri ve olayları

Flash Player 10.1 ve üstü, Adobe AIR 1.0 ve üstü

NetStream sınıfı, Flash Player veya bir AIR uygulamasıyla Flash Media Server veya yerel dosya sistemi arasında tek yönlü bir akış sağlar. (NetStream sınıfı ayrıca aşamalı indirmeyi de destekler.) Bir NetStream nesnesi, NetConnection nesnesinin içindeki kanaldır. NetStream sınıfı DRM ile ilgili dört olay gönderir:

Olay	Açıklama
drmAuthenticate (Yalnızca AIR)	DRMAuthenticateEvent sınıfında tanımlanır. Bu olay bir NetStream nesnesi, oynatılmadan önce kimlik doğrulama için kullanıcı kimlik bilgisi gerektiren korumalı içeriği oynatmaya çalıştığında gönderilir. Bu olayın özellikleri, kullanıcının kimlik bilgilerini almada ve ayarlamada kullanılabilen header, usernamePrompt, passwordPrompt ve urlPrompt özelliklerini içerir. Bu olay, NetStream nesnesi geçerli kullanıcı kimlik bilgilerini alana kadar tekrarlanır.
drmError	DRMErrorEvent sınıfında tanımlanıp bir NetStream nesnesi korumalı içeriği oynatmaya çalıştığında ve DRM ile ilgili bir hatayla karşılaştığında gönderilir. Örneğin DRM error olayı nesnesi, kullanıcı kimlik denetimi başarısız olduğunda gönderilir. Bu hata kullanıcı içeriği görmek için gereken hakları satın almadığından oluşabilir. İçerik sağlayıcısı uygulamanın görüntülenmesini desteklemediği için de oluşabilir.
drmStatus	DRMStatusEvent sınıfında tanımlanır. Bu olay, korumalı içerik oynamaya başladığında (kullanıcı doğrulandığında ve içeriği oynatmasına izin verildiğinde) gönderilir. DRMStatusEvent nesnesi makbuzla ilgili bilgiler içerir. Makbuz bilgileri, çevrimdışıyken veya makbuzun süresi dolduğunda ve içerik artık görüntülenemediğinde içeriğin kullanılabilir kılınıp kılınamayacağına dair bilgileri içerir.
status	events.StatusEvent içerisinde tanımlanır ve yalnızca uygulama korumalı içeriği oynatmaya çalıştığında NetStream.play() yöntemini çağırarak gönderilir. status kod özelliği "DRM.encryptedFLV" şeklindedir.

NetStream sınıfı yalnızca AIR'de kullanım için aşağıdaki DRM'ye özel yöntemlere içerir:

Yöntem	Açıklama
resetDRMVouchers()	Yerel olarak önbelleğe alınmış tüm dijital haklar yönetimi (DRM) makbuz bilgilerini siler. Kullanıcının şifreli içeriğe erişebilmesi için uygulamanın makbuzları yeniden indirmesi gerekir. Örneğin, aşağıdaki kod tüm makbuzları önbellekten kaldırır: NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	Kimlik doğrulama için NetStream nesnesine kullanıcı adı, şifre ve kimlik doğrulamasından oluşan bir kimlik doğrulama bilgileri kümesi iletir. Geçerli kimlik denetimi türleri `"drm"` ve `"proxy"` türleridir. `"drm"` kimlik doğrulama türü ile, temin edilen kimlik bilgileri Adobe Access için doğrulanır. `"proxy"` kimlik doğrulama türüyle, kimlik bilgileri proxy sunucusu için doğrulanır ve bilgilerin proxy sunucusu tarafından gerekli kılınanlarla eşleşmesi gerekir. Örneğin, bir kuruluş kullanıcı Internete erişmeden önce uygulamanın bir proxy sunucusu için kimlik doğrulaması yapmasını gerektirebilir. Proxy seçeneği bu tür kimlik doğrulamasına izin verir. Adsız kimlik doğrulaması kullanılmadığı sürece, kullanıcı proxy kimlik doğrulamasından sonra makbuzu almak ve içeriği oynatmak üzere Adobe Access için kimlik doğrulaması yapmalıdır. Adobe Access kimlik doğrulaması yapmak için `setDRMAuthenticationCredentials()` öğesini `"drm"` seçeneğiyle ikinci kez kullanabilirsiniz.
preloadEmbeddedMetadata()	Gömülü meta veriler için bir yerel medya dosyasını ayrıştırır. DRM ilişkili meta veriler bulunduğunda, AIR `onDRMContentData()` geri çağırma işlevini çağırır.

Ayrıca, AIR'de bir NetStream nesnesi `preloadEmbeddedMetaData()` yöntemine yapılan çağrının sonucu olarak `onDRMContentData()` ve `onPlayStatus()` geri çağırma işlevlerini çağırır. Bir medya dosyasında DRM meta verileriyle karşılaşıldığında `onDRMContentData()` işlevi çağrılır. Dosya ayrıştırıldığında `onPlayStatus()` işlevi çağrılır. NetStream örneğine atanan `client` nesnesinde `onDRMContentData()` ve `onPlayStatus()` işlevleri tanımlanmalıdır. Makbuzları önceden yüklemek ve içeriği oynatmak için aynı NetStream nesnesini kullanıyorsanız, oynatmaya başlamadan önce `preloadEmbeddedMetaData()` tarafından oluşturulan `onPlayStatus()` çağrısını beklemelisiniz.

Aşağıdaki AIR kodunda kullanıcı adı ("administrator"), şifre ("password") ve "drm" kimlik doğrulama türü, kullanıcının kimliğini denetlemek üzere ayarlanmıştır. setDRMAuthenticationCredentials() yöntemi, içerik sağlayıcısı tarafından bilinen ve kabul edilen kimlik bilgileriyle eşleşen kimlik bilgilerini sağlamak zorundadır. Bu kimlik bilgileri, kullanıcının içeriği görüntülemesine izin veren kullanıcı bilgileriyle aynıdır. Video oynatmaya ve video akışına başarılı bir bağlantı yapıldığını doğrulamaya ilişkin kod burada yer almamaktadır.

```
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");
}
```

DRMStatusEvent sınıfını kullanma

Flash Player 10.1, Adobe AIR 1.0 ve üstü

Adobe Access tarafından korunan içerik başarılı bir şekilde oynamaya başladığında, NetStream nesnesi bir DRMStatusEvent nesnesi gönderir. (Başarı, lisansın doğrulandığı ve kullanıcının kimlik doğrulamasından geçip içeriği görüntüleme iznini aldığı anlamına gelir). DRMStatusEvent erişim izni olan adsız kullanıcılar için de gönderilir. Lisans, kimlik doğrulamasından geçmesi gerekmeyen adsız kullanıcının içeriği oynatmak için izni olup olmadığının doğrulanması amacıyla kontrol edilir. Adsız kullanıcıların erişimi çeşitli nedenlerden reddedilebilir. Örneğin, adsız bir kullanıcı içeriğe lisansın süresi dolduğu için erişemez.

DRMStatusEvent nesnesi lisansla ilgili bilgiler içerir. Bu bilgilere, çevrimdışıyken ya da makbuzun süresi dolduğunda ve içerik artık görüntülenemediğinde içeriğin kullanılır kılınıp kılınamayacağına dair bilgiler dahildir. Uygulama bu verileri kullanıcının politika durumunu ve izinlerini taşımak için kullanabilir.

DRMStatusEvent özellikleri

Flash Player 10.1, Adobe AIR 1.0 ve üstü

DRMStatusEvent sınıfı aşağıdaki özellikleri içerir. Bazı özellikler AIR'nin 1.0 üstü sürümlerinde kullanılır hale gelmiştir. Tam sürüm bilgileri için bkz. *Adobe® Flash® Professional CS5 için ActionScript 3.0 Başvurusu*.

Flash Player 10.1'de desteklenmeyen özellikler için, DRMVoucher sınıfı Flash Player için benzer özellikler sağlar.

Özellik	Açıklama
contentData	İçerikte gömülü olan DRM meta verilerini içeren bir DRMContentData nesnesi.
detail (Yalnızca AIR)	status olayının bağlamını açıklayan dize. DRM 1.0 sürümünde tek geçerli değer DRM.voucherObtained değeridir.
isAnonymous (Yalnızca AIR)	Adobe Access ile korunan içeriğin, kullanıcının kimlik doğrulama bilgilerini vermesini gerektirmeden kullanılabilir (true) veya kullanılmaz (false) olduğunu belirtir. Bir false değeri, kullanıcının, içerik sağlayıcı tarafından bilinen ve beklenenle eşleşen bir kullanıcı adı ve şifresi sağlaması gerektiği anlamına gelir.
isAvailableOffline (yalnızca AIR)	Adobe Access ile korunan içeriğin çevrimdışıyken kullanılabilir (true) veya kullanılamaz (false) olduğunu belirtir. Dijital olarak korunan içeriğin çevrimdışıyken kullanılabilir olması için, makbuzunun kullanıcının yerel bilgisayarında önbelleğe alınmış olması gerekir.
isLocal	İçeriği oynatmak için gerekli olan makbuzun yerel olarak önbelleğe alınmış olup olmadığını gösterir.

Özellik	Açıklama
offlineLeasePeriod (yalnızca AIR)	İçeriğin çevrimdışıyken görüntülenebileceği kalan gün sayısı.
policies (Yalnızca AIR)	Özel DRM özellikleri içerebilen özel nesne.
voucher	DRMVoucher.
voucherEndDate (Yalnızca AIR)	Makbuzun süresinin dolduğu ve içeriğin artık görüntülenemediği mutlak tarih.

DRMStatusEvent işleyicisi oluşturma

Flash Player 10.1, Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, olayı başlatan NetStream nesnesi için DRM içerik durumu bilgilerini çıkaran bir olay işleyicisi oluşturur. Bu olay işleyicisini korumalı içeriği işaret eden bir NetStream nesnesine ekleyin.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

DRMAuthenticateEvent sınıfını kullanma

Adobe AIR 1.0 ve üstü

DRMAuthenticateEvent nesnesi bir NetStream nesnesi, oynatılmadan önce kimlik doğrulama için kullanıcı kimlik bilgisi gerektiren korumalı içeriği oynatmaya çalıştığında gönderilir.

DRMAuthenticateEvent işleyicisi gerekli kimlik bilgilerini (kullanıcı adı, şifre ve tür) toplamaktan ve değerleri doğrulama için `NetStream.setDRMAuthenticationCredentials()` yöntemine iletmekten sorumludur. Her AIR uygulaması kullanıcı kimlik belgelerini edinmek için bazı mekanizmalar sağlamalıdır. Örneğin uygulama, kullanıcı adı ve şifre değerlerini girmesi için kullanıcıya basit bir kullanıcı arabirimi sağlayabilir. Bunun yanında, tekrarlayan kimlik doğrulama girişimlerini işlemek ve sınırlamak için bir mekanizma oluşturun.

DRMAuthenticateEvent özellikleri

Adobe AIR 1.0 ve üstü

DRMAuthenticateEvent sınıfı aşağıdaki özellikleri içerir:

Özellik	Açıklama
authenticationType	Verilen kimlik bilgilerinin Adobe Access (“drm”) veya proxy sunucusu (“proxy”) öğelerinden hangisine yönelik kimlik doğrulama yaptığını belirtir. Örneğin, "proxy" seçeneği kullanıcının İnternet’e erişebilmesi için uygulamanın gerektiğinde proxy sunucusuna yönelik kimlik doğrulaması yapmasına izin verir. Adsız kimlik doğrulaması kullanılmadığı sürece, kullanıcı proxy kimlik doğrulamasından sonra makbuzu almak ve içeriği oynatmak üzere Adobe Access için kimlik doğrulaması yapmalıdır. Adobe Access kimlik doğrulaması yapmak için setDRMAuthenticationcredentials() öğesini "drm" seçeneğiyle ikinci kez kullanabilirsiniz.
header	Sunucu tarafından sağlanan şifreli içerik dosya üstbilgisi. Şifreli içeriğin bağlamına ilişkin bilgi içerir. Bu üstbilgi dizesi, uygulamanın bir kullanıcı adı - şifre iletişim kutusu oluşturmasını sağlamak üzere Flash uygulamasına iletilebilir. Üstbilgi dizesi, iletişim kutusu talimatları olarak kullanılabilir. Örneğin üstbilgi, "Kullanıcı adınızı ve şifrenizi girin" olabilir.
netstream	Bu olayı başlatan NetStream nesnesi.
passwordPrompt	Sunucu tarafından sağlanan, şifre kimlik bilgisi istemi. Dize, istenilen şifre türü için talimat içerebilir.
urlPrompt	Sunucu tarafından sağlanan, URL dizesi istemi. Dize, kullanıcı adı ve şifrenin gönderildiği konumu sağlayabilir.
usernamePrompt	Sunucu tarafından sağlanan, kullanıcı adı kimlik bilgisi istemi. Dize, istenilen kullanıcı adı türü için talimat içerebilir. Örneğin, bir içerik sağlayıcı, kullanıcı adı olarak e-posta adresi isteyebilir.

Bahsi geçen dizeler yalnızca FMRMS sunucusu tarafından sağlanır. Adobe Access Sunucusu bu dizeleri kullanmaz.

DRMAuthenticateEvent işleyicisi oluşturma

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, olayı başlatan NetStream nesnesine kodlanmış kimlik denetimi bilgileri ileten bir olay işleyicisi oluşturur. (Video oynatmaya ve video akışına başarılı bir bağlantı yapıldığını doğrulamaya ilişkin kod burada yer almamaktadır.)

```
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");
}
```

Kullanıcı kimlik bilgilerini almak için bir arabirim oluşturma

Adobe AIR 1.0 ve üstü

Korumalı içeriğin kullanıcı kimlik bilgilerini gerektirdiği durumlarda, AIR uygulamasının kullanıcının kimlik doğrulama bilgilerini genellikle bir kullanıcı arabirimi üzerinden alması gerekir.

Aşağıda kullanıcı kimlik bilgilerini almaya yönelik basit bir Flex kullanıcı arabirimi örneği bulunmaktadır. Kullanıcı adı ve şifre kimlik bilgilerinin her biri için birer tane olmak üzere iki TextInput nesnesi içeren bir panel nesnesinden oluşur. Panel ayrıca `credentials()` yöntemini başlatan bir düğme de içerir.

```
<mx:Panel x="236.5" y="113" width="325" height="204" layout="absolute" title="Login">
    <mx:TextInput x="110" y="46" id="uName"/>
    <mx:TextInput x="110" y="76" id="pWord" displayAsPassword="true"/>
    <mx:Text x="35" y="48" text="Username:"/>
    <mx:Text x="35" y="78" text="Password:"/>
    <mx:Button x="120" y="115" label="Login" click="credentials()"/>
</mx:Panel>
```

`credentials()` yöntemi, kullanıcı adı ve şifre değerlerini `setDRMAuthenticationCredentials()` yöntemine ileten kullanıcı tanımlı bir yöntemdir. Değerler iletildikten sonra, `credentials()` yöntemi TextInput nesnelerinin değerlerini sıfırlar.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

Bu tür basit bir arabirimi yüklemenin bir yolu paneli yeni durumun bir parçası olarak eklemektir. Yeni durum, DRMAuthenticateEvent nesnesi atıldığında temel durumdan oluşur. Aşağıdaki örnek, korumalı bir FLV'yi işaret eden kaynak niteliğine sahip bir VideoDisplay nesnesi içerir. Bu durumda, uygulamayı da temel duruma döndürmesi için `credentials()` yöntemi değiştirilir. Bu yöntem kullanıcı kimlik bilgilerini ilettikten ve TextInput nesne değerlerini sıfırladıktan sonra bu işlemi gerçekleştirir.

```
<?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>
```

DRMErrorEvent sınıfını kullanma

Flash Player 10.1 ve üstü, Adobe AIR 1.0 ve üstü

Adobe Flash Player ve Adobe AIR, korumalı içerik oynatmaya çalışan bir NetStream nesnesi DRM ile ilgili bir hatayla karşılaştığında bir DRMErrorEvent nesnesi gönderir. Bir AIR uygulamasında kullanıcı bilgileri geçersizse, DRMAuthenticateEvent nesnesi kullanıcı geçerli bilgileri girene veya uygulama daha fazla girişimde bulunmayı engelleyene kadar tekrar tekrar gönderir. Uygulama DRM ile ilgili hataları saptamak, tanımlamak ve işlemek için tüm diğer DRM hata olaylarını dinlemekten sorumludur.

Geçerli kimlik bilgilerine sahip olunsa bile içerik makbuzunun şartları hala bir kullanıcının şifreli içeriği görmesine engel olabilir. Örneğin, bir kullanıcının yetkisiz bir uygulamadaki içeriği görmeye çalışırken erişimi engellenebilir. Yetkisiz bir uygulama, şifreli içeriğin yayıncısının doğrulamadığı uygulamadır. Bu durumda bir DRMErrorEvent nesnesi gönderilir.

İçerik bozuksa veya uygulamanın sürümü makbuz tarafından belirtilenle eşleşmiyorsa, error olayları da çalıştırılabilir. Uygulama hataların işlenmesi için uygun mekanizmayı sağlamalıdır.

DRMErrorEvent özellikleri

Flash Player 10.1 ve üstü, Adobe AIR 1.0 ve üstü

Hataların tam listesi için bkz. Adobe® Flash® Professional CS5 için ActionScript® 3.0 Başvurusu içinde Çalışma Zamanı Hata Kodları. DRM ile ilgili hatalar, hata 3300'de başlar.

DRMErrorEvent işleyicisi oluşturma

Flash Player 10.1 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, olayı başlatan NetStream nesnesi için bir olay işleyicisi oluşturur. NetStream, korumalı içeriği oynatma girişiminde bulunurken bir hatayla karşılaşırsa çağrılır. Genelde bir uygulama hatayla karşılaştığında herhangi bir sayıda temizleme görevi gerçekleştirir. Ardından kullanıcıyı hatayla ilgili bilgilendirir ve sorunu çözmek için seçenekler sağlar.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

DRMManager sınıfını kullanma

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

Uygulamalarda makbuzları ve medya hakları sunucu oturumlarını yönetmek için DRMManager sınıfını kullanın.

Makbuz yönetimi (yalnızca AIR)

Bir kullanıcı ne zaman korumalı içerik oynatmak istese, çalışma zamanı içeriğin görüntülenmesi için gereken lisansı elde edip ön belleğe alır. Uygulama dosyayı yerel olarak kaydediyorsa ve lisans çevrimdışı oynatmaya izin veriyorsa, kullanıcı içeriği AIR uygulamasında görüntüleyebilir. Bu tür yerel çevrimdışı oynatma işlemleri, medya hakları sunucusuna bağlantı olmasa bile başarılı olur. DRMManager ve NetStream `preloadEmbeddedMetadata()` yöntemi kullanılarak, makbuzu ön belleğe alabilirsiniz. Uygulama, içeriği görüntülemek için gereken lisansı elde etmek zorunda değildir. Örneğin, uygulamanız medya dosyasını indirebilir ve daha sonra kullanıcı halen çevrimiçiyken makbuzu alabilir.

Makbuzu önceden yüklemek için DRMContentData nesnesini almak amacıyla NetStream `preloadEmbeddedMetadata()` yöntemini kullanın. DRMContentData nesnesi, lisansı sağlayabilen ve kullanıcı kimlik doğrulamasının gerekli olup olmadığını açıklayan medya hakları sunucusunun URL'si ve etki alanını içerir. Bu bilgiyle, makbuzu almak ve önbelleğe kaydetmek için DRMManager `loadVoucher()` yöntemini çağırabilirsiniz. Makbuzların önceden yüklenmesine dair iş akışı "Çevrimdışı oynatma için makbuzları önceden yükleme" sayfa 538 öğesinde daha ayrıntılı bir şekilde anlatılmıştır.

Oturum yönetimi

Medya hakları sunucusunda kullanıcının kimlik doğrulamasını yapmak ve kalıcı oturumları yönetmek için DRMManager da kullanabilirsiniz.

Medya hakları sunucusuyla bir oturum oluşturmak için DRMManager `authenticate()` yöntemini kullanın. Kimlik doğrulama başarıyla tamamlandığında, DRMManager bir DRMAuthenticationCompleteEvent nesnesi gönderir. Bu nesne bir oturum belirteci içerir. Bu belirteci daha sonraki oturumları oluşturmak için kaydedebilir ve kullanıcının hesap kimlik bilgilerini girmesine gerek kalmamasını sağlayabilirsiniz. Yeni bir kimliği doğrulanmış oturum oluşturmak için simgeyi `setAuthenticationToken()` yöntemine iletin. (İşareti oluşturan sunucunun ayarları işaretin süresinin dolma zamanını ve diğer nitelikleri belirler. Yapı gelecekteki AIR güncellemelerinde değişebileceğinden, AIR uygulama kodu işaret verisi yapısını yorumlamamalıdır.)

Kimlik doğrulama işaretleri diğer bilgisayarlara aktarılabilir. Belirteçleri korumak için AIR Şifreli Yerel Depo'ya kaydedebilirsiniz. Daha fazla bilgi için bkz. "Şifrelenmiş yerel depolama" sayfa 718.

DRMStatus Olayları

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

`loadVoucher()` yöntemine yapılan çağrı başarıyla tamamlandıktan sonra, DRMManager bir DRMStatusEvent nesnesi gönderir.

Makbuz alındığında, event nesnesinin `detail` özelliği (yalnızca AIR) "DRM.voucherObtained" değerini alır ve `voucher` özelliği DRMVoucher nesnesini içerir.

Makbuz alınmazsa, `detail` özelliği (yalnızca AIR) "DRM.voucherObtained" değerinde kalır; ancak `voucher` özelliği `null` değerindedir. Örneğin, *localOnly* LoadVoucherSetting özelliğini kullanıyorsanız ve yerel olarak önbelleğe alınmış makbuz mevcut değilse, makbuz alınamaz.

`loadVoucher()` çağrısı muhtemelen bir kimlik doğrulama veya iletişim hatası nedeniyle başarıyla tamamlanmadığında, DRMManager bunun yerine bir DRMErrorEvent veya DRMAuthenticationErrorEvent nesnesi gönderir.

DRMAuthenticationComplete olayları

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

`authenticate()` yöntemine yapılan çağrı aracılığıyla kullanıcının kimlik doğrulaması başarıyla tamamladığında, DRMManager bir DRMAuthenticationCompleteEvent nesnesi gönderir.

DRMAuthenticationCompleteEvent nesnesi, uygulama oturumları boyunca kullanıcı kimlik doğrulaması için kalıcı olacak yeniden kullanılabilir bir belirteç içerir. Oturumu yeniden oluşturmak için bu belirteci DRMManager `setAuthenticationToken()` yöntemine aktarın. (Süre dolması gibi işaret niteliklerini işaret oluşturucusu ayarlar. Adobe, belirteç niteliklerini inceleme amaçlı API sağlamaz.)

DRMAuthenticationError olayları

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

Kullanıcının kimlik doğrulaması `authenticate()` veya `setAuthenticationToken()` yöntemleri aracılığıyla başarıyla tamamlandığında, DRMManager bir DRMAuthenticationErrorEvent nesnesi gönderir.

DRMContentData sınıfını kullanma

Flash Player 10.1 ve üstü, Adobe AIR 1.5 ve üstü

DRMContentData nesnesi Adobe Access tarafından korunan içeriğin meta veri özelliklerini içerir. DRMContentData özellikleri, içeriği görüntülemeye ilişkin lisans makbuzunu almak için gerekli bilgileri içerir. DRMContentData sınıfını kullanarak içerikle ilişkili meta veri dosyasını "Ayrıntılı API iş akışı" sayfa 536 bölümünde anlatılan şekilde elde edebilirsiniz.

Daha fazla bilgi için Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki DRMContentData sınıfına bakın.

Adobe Access desteği için Flash Player'ı güncelleme

Flash Player 10.1 ve üzeri

Önemli: Flash Player 11.5 ve üzeri, Adobe Access modülü ile entegre olur, böylece güncelleme adımına (`SystemUpdater.update(SystemUpdaterType.DRM)` öğesinin çağrılmasına) gerek kalmaz. Bu aşağıdaki tarayıcıları ve platformları içerir:

- Windows 8'deki Internet Explorer haricindeki tüm platformlar için Flash Player 11.5 ActiveX denetimi
- Tüm tarayıcılar için Flash Player 11.5 eklentisi
- Adobe AIR (masaüstü ve mobil)

Bu, aşağıdaki durumlar için güncelleme adımının *hala gerekli olduğu* anlamına gelir:

- Windows 8'de Internet Explorer
- Flash Player 11.4 ve altı, Google Chrome 22 ve üstü (tüm platformlar) veya 21 ve üstü (Windows) hariç

Not: *Flash Player 11.5 veya üstüne sahip bir sistemde yine de güvenli şekilde* `SystemUpdater.update(SystemUpdaterType.DRM)` *öğesini çağırabilirsiniz, ancak hiçbir öğe indirilmez.*

Adobe Access'i desteklemek için Flash Player Adobe Access modülünü gerektirir. Flash Player korumalı içeriği oynatmaya çalıştığında, çalışma zamanı modülün mü, yoksa daha yeni bir Flash Player sürümünün mü indirilmesi gerektiğini belirtir. Bu yolla, Flash Player SWF geliştiricilerine arzu ederlerse güncelleme yapmama seçeneği sunar.

Çoğu durumda, korumalı içeriği oynatırken, SWF geliştiricileri gerekli Adobe Access modülüne veya oynatıcısına güncelleme yapar. Güncelleme için, Adobe Access modülü veya Flash Player uygulamasının son sürümünü elde etmek üzere SystemUpdater API'sini kullanabilirsiniz.

SystemUpdater API’si bir defada yalnızca bir güncellemeye izin verir. Hata kodu 2202, geçerli çalışma zamanı örneğinde veya başka bir örnekte bir güncellemenin zaten devam etmekte olduğunu belirtir. Örneğin, Internet Explorer’da bir Flash Player örneğinde güncelleme yapılırken Firefox’taki Flash Player örneğinde başka bir güncelleme devam edemez.

SystemUpdater API’si yalnızca masaüstü platformları için desteklenir.

Not: *Flash Player’ın 10.1’den önceki sürümleri için, eski oynatıcı sürümlerinde desteklenen güncelleme mekanizmasını kullanın (manüel indirme ve www.adobe.com.tr adresinden veya ExpressInstall ile yükleme). Ayrıca, AIR yükleyicisi Adobe Access için gereken güncellemeleri yapar ve SystemUpdater API'sini desteklemez.*

Bir güncelleme olayını dinleme

Flash Player 10.1 ve üstü

Adobe Access modülü için bir güncelleme gerektiğinde, NetStream nesnesi `DRM.UpdateNeeded` kod değeri ile bir NetStatusEvent öğesi gönderir. Bu değer, NetStream nesnesinin korumalı akışı geçerli olarak yüklü Adobe Access modüllerinden hiçbiriyle oynatamadığını gösterir. Bu olayı dinleyin ve aşağıdaki kodu çağırın:

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

Bu kod, oynatıcıda yüklü Adobe Access modülünü günceller. Bu modül güncellemesi için kullanıcı onayı gerekmez.

Eğer Adobe Access modülü bulunmuyorsa, bir hata verilir. Bkz. “Ayrıntılı API iş akışı” sayfa 536, adım 3.

Not: *play(), oynatıcıların 10.1 önceki sürümlerindeki şifrelenmiş bir akışta çağrılırsa, kod değeri NetStream.Play.StreamNotFound olan bir NetStatusEvent gönderilir. Eski oynatıcılarda, bu oynatıcılar için desteklenen güncelleme mekanizmasını kullanın (manüel indirme ve www.adobe.com.tr adresinden ya da ExpressInstall ile yükleme).*

Oynatıcının kendisi için bir güncelleme gerekiyorsa, SystemUpdater nesnesi kod değeri `DRM.UpdateNeededButIncompatible` olan bir StatusEvent gönderir. Oynatıcının güncellenmesi için kullanıcı onayı gerekir. Uygulamanızda, kullanıcının oynatıcı güncellemesini kabul edip başlatabilmesine izin verecek bir arabirim sağlayın. StatusEvent olayını dinleyin ve aşağıdaki kodu çağırın:

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

Bu kod, oynatıcının güncellenmesini başlatır.

SystemUpdater sınıfı için ek olaylar Adobe Flash Platformu için ActionScript 3.0 Başvurusu'nda açıklanmıştır.

Oynatıcı güncellemesi tamamlandıktan sonra, kullanıcı güncellemenin başladığı sayfaya yeniden yönlendirilir. Adobe Access modülü indirilir ve akış oynamaya başlayabilir.

Bant Dışı Lisanslar

Flash Player 11 ve üstü, Adobe AIR 3.0 ve üstü

Lisanslar, `storeVoucher` yöntemini kullanılarak ve makbuzu (lisansı) diskte veya bellekte saklayarak bant dışı olarak (bir Adobe Access Lisans Sunucusu'na bağlanmadan) alınabilir.

İlgili çalışma zamanının, şifrelenmiş videoları Flash Player ve AIR'de oynatması için bu videonun DRM makbuzunu edinmesi gerekir. DRM makbuzu videonun şifre çözme anahtarını içerir ve müşterinin dağıttığı Adobe Access Lisans Sunucusu tarafından üretilir.

Flash Player/AIR çalışma zamanı bu makbuzu genellikle video DRM meta verilerinde (`DRMContentData` sınıfı) gösterilen Adobe Access Lisans Sunucusu'na bir makbuz isteği göndererek edinir. Flash/AIR uygulaması bu lisans isteğini `DRMManager.loadVoucher()` yöntemini çağırarak tetikleyebilir. Ya da, diskteki veya bellekteki içerik için lisans yoksa Flash Player/AIR çalışma zamanı, şifrelenmiş videonun kayıttan oynatımı başlangıcında lisansı otomatik olarak isteyebilir. Her iki durumda da, Flash/AIR uygulamasının performansı Adobe Access Lisans Sunucusu ile iletişimden etkilenir.

`DRMManager.storeVoucher()`, Flash/AIR uygulamasının, bant dışı edindiği DRM makbuzlarını Flash Player/AIR çalışma zamanına göndermesine izin verir. Sonra, çalışma zamanı lisans isteği işlemini atlayabilir ve şifrelenmiş videoları oynatmak için iletilmiş makbuzları kullanabilir. Buna rağmen, DRM makbuzu bant dışı olarak edinilmeden önce Adobe Access Lisans Sunucusu tarafından oluşturulmalıdır. Bununla birlikte, makbuzlarınızı genel kullanıma yönelik bir bir Adobe Access lisans sunucusu yerine herhangi bir HTTP sunucusu üzerinde barındırma seçeneğine de sahipsinizdir.

`DRMManager.storeVoucher()`, ayrıca birden çok cihaz arasında DRM makbuz paylaşımını desteklemek için de kullanılır. Bu özellik, Adobe Access 3.0'da "Etki Alanı Desteği" olarak geçer. Dağıtımınız bu kullanım durumunu destekliyorsa, çoklu makineleri `DRMManager.addToDeviceGroup()` yöntemini kullanarak bir aygıt grubuna kaydedebilirsiniz. Verilen bir içerik için geçerli bir etki alanına bağımlı makbuzu olan bir makine varsa, AIR uygulaması `DRMVoucher.toByteArray()` yöntemini kullanarak sıralı DRM makbuzlarını aktarabilir ve diğer makinelerinizde makbuzları `DRMManager.storeVoucher()` yöntemini kullanarak içe aktarabilirsiniz.

Aygıt kaydı

DRM makbuzları son kullanıcının makinesine bağlıdır. Bu nedenle Flash /AIR uygulamaları, doğru sıralanmış DRM makbuz nesnesine başvurmak için kullanıcı makinesinin benzersiz bir kimliğine gereksinim duyar. Aşağıdaki senaryo bir aygıt kayıt işlemini betimlemektedir:

Aşağıdaki işlemleri gerçekleştirdiğiniz varsayılırsa:

- Bir Adobe Access Sunucusu SDK'sını kurdunuz.
- Önceden oluşturulmuş lisansları edinmek için bir HTTP sunucusu kurdunuz.
- Korunan içeriği görüntülemek için bir Flash uygulaması oluşturdunuz.

Aygıt kayıt aşaması aşağıdaki eylemleri içerir:

1. Flash uygulaması rastgele oluşturulmuş bir kimlik meydana getirir.
2. Flash uygulaması `DRMManager.authenticate()` yöntemini çağırır. Uygulama, kimlik doğrulama isteğinde rastgele oluşturulmuş kimliği içermelidir. Örneğin, kullanıcı adı alanında kimliği içermelidir.
3. Adım 2'de bahsi geçen eylem, Adobe Access'in müşteri sunucusuna bir kimlik doğrulama isteği göndermesiyle sonuçlanır. Bu istek aygıt sertifikasını içerir.
   a. Sunucu, aygıt sertifikasını ve oluşturulan kimliği istek ve depolardan ayıklar.
   b. Müşteri alt sistemi bu aygıt sertifikası için lisanslar oluşturur, bunları depolar ve oluşturulan kimlik ile ilişkilendirerek bunlara erişim hakkı verir.
4. Sunucu, isteğe bir "başarı" mesajı ile karşılık verir.
5. Flash uygulaması oluşturulan kimliği yerel olarak bir Local Shared Object (LSO) (Yerel Paylaşılan Nesne) içinde depolar.

Aygıt kaydından sonra Flash uygulaması, oluşturulan kimliği aygıt kimliğini önceki şemada kullanmış olduğu şekilde kullanır.

1. Flash uygulaması oluşturulan kimliği LSO içinde bulmayı dener.
2. Oluşturulan kimlik bulunursa, Flash uygulaması oluşturulan kimliği önceden oluşturulmuş lisansları indirirken kullanır. Flash uygulaması `DRMManager.storeVoucher()` yöntemini kullanarak Adobe Access istemcisine tüketim için lisansları gönderir.
3. Oluşturulan kimlik bulunamazsa Flash uygulaması aygıt kayıt işlemini uygular.

Fabrika sıfırlaması

Aygıt kullanıcısı fabrika sıfırlama seçeneğini çağırdığında aygıt sertifikası temizlenir. Korunan içeriği yürütmeye devam etmek için Flash uygulamasının aygıt kayıt işleminden tekrar geçmesi gerekir. Eğer Flash uygulaması zaman aşımına uğramış bir önceden oluşturulmuş lisans gönderirse, Adobe Access istemcisi daha eski bir aygıt kimliği için şifrelenmiş olduğundan lisansı reddeder.

Etki alanı desteği

Flash Player 11 ve üstü, Adobe AIR 3.0 ve üstü

İçerik meta verisi, etki alanı kaydının gerekli olduğunu belirtirse, AIR uygulaması API'yi bir aygıt grubuna katılması için çağırabilir. Bu eylem, bir etki alanı kayıt isteğinin etki alanı sunucusuna gönderilmesini tetikler. Bir aygıt grubuna lisans verilir verilmez lisans, aygıt grubuna katılan diğer cihazlara dışa aktarılabilir ve bunlarla paylaşılabilir.

Sonrasında aygıt grubu bilgisi, daha sonra bir makbuzu başarılı bir şekilde almak ve tüketmek amacıyla istenen bilgiyi sunmak için kullanılacak olan `DRMContentData` öğesinin `VoucherAccessInfo` nesnesinde kullanılır.

Etki alanı desteğini kullanarak şifrelenmiş içeriği yürütme

Şifrelenmiş içeriği Adobe Access kullanarak oynatmak için, aşağıdaki adımları uygulayın:

1. `VoucherAccessInfo.deviceGroup` öğesini kullanarak aygıt grubu kaydının gerekli olup olmadığını kontrol edin.
2. Kimlik doğrulaması gerekli ise:
   - **a** Kimlik doğrulaması gerekliyse `DeviceGroupInfo.authenticationMethod` özelliğini bulmayı kullanın.
   - **b** Kimlik doğrulaması gerekliyse, kullanıcıyı aşağıdaki adımlardan BİR tanesini gerçekleştirerek doğrulayın:
     - Kullanıcının kullanıcı adı ve şifresini edinin. `DRMManager.authenticate(deviceGroup.serverURL, deviceGroup.domain, username, password)` öğesini çağırın.
     - Önbelleğe alınmış/önceden oluşturulmuş bir kimlik doğrulama simgesi edinin ve `DRMManager.setAuthenticationToken()` öğesini çağırın.
   - **c** `DRMManager.addToDeviceGroup()` öğesini çağırın.
3. Aşağıdaki görevlerden birini gerçekleştirerek içerik makbuzunu alın:
   - **a** `DRMManager.loadVoucher()` yöntemini kullanın.
   - **b** Makbuzu, aynı aygıt grubunda kayıtlı farklı bir aygıttan edinin. `DRMManager` öğesine makbuzu, `DRMManager.storeVoucher()` yöntemi ile sağlayın.
4. Şifrelenmiş içeriği `NetStream.play()` yöntemini kullanarak yürütün.

İçerik lisansını dışa aktarmak için, Adobe Access Lisans Sunucusu'ndan lisans edindikten sonra, cihazlardan herhangi biri `DRMVoucher.toByteArray()` yöntemini kullanarak lisans baytlarını ham halde sağlayabilir. İçerik sağlayıcılar tipik olarak bir aygıt grubundaki aygıt sayısını sınırlandırabilir. Sınıra ulaşılırsa, geçerli aygıtı kaydetmeden önce kullanılmamış bir aygıtta `DRMManager.removeFromDeviceGroup()` yöntemini çağırmanız gerekebilir.

Lisans önizlemesi

Flash uygulaması bir lisans önizlemesi talebi gönderebilir. Bu, uygulamanın kullanıcının makinesinin oynatım için gerekli olan kriterleri gerçekten karşılayıp karşılamadığını belirlemek için kullanıcıdan içeriği almasını istemeden önce bir önizleme işlemi gerçekleştirebileceği anlamına gelir. Lisans önizlemesi, müşterinin içeriği önizleyebilmesine (satın almaya karar vermeden önce içeriğin bir kısmını görüntüleme) karşılık lisansı önizleyebilmesi (lisansın hangi hakları sunduğunu görmesi) ile ilgilidir. Her bir makinede eşsiz olan bazı parametreler şunlardır: mevcut çıktılar ve bunların

korunma durumları, mevcut çalışma zamanı/DRM sürümü ile DRM istemci güvenliği düzeyidir. Lisans önizleme modu, çalışma zamanı/DRM istemcisine lisans sunucusu iş mantığını test etmesi ve kullanıcıya geri bildirim sağlaması için izin verir. Böylece müşteri bilinçli bir karar alabilir. Böylece müşteri geçerli bir lisansın nasıl olduğunu görür ancak içeriğin şifresini çözmek için anahtarı almaz. Lisans önizleme desteği isteğe bağlıdır ve yalnızca bu işlevi kullanan özel bir uygulamayı kullanıyorsanız gereklidir.

İçeriği sunma

Flash Player, ağ katmanını soyutlayıp korunan içeriği Adobe Access alt sistemine sağladığından Adobe Access içeriğin sunulma mekanizmasında belirsizdir. Bu nedenle, içerik HTTP, HTTP Dinamik Akış, RTMP veya RTMPE ile sunulabilir.

Ancak, Adobe Access uygulamasının içeriğin şifresini çözmek üzere bir lisans almasından önce, korunan içeriğin meta verisi gerekli olduğundan (genellikle '.metadata' dosyası biçiminde) bazı sorunlar yaşayabilirsiniz. Özellikle, RTMP/RTMPE protokolü ile yalnızca FLV ve F4V verileri Flash Media Server (FMS) aracılığıyla istemciye ulaştırılabilir. Bu nedenle müşteri, meta veri ikili büyük nesnesini başka yollarla almalıdır: Bu problemi çözme seçeneklerinden biri, meta verileri bir HTTP web sunucusunda barındırmak ve uygun meta verileri alma amacıyla arka planda yürütülen içeriğe bağlı olarak müşteri video oynatıcısını uygulamaktır.

```
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 (Açık Kaynak Medya Çerçevesi)

Open Source Media Framework (OSMF), kendi zengin medya deneyimlerinizi oluşturmanız sırasında size tam esneklik ve kontrol olanağı sunan ActionScript bazlı bir çerçevedir. OSMF ile ilgili daha fazla bilgi için OSMF Geliştirici Sitesi'ni ziyaret edin.

Korunan içeriği yürütme iş akışı

1 Bir `MediaPlayer` örneği oluşturun.

```
player = new MediaPlayer();
```

2 `MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE` olayını oynatıcıya kaydedin. Bu olay içerik DRM korumalı ise gönderilir.

```
player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE,
onDRMCapabilityChange);
```

3 Olay işleyicide `DRMTrait` örneğini edinin. `DRMTrait`, `authenticate()` gibi DRM ile ilişkili yöntemleri çağırdığınız arabirimdir. DRM korumalı bir içeriği yüklerken OSMF, DRM doğrulama işlemlerini gerçekleştirir ve durum olayları gönderir. `DRMTrait` öğesine bir `DRMEvent.DRM_STATE_CHANGE` olay işleyicisi ekleyin.

```
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 DRM olaylarını `onDRMStateChange()` yönteminde işleyin.

```
        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;
        }
    }
```

Bölüm 28: AIR'ye PDF içeriği ekleme

Adobe AIR 1.0 ve üstü

Adobe® AIR®'de çalışan uygulamalar yalnızca SWF ve HTML içeriği değil, PDF içeriği de oluşturabilir. AIR uygulamaları, HTMLLoader sınıfını, WebKit motorunu ve Adobe® Reader® tarayıcı eklentisini kullanarak PDF içeriği oluşturur. AIR uygulamasında, PDF içeriği uygulamanızın tam yükseklik ve genişliğine göre veya alternatif olarak arabiriminin bir parçası olarak uzayabilir. Adobe Reader tarayıcı eklentisi, AIR uygulamasında PDF dosyalarının görüntülenmesini kontrol eder. Okuyucu araç menüsü arabiriminde yapılan değişiklikler (konum, sabitleme ve görünürlük), hem AIR uygulamalarındaki hem de tarayıcıdaki PDF dosyalarının daha sonra görüntülenmesini devam ettirir

Önemli: *Kullanıcı, AIR uygulamasında PDF içeriği oluşturmak için Adobe Reader ya da Adobe® Acrobat® 8.1 veya daha yeni bir sürümü yüklemiş olmalıdır.*

PDF Özelliklerini Saptama

Adobe AIR 1.0 ve üstü

Kullanıcı Adobe Reader veya Adobe Acrobat 8.1 ya da daha yeni bir sürüme sahip değilse, PDF içeriği AIR uygulamasında görüntülenmez. Kullanıcının PDF içeriği oluşturup oluşturamayacağını saptamak için ilk önce `HTMLLoader.pdfCapability` özelliğini kontrol edin. Bu özellik HTMLPDFCapability sınıfının aşağıdaki sabitlerinden biri olarak ayarlanır:

Sabit	Açıklama
HTMLPDFCapability.STATUS_OK	Yeterli bir Adobe Reader sürümü (8.1 veya daha yeni) saptandı ve PDF içeriği bir HTMLLoader nesnesine yüklenebilir.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	Herhangi bir Adobe Reader sürümü saptanmadı. Bir HTMLLoader nesnesi PDF içeriğini görüntüleyemez.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Adobe Reader saptandı, ancak sürüm çok eski. Bir HTMLLoader nesnesi PDF içeriğini görüntüleyemez.
HTMLPDFCapability.ERROR_PREFERRED_READER_TOO_OLD	Yeterli bir Adobe Reader sürümü (8.1 veya daha yeni) saptandı, ancak Adobe Reader'ın PDF içeriğini işlemek üzere ayarlanan sürümü Reader 8.1'den eski. Bir HTMLLoader nesnesi PDF içeriğini görüntüleyemez.

Windows'ta, kullanıcının sisteminde Adobe Acrobat veya Adobe Reader 7.x ya da daha yeni bir sürüm çalışmaktaysa, PDF yüklemeyi destekleyen daha yeni bir sürüm yüklenmiş olsa bile bu sürüm kullanılır. Bu durumda, `pdfCampability` özelliğinin değeri `HTMLPDFCapability.STATUS_OK` ise, AIR uygulaması PDF içeriğini yüklemeye çalıştığında daha eski Acrobat veya Reader sürümü bir uyarı görüntüler (ve AIR uygulamasında istisna verilmez). Bu son kullanıcılarınız için olası bir durumsa, onlara uygulamanızı çalıştırdıkları sırada Acrobat'ı kapatmalarına ilişkin talimatlar vermeyi deneyebilirsiniz. PDF içeriği kabul edilebilir bir zaman diliminde yüklenmiyorsa, bu talimatları görüntülemek isteyebilirsiniz.

Linux'ta, AIR kullanıcının dışa aktardığı PATH içinde (acroread komutunu içeriyorsa) ve /opt/Adobe/Reader dizininde Adobe Reader'ı arar.

Aşağıdaki kod bir kullanıcının bir AIR uygulamasında PDF içeriği görüntüleyebilip görüntülemeyeceğini algılar. Kullanıcı PDF görüntüleyemezse kod, HTMLPDFCapability hata nesnesine denk gelen hata kodunu izler.

```
 if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

PDF içeriği yükleme

Adobe AIR 1.0 ve üstü

Bir HTMLLoader örneği oluşturarak, bu örneğin boyutlarını ayarlayarak ve PDF yolunu yükleyerek AIR uygulamasına bir PDF ekleyebilirsiniz.

Aşağıdaki örnek, harici bir siteden PDF yükler. URLRequest öğesini kullanılabilir harici bir PDF'ye giden yolla değiştirin.

```
 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);
```

app ve app-storage gibi dosya URL'leri ve AIR'ye özgü URL şemalarından da içerik yükleyebilirsiniz. Örneğin, aşağıdaki kod uygulama dizininin PDF alt dizinindeki test.pdf dosyasını yükler:

app:/js_api_reference.pdf

AIR URL şemalarıyla ilgili daha fazla bilgi için bkz. “URI şemaları” sayfa 822.

Komut dosyası PDF içeriği

Adobe AIR 1.0 ve üstü

PDF içeriğini tarayıcıdaki web sayfasındaki gibi kontrol etmek için JavaScript'i kullanabilirsiniz.

Acrobat'a ilişkin JavaScript geliştirmeleri aşağıdaki ek özellikleri sunar:

- Sayfa gezinmesini denetleme ve büyütme
- Belge içinde formları işleme
- Multimedya olaylarını denetleme

Adobe Acrobat'a yönelik JavaScript geliştirmelerine ilişkin tüm ayrıntılar http://www.adobe.com/devnet/acrobat/javascript.html adreslerindeki Adobe Acrobat Geliştirici Bağlantısı'nda sunulmuştur.

HTML-PDF iletişim temelleri

Adobe AIR 1.0 ve üstü

HTML sayfasındaki JavaScript, PDF içeriğini temsil eden DOM nesnesinin `postMessage()` yöntemini çağırarak PDF içeriğindeki JavaScript'e mesaj gönderebilir. Örneğin, aşağıdaki gömülü PDF içeriğini dikkate alın:

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

Mevcut HTML içeriğinde aşağıdaki JavaScript kodu, PDF dosyasındaki JavaScript'e bir mesaj gönderir:

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

PDF dosyası, bu mesajın alınması için JavaScript'i dahil edebilir. Belge, klasör, sayfa, alan düzeyi ve toplu düzey bağlamları dahil bazı bağlamlardaki PDF dosyalarına JavaScript kodu ekleyebilirsiniz. Burada yalnızca, PDF dokümanı açıldığında değerlendirilen komut dosyalarını tanımlayan belge düzeyi bağlamı tartışılır.

PDF dosyası `hostContainer` nesnesine `messageHandler` özelliğini ekleyebilir. `messageHandler` özelliği, mesajları yanıtlamak için işleyici işlevlerini tanımlayan bir nesnedir. Örneğin aşağıdaki kod, PDF dosyası tarafından ana bilgisayar kabından (PDF dosyasını içeren HTML içeriği) alınan mesajları işlemeye yönelik işlevi tanımlar:

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

HTML sayfasındaki JavaScript kodu, sayfada yer alan PDF nesnesinin `postMessage()` yöntemini çağırabilir. Bu yöntemin çağrılması, PDF dosyasındaki belge düzeyinde JavaScript'e bir mesaj (`"Hello from HTML"`) gönderir:

```
<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>
```

Daha gelişmiş bir örnek için ve bir PDF dosyasına JavaScript eklemek üzere Acrobat 8 kullanımıyla ilgili bilgi almak için bkz. Adobe AIR uygulamasında PDF içeriğinde çapraz komut dosyası oluşturma.

ActionScript'ten PDF içeriği için komut dosyası

Adobe AIR 1.0 ve üstü

ActionScript kodu (SWF içeriğindeki), PDF içeriğindeki JavaScript ile doğrudan iletişim kuramaz. Ancak, ActionScript PDF içeriğini yükleyen bir HTMLLoader nesnesinde yüklenen HTML sayfasındaki JavaScript ile iletişim kurabilir ve bu JavaScript kodu, yüklenen PDF sayfasındaki JavaScript ile iletişim kurabilir. Daha fazla bilgi için bkz. "AIR uygulamasında HTML ve JavaScript programlama" sayfa 991.

AIR'de PDF içeriği için bilinen sınırlamalar

Adobe AIR 1.0 ve üstü

Adobe AIR'deki PDF içeriği aşağıdaki sınırlamaları içerir:

- PDF içeriği saydam (`transparent` özelliğinin `true` olarak ayarlandığı) bir pencerede (bir NativeWindow nesnesi) görüntülenemez.
- Bir PDF dosyasının görüntü düzeni, AIR uygulamasındaki diğer display nesnelerinden farklı çalışır. PDF içeriği HTML görüntü düzenine göre doğru biçimde kesilse de, her zaman AIR uygulamasının görüntü düzenindeki içeriğin en üstünde bulunacaktır.

- PDF belgesi içeren HTMLLoader nesnesinin belirli görsel özellikleri değiştirildiğinde, PDF belgesi görünür duruma gelir. Bu özellikler `filters`, `alpha`, `rotation`, ve `scaling` özelliklerini içerir. Bu özelliklerin değiştirilmesi PDF dosyasını, özellikler sıfırlanana kadar görünmez duruma getirir. Bu durum, HTMLLoader nesnesi içeren display nesne kabının özelliklerini değiştirdiğinizde de geçerlidir.
- PDF içeriği yalnızca PDF içeriğine sahip NativeWindow nesnesinin Stage nesnesine ilişkin `scaleMode` özelliği `StageScaleMode.NO_SCALE` olarak ayarlandığında görünür. Bunun dışında herhangi bir değere ayarlandığında, PDF içeriği görünmez.
- PDF dosyası içindeki içerik bağlarının tıklatılması, PDF içeriğinin kaydırma konumunu günceller. PDF dosyasının dışındaki içerik bağlarının tıklatılması, PDF içeren HTMLLoader nesnesini yeniden yönlendirir (bağın hedefi yeni bir pencere olsa da).
- PDF yorum oluşturma iş akışları AIR'de çalışmaz.

Bölüm 29: Kullanıcı etkileşiminin temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız kullanıcı etkinliğine yanıt vermek için ActionScript 3.0 kullanarak etkileşim oluşturabilir. Bu bölüm, ActionScript 3.0 olay modelini zaten bildiğinizi varsaymaktadır. Daha fazla bilgi için "Olayları işleme" sayfa 126 bölümünü inceleyin.

Kullanıcı girdisini yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İster klavye isterse fare ile ya da her ikisi ile kurulan kullanıcı etkileşimi, etkileşimin temelini oluşturur. ActionScript 3.0 içerisinde kullanıcı etkileşimini tanımlamak ve yanıt vermek öncelikle bu olayları dinlemeye dayanır.

DisplayObject sınıfının bir alt sınıfı olan InteractiveObject sınıfı kullanıcı etkileşimini işlemek için gerekli olan yaygın olay yapısını ve işlevselliğini sağlar. Doğrudan bir InteractiveObject sınıfı örneği oluşturmazsınız. Bunun yerine SimpleButton, Sprite, TextField gibi görüntüleme nesneleri ve çeşitli Flash geliştirme aracı ve Flex bileşenleri, kullanıcı etkileşimi modelini bu sınıftan miras alır ve bu nedenle ortak bir yapıya sahiptir. Bu da InteractiveObject öğesinden derlenmiş olan bir nesnede kullanıcı etkileşimini işlemek için öğrendiğiniz tekniklerin ve yazdığınız kodun diğerlerine de uygulanabileceği anlamına gelir.

Önemli kavramlar ve terimler

Devam etmeden önce aşağıdaki önemli kullanıcı etkileşimi terimlerini öğrenmeniz gereklidir:

Karakter kodu Geçerli karakter kümesinde bir karakteri temsil eden sayısal kod (klavyede basılan bir tuş ile ilişkilendirilmiş). Örneğin, "D" ve "d" karakterleri İngilizce bir klavyede aynı tuşa basılarak oluşturulmalarına rağmen karakter kodları farklıdır.

Bağlam menüsü Bir kullanıcı sağ tıklattığında veya belirli bir klavye-fare birleşimini kullandığında görünen menü. Bağlam menüsü komutları seçilen eylemleri uygular. Örneğin, bir görüntünün bağlam menüsü görüntüyü farklı bir pencerede açma komutu ile görüntüyü indirme komutu içerebilir.

Odak Seçili öğenin etkin ve klavye ya da fare etkileşiminin hedefi olduğunun göstergesidir.

Tuş kodu Klavyedeki fiziksel bir tuşa karşılık gelen sayısal kod.

Daha fazla Yardım konusu

InteractiveObject

Keyboard

Mouse

ContextMenu

Odak yönetme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Etkileşimli bir nesne, programlama veya kullanıcı eylemi ile odak alabilir. Ayrıca, `tabEnabled` özelliği `true` olarak ayarlanmışsa kullanıcı Sekme tuşu ile odağı bir nesneden diğerine geçirebilir. Aşağıdaki durumlar haricinde `tabEnabled` değerinin varsayılan olarak `false` şeklinde olduğuna dikkat edin:

- Bir SimpleButton nesnesi için değer `true` olur.
- Bir giriş metni alanı için değer `true` olur.
- `buttonMode` değeri `true` olan bir Sprite veya MovieClip nesnesi için değer `true` olur.

Her iki durumda da odak değiştiğinde ek davranış sağlamak için `FocusEvent.FOCUS_IN` veya `FocusEvent.FOCUS_OUT` öğelerine dinleyici ekleyebilirsiniz. Bu durum özellikle metin alanları ve formlar için kullanışlıdır ancak InteractiveObject sınıfından miras alan hareketli grafikler, film klipleri veya diğer nesneler için de kullanılabilir. Aşağıdaki örnek Sekme tuşu ile odak değiştirmenin nasıl etkinleştirileceğini ve sonra gelen odak olayına nasıl yanıt verileceğini gösterir. Bu durumda, karelerin her biri odak haline geldiğinde renk değiştirir.

Not: *Flash Professional odak yönetme için klavye kısayolları kullanır ve bu nedenle uygun odak simülasyonu için SWF dosyaları Flash içerisinde değil de bir tarayıcı veya AIR ile test edilmelidir.*

```
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);
}
```

Girdi türlerini keşfetme

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Flash Player 10.1 ve Adobe AIR 2 sürümleri, çalışma zamanı ortamlarını belli girdi türlerinin desteklenmesi açısından test edebilmeyi sağlamıştır. Çalışma zamanının dağıtılmış olduğu aygıtın açık olup olmadığını sınamak için ActionScript'i kullanabilirsiniz:

- Stilus ile veya parmakla girişi (veya hiç dokunmadan girişi) destekler.
- Kullanıcı için sanal veya fiziksel klavyeye sahiptir (veya klavyesizdir).
- Bir imleç görüntüler (görüntülemiyorsa, imleci bir nesnenin üzerinde bekletmeye bağlı özellikler çalışmaz)

Girdi keşfetme ActionScript API'leri şunları içerir:

- flash.system.Capabilities.touchscreenType özelliği: Geçerli ortamda hangi girdi türünün desteklendiğini göstermek üzere çalışma zamanında sağlanan bir değer.
- flash.system.TouchScreenType sınıfı: Capabilities.touchScreenType özelliği için bir sayım değeri sabitleri sınıfı.
- flash.ui.Mouse.supportsCursor özelliği: Sürekli bir imlecin bulunup bulunmadığına ilişkin çalışma zamanında sağlanan bir değer.
- flash.ui.Keyboard.physicalKeyboardType özelliği: Tam fiziksel bir klavyenin veya sayısal bir tuş takımının olup olmadığını ya da hiç klavye olmadığını belirtmek üzere çalışma zamanında sağlanan bir değer.
- flash.ui.KeyboardType sınıfı: flash.ui.Keyboard.physicalKeyboardType özelliği için bir sayım değeri sabitleri sınıfı.
- flash.ui.Keyboard.hasVirtualKeyboard özelliği: Kullanıcıya bir sanal klavye (fiziksel klavyenin yerine veya fiziksel klavyeye ek olarak) sağlanıp sağlanmadığını belirtmek üzere çalışma zamanında sağlanan bir değer.

Girdi keşfetme API'leri, bir kullanıcının aygıt yetkilerinden faydalanabilmenize imkan verir veya bu yetkiler mevcut olmadığında alternatifler sunar. Bu API'ler özellikle mobil ve dokunmatik uygulamalar geliştirmek için faydalıdır. Örneğin, stilus için küçük düğmeleri bulunan bir mobil aygıt arabirimine sahipseniz, kullanıcının parmakla giriş yapabileceği daha büyük düğmelere sahip alternatif bir arabirim sağlayabilirsiniz. Aşağıdaki kod, stilus etkileşimine uygun bir kullanıcı arabirimi öğesi grubu atayan createStylusUI() adlı işleve sahip bir uygulama içindir. createTouchUI() adı verilen bir diğer işlev ise parmak etkileşimine uygun başka bir kullanıcı arabirimi öğesi grubu atar:

```
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();
}
```

Farklı girdi ortamları için uygulamalar geliştirirken aşağıdaki uyumluluk çizelgesini göz önünde bulundurun:

Ortam	**supportsCursor**	**touchscreenType == FINGER**	**touchscreenType == STYLUS**	**touchscreenType == NONE**
Klasik Masaüstü	true	false	false	true
Kapasitif Dokunmatik Ekranlı Aygıtlar (tabletler, PDA'lar ve Apple iPhone ya da Palm Pre gibi hafif dokunuşları algılayan telefonlar)	false	true	false	false
Dirençli Dokunmatik Ekranlı aygıtlar (tabletler, PDA'lar ve HTC Fuze gibi hassas, yüksek basınçlı teması algılayan telefonlar)	false	false	true	false
Dokunmatik Olmayan Ekranlı aygıtlar (uygulamaları çalıştırıp teması algılayan ekranlara sahip olmayan, özellikli telefonlar ve aygıtlar)	false	false	false	true

Not: *Farklı aygıt platformları çok sayıda girdi türü kombinasyonunu destekleyebilir. Bu çizelgeyi genel bir kılavuz olarak kullanın.*

Bölüm 30: Klavye girdisi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız klavye girdisini yakalayabilir, onu yanıtlayabilir ve bir IME'yi kullanıcıların ASCII olmayan metin karakterlerini çok bitli dillerde yazmalarına olanak vermek için değiştirebilir. Bu bölüm, ActionScript 3.0 olay modelini zaten bildiğinizi varsaymaktadır. Daha fazla bilgi için "Olayları işleme" sayfa 126 bölümünü inceleyin.

Çalışma zamanı boyunca hangi klavye desteği türlerinin (fiziksel, sanal, alfa nümerik veya 12 düğmeli sayısal) mevcut olduğunu keşfetmeyle ilgili bilgi edinmek için bkz. "Girdi türlerini keşfetme" sayfa 564.

Bir Giriş Yöntemi Düzenleyicisi (IME), kullanıcıların standart klavye kullanarak karmaşık karakterler ve semboller yazmalarına izin verir. IME sınıflarını, kullanıcıların sizin uygulamanızda sistem IME'lerinden faydalanmalarını sağlamak için kullanabilirsiniz.

Daha fazla Yardım konusu

flash.events.KeyboardEvent

flash.system.IME

Klavye girdisini yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Etkileşim modelini InteractiveObject sınıfından miras alan görüntüleme nesneleri olay dinleyici kullanarak klavye olaylarına yanıt verebilir. Mesela Sahne alanına klavye girişini dinleyecek ve yanıt verecek bir olay dinleyici yerleştirebilirsiniz. Aşağıdaki kodda bir olay dinleyici basılan bir tuşu yakalar ve tuş adı ile tuş kod özellikleri görüntülenir:

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " +
event.charCode + ")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Ctrl tuşu gibi bazı tuşlar herhangi bir glif temsil etmiyor olsa da olay oluştururlar.

Bir önceki kod örneğinde, klavye olay dinleyicisi Sahne alanının tümü için klavye girişi yakalamıştı. Ayrıca Sahne alanı üzerindeki belirli bir görüntüleme nesnesi için bir olay dinleyicisi yazabilirsiniz; bu olay dinleyicisi nesne odaklı olduğunda tetiklenir.

Aşağıdaki örnekte tuş vuruşları sadece kullanıcı TextField örneği içerisine yazdığı zaman Çıktı paneline yansıtılmaktadır. Shift tuşu basılı tutulduğunda TextField öğesinin kenarlık rengi geçici olarak kırmızıya dönüşür.

Bu kod Sahne alanında `tf` adlı bir TextField örneği olduğunu varsayar.

```
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;
    }
}
```

TextField sınıfı ayrıca bir kullanıcı metin girdiğinde dinleyebileceğiniz bir `textInput` olayı rapor eder. Daha fazla bilgi için "Metin girdisi yakalama" sayfa 381 bölümünü inceleyin.

Not: *AIR çalışma zamanında, bir klavye olayı iptal edilebilir. Flash Player çalışma zamanında, bir klavye olayı iptal edilemez.*

Tuş kodları ve karakter kodları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Basılan tuşu belirlemek ve diğer eylemleri tetiklemek için bir klavye olayının `keyCode` ve `charCode` özelliklerine erişebilirsiniz. `keyCode` özelliği klavyedeki bir tuşun değerine karşılık gelen sayısal değerdir. `charCode` özelliği ise o tuşun geçerli karakter kümesi içindeki sayısal değeridir. (Varsayılan karakter kümesi ASCII destekleyen UTF-8'dir.)

Tuş kodu ve karakter değeri arasındaki temel fark bir tuş kodu değerinin klavye üzerindeki belirli bir tuşu temsil etmesi (sayısal tuş takımındaki 1 harf tuşlarının üst sırasındaki 1'den farklıdır ancak "1"i çıkaran tuş ve "!" çıkaran tuş aynıdır) ve karakter değerinin belirli bir karakteri temsil etmesidir. (R ve r karakterleri birbirinden farklıdır.)

Not: *ASCII'da anahtarlar ve karakter kod değerleri arasındaki eşlemeler için, Adobe Flash Platformu için ActionScript 3.0 Başvurusu'ndaki flash.ui.Keyboard sınıfına bakın.*

Tuşlar ve tuş kodları arasındaki eşlemeler cihaza ve işletim sistemine bağlıdır. Bu nedenle eylemleri tetiklemek için tuş eşlemeleri kullanmamanız gerekir. Onun yerine uygun `keyCode` özelliklerine başvurmak için Keyboard sınıfı tarafından sağlanan önceden tanımlı sabit değerleri kullanmanız gerekir. Örneğin, Shift tuşu için tuş eşleme kullanmak yerine `Keyboard.SHIFT` sabitini kullanın (yukarıdaki kod örneğinde gösterildiği gibi).

KeyboardEvent önceliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Diğer olaylarda olduğu gibi klavye olay sırası kod içindeki `addEventListener()` yöntemlerinin atanma sırasına göre değil görüntüleme nesnesi hiyerarşisine göre belirlenir.

Aşağıdaki örnekte olduğu gibi `container` adlı bir film klibinin içerisine `tf` adlı bir metin alanı yerleştirdiğinizi ve her iki örneğe bir klavye olayı için bir olay dinleyicisi eklediğinizi düşünün:

```
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 + ")");
}
```

Hem metin alanında hem de üst kabında bir dinleyici olduğundan TextField içerisindeki her tuş vuruşu için `reportKeyDown()` işlevi iki kez çağrılır. Her basılan tuş için `container` film klibinden önce metin alanının bir olay gönderdiğine dikkat edin.

İşletim sistemi ve web tarayıcı klavye olaylarını Adobe Flash Player ve AIR uygulamalarından önce işleyecektir. Örneğin, Microsoft Internet Explorer uygulamasında Ctrl+W tuş bileşimi herhangi bir SWF dosyası bir klavye olayı göndermeden tarayıcı penceresini kapatacaktır.

IME sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

IME sınıfı, Flash Player veya Adobe AIR uygulamasında, işletim sisteminin IME'sini işlemenize olanak sağlar.

ActionScript'i kullanarak şunları belirleyebilirsiniz:

- Kullanıcının bilgisayarında bir IME yüklü olup olmadığı (`Capabilities.hasIME`)
- Kullanıcının bilgisayarında IME'nin etkinleştirilmiş mi yoksa devre dışı mı bırakılmış olduğu (`IME.enabled`)
- Geçerli IME'nin kullandığı dönüştürme modu (`IME.conversionMode`)

Girdi metni alanını belirli bir IME bağlamıyla ilişkilendirebilirsiniz. Girdi alanları arasında geçiş yaptığınızda, Hiragana (Japonca), tam genişlikte sayılar, yarım genişlikte sayılar, doğrudan girdi, vb. arasında da IME'yi değiştirebilirsiniz.

IME, kullanıcıların Çince, Japonca ve Korece gibi çok baytlı dillerde ASCII olmayan metin karakterlerini yazmasına olanak sağlar.

IME'lerle çalışma hakkında daha fazla bilgi için, uygulamanızı geliştirdiğiniz işletim sisteminin belgelerine bakın. Ek kaynaklar için, aşağıdaki web sitelerine bakın:

- http://www.msdn.microsoft.com/goglobal/
- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

Not: *Bir IME, kullanıcının bilgisayarında etkin değilse, `Capabilities.hasIME` dışındaki IME yöntemlerine veya özelliklerine yapılan çağrılar başarısız olur. Bir IME'yi elle etkinleştirdikten sonra, ardından IME yöntemlerine ve özelliklerine yapılan ActionScript çağrıları beklenen şekilde gerçekleşir. Örneğin, Japonca IME kullanıyorsanız, herhangi bir IME yöntemini veya özelliğini çağırabilmeniz için öncelikle bu IME'yi etkinleştirmeniz gerekir.*

IME'nin yüklü ve etkinleştirilmiş olup olmadığını kontrol etme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Herhangi bir IME yöntemini veya özelliğini çağırmadan önce, kullanıcının bilgisayarında geçerli olarak IME'nin yüklü ve etkinleştirilmiş olup olmadığını kontrol etmeniz gerekir. Aşağıdaki kod, siz herhangi bir yöntem çağırmadan önce kullanıcıda IME yüklü ve etkinleştirilmiş olup olmadığının nasıl kontrol edildiğini gösterir:

```
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.");
}
```

Önceki kod `Capabilities.hasIME` özelliğini kullanarak, kullanıcının bir IME yüklemiş olup olmadığını kontrol eder. Bu özellik `true` değerine ayarlanırsa, kod, `IME.enabled` özelliğini kullanarak, kullanıcının IME'sinin geçerli olarak etkinleştirilmiş olup olmadığını kontrol eder.

Geçerli olarak hangi IME dönüştürme modunun etkinleştirilmiş olduğunu belirleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çok dilli uygulamalar oluşturduğunuzda, geçerli olarak kullanıcıda hangi dönüştürme modunun etkin olduğunu belirlemeniz gerekebilir. Aşağıdaki kod, kullanıcıda IME yüklü olup olmadığının ve IME yüklüyse, hangi IME dönüştürme modunun geçerli olarak etkin olduğunun nasıl kontrol edildiğini gösterir:

```
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.";
}
```

Önceki kod ilk olarak kullanıcıda IME yüklü olup olmadığını kontrol eder. Daha sonra `IME.conversionMode` özelliğini, IMEConversionMode sınıfındaki sabitlerin her biriyle karşılaştırarak geçerli IME'nin hangi dönüştürme modunu kullandığını kontrol eder.

IME dönüştürme modunu ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcının IME'sinin dönüştürme modunu değiştirdiğinizde, kodun bir `try..catch` bloğu içinde sarıldığından emin olmanız gerekir, bunun nedeni, IME dönüştürme modunu ayarlayamazsa, `conversionMode` özelliği kullanılarak dönüştürme modu ayarlandığında bir hata atılabilmesidir. Aşağıdaki kod, `try..catch` bloğunun, `IME.conversionMode` özelliği ayarlanırken nasıl kullanıldığını gösterir:

```
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;
    }
}
```

Önceki kod ilk önce, kullanıcıya durum mesajı görüntülemek için kullanılan bir metin alanını oluşturur. Daha sonra, IME yüklüyse, kod IME'yi etkinleştirir ve dönüştürme modunu Korece olarak ayarlar. Kullanıcının bilgisayarında Korece IME yükü değilse, Flash Player veya AIR tarafından bir hata atılır ve bu hata `try..catch` bloğu tarafından yakalanır. `try..catch` bloğu, önceden oluşturulan metin alanında hata mesajını görüntüler.

Belirli metin alanları için IME'yi devre dışı bırakma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazı durumlarda, kullanıcılar karakterleri yazarken kullanıcıların IME'sini devre dışı bırakmak isteyebilirsiniz. Örneğin, yalnızca sayısal girdiyi kabul eden bir metin alanınız varsa, IME'nin gelip veri girişini yavaşlatmasını istemeyebilirsiniz.

Aşağıdaki örnek, `FocusEvent.FOCUS_IN` ve `FocusEvent.FOCUS_OUT` olaylarını nasıl dinleyip buna göre kullanıcının IME'sini devre dışı bırakabileceğinizi göstermektedir:

```
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;
    }
}
```

Bu örnek, `phoneTxt` ve `nameTxt` adında iki girdi metni alanı oluşturur ve sonra `phoneTxt` metin alanına iki olay dinleyicisi ekler. Kullanıcı odağı `phoneTxt` metin alanına ayarladığında, bir `FocusEvent.FOCUS_IN` olayı gönderilir ve IME devre dışı bırakılır. Odak `phoneTxt` metin alanından çıktığında, IME'yi yeniden etkinleştirmek için `FocusEvent.FOCUS_OUT` olayı gönderilir.

IME oluşturma olaylarını dinleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

IME oluşturma olayları, bir oluşturma dizesi ayarlandığında gönderilir. Örneğin, kullanıcının IME'si etkinleştirilmişse ve aktif durumdaysa, kullanıcı Japonca bir dize yazdığında, oluşturma dizesini seçtiği anda `IMEEvent.IME_COMPOSITION` olayı gönderilir. `IMEEvent.IME_COMPOSITION` olayını dinlemek için, aşağıdaki örnekte gösterildiği gibi, System sınıfında (`flash.system.System.ime.addEventListener(...)`) statik `ime` özelliğine bir olay dinleyicisi eklemeniz gerekir:

```
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;
}
```

Önceki kod iki metin alanı oluşturur ve bu metin alanlarını görüntüleme listesine ekler. Birinci metin alanı olan `inputTxt`, kullanıcının Japonca metin girmesine olanak sağlayan bir girdi metni alanıdır. İkinci metin alanı olan `outputTxt`, kullanıcıya hata mesajları görüntüleyen veya kullanıcının `inputTxt` metin alanına yazdığı Japonca dizeyi yankılayan bir dinamik metin alanıdır.

Sanal klavyeler

Flash Player 10.2 ve üstü, AIR 2.6 ve üstü

Telefonlar ve tabletler gibi mobil aygıtlar, fiziksel bir klavye yerine genellikle sanal, yazılım klavyesi sağlar. Flash API'sindeki sınıflar şunları gerçekleştirmenize izin verir:

- Sanal klavyenin yükseltildiğini ve kapatıldığında algılama.
- Klavyenin yükselmesini engelleme.

- Sahne alanının sanal klavye tarafından gizlenen alanını belirleme.
- Odak kazandığında klavyeyi yükselten etkileşimli nesneler oluşturma. (iOS'deki AIR uygulamaları tarafından desteklenmez.)
- (Yalnızca AIR) Uygulamanızın klavyeye uyum sağlamak üzere kendi görüntüsünü değiştirmesini sağlamak için otomatik kaydırma hareketini devre dışı bırakın.

Sanal klavye davranışı denetleme

Kullanıcı bir metin alanına veya özel olarak konfigüre edilmiş etkileşimli nesneye dokunduğunda çalışma zamanı otomatik olarak sanal klavyeyi açar. Klavye açıldığında, kullanıcının yazım sırasında metni görmesini sağlamak için çalışma zamanı uygulama içeriğini kaydırırken ve yeniden boyutlandırırken yerel platform kurallarını izler.

Klavye açıldığında odaklanılan nesne sırayla aşağıdaki olayları gönderir:

`softKeyboardActivating` olayı: klavye sahne alanına yükselmeye başlamadan hemen önce gönderilir. Gönderilen olay nesnesinin `preventDefault()` yöntemini çağırmazsanız sanal klavye açılmaz.

`softKeyboardActivate` olayı: `softKeyboardActivating` olay işlemesi tamamlandıktan sonra gönderilir. Odaklanılan nesne bu olayı gönderdiğinde Sahne Alanı nesnesinin `softKeyboardRect` özelliği sahne alanının sanal klavye tarafından gizlenen alanını yansıtmak için güncelleştirildi. Bu olay iptal edilemez.

Not: *Örneğin, kullanıcı klavye türünü değiştirdiğinde klavyenin boyutu değişirse, odaklanılan nesne ikinci bir softKeyboardActivate olayı gönderir.*

`softKeyboardDeactivate` olayı: sanal klavye herhangi bir neden dolayı kapatıldığında gönderilir. Bu olay iptal edilemez.

Aşağıdaki örnek, sahne alanı üzerine iki TextField nesnesi ekler. Üst TextField alanı alana dokunduğunuzda klavyenin yükselmesini engeller ve zaten yükselmişse klavyeyi kapatır. Alt TextField varsayılan davranışı gösterir. Örnek iki metin alanı tarafından gönderilen yazılım klavyesi olaylarını raporlar.

```
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 );
    }
}
}
```

Etkileşimli nesneler için sanal klavye desteği ekleme

Flash Player 10.2 ve üstü, AIR 2.6 ve üstü (ancak iOS'de desteklenmez)

Normalde sanal klavye yalnızca bir TextField alanına dokunulduğunda açılır. Sanal klavyeyi odak kazandığında açmak için InteractiveObject sınıfının bir örneğini konfigüre edebilirsiniz.

Yazılım klavyesini açma amacıyla bir InteractiveObject örneğini konfigüre etmek için `needsSoftKeyboard` özelliğini `true` olarak ayarlayın. Nesne sahne alanı odak özelliğine her atandığında yazılım klavyesi otomatik olarak açılır. Buna ek olarak, InteractiveObject nesnesinin `requestSoftKeyboard()` yöntemini çağırarak klavyeyi yükseltebilirsiniz.

Aşağıdaki örnek bir InteractiveObject nesnesinin metin giriş alanı olarak davranması için nasıl programlanabileceğini gösterir. Örnekte gösterilen TextInput sınıfı, klavyenin gerektiğinde yükselmesini sağlamak için `needsSoftKeyboard` özelliğini ayarlar. Ardından, nesne `keyDown` olaylarını dinler ve yazılan karakteri alana girer.

Örnek, yazılan metni eklemek ve göstermek için Flash metin motorunu kullanır ve bazı önemli olayları işler. Kolaylık sağlamak için örnek tam özellikli bir metin alanı uygulamaz.

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

Aşağıdaki ana uygulama sınıfı, TextInput sınıfının nasıl kullanılacağını ve klavye yükseldiğinde veya aygıt yönlendirmesi değiştiğinde uygulama mizanpajının nasıl yönetileceğini gösterir. Ana sınıf bir TextInput nesnesi oluşturur ve sınırlarını sahne alanını dolduracak şekilde ayarlar. Sınıf, yazılım klavyesi yükseltildiğinde veya sahne alanı boyut değiştirdiğinde TextInput nesnesinin boyutunu ayarlar. Sınıf, TextInput nesnesinden gelecek yazılım klavyesi olaylarını dinler ve sahne alanındaki olayları yeniden boyutlandırır. Olayın nedeninden bağımsız olarak uygulama sahne alanının görülebilir alanını belirler ve bunu doldurmak için girdi denetimini yeniden boyutlandırır. Doğal olarak, gerçek bir uygulamada, daha karmaşık bir mizanpaj algoritması gerekir.

```
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;
        }
    }
}
```

Not: *Sahne alanı yalnızca* `scaleMode` *özelliği* `noScale` *olarak ayarlandığında yönlendirme değişikliğine yanıt olarak resize olaylarını gönderir. Diğer modlarda, sahne alanının boyutları değişmez; bunun yerine, yerini tutması için içerik ölçeklenir.*

Uygulama görüntüleme değişikliklerini işleme

AIR 2.6 ve üstü

AIR'de, uygulama tanımlayıcısındaki `softKeyboardBehavior` öğesini `none` olarak ayarlayarak, yazılım klavyesini kaldırmayla ilişkili varsayılan kaydırma ve yeniden boyutlandırma işlevlerini kapatabilirsiniz.

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

Otomatik davranışı devre dışı bıraktığınızda, uygulama ekranındaki gerekli ayarlamaları yapmak uygulamanızın sorumluluğundadır. Klavye açıldığında bir softKeyboardActivate olayı gönderilir. `softKeyboardActivate` olayı gönderildiğinde, sahne alanının `softKeyboardRect` özelliği açık klavye tarafından gizlenen alanın boyutlarını içerir. Bu boyutları, klavye açıkken veya kullanıcı yazı yazarken düzgün bir şekilde görüntülenmesini sağlayacak şekilde içeriğinizi taşımak veya yeniden boyutlandırmak için kullanın. (Klavye kapalıyken softKeyboardRect dikdörtgeninin tüm boyutları sıfırdır.)

Klavye kapandığında bir `softKeyboardDeactivate` olayı gönderilir ve uygulama ekranını normale döndürebilirsiniz.

```
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;
        }
    }
}
```

Not: *Android'de, klavyenin tam boyutlarının işletim sisteminde bulunmadığı tam ekran modunu da içeren durumlar vardır. Bu durumlarda boyut tahmin edilir. Ayrıca, yatay yönlendirmelerde, tüm metin girişleri için yerel tam ekran IME klavyesi kullanılır. Bu IME klavyesi yerleşik metin girişi alanına sahiptir ve tüm sahne alanını gizler. Ekranı doldurmayan bir yatay klavyeyi görüntülemenin bir yolu yoktur.*

Bölüm 31: Fare girdisi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanız fare girdisini yakalayıp yanıtlayarak etkileşim oluşturabilir. Bu bölüm, ActionScript 3.0 olay modelini zaten bildiğinizi varsaymaktadır. Daha fazla bilgi için "Olayları işleme" sayfa 126 bölümünü inceleyin.

Çalışma zamanı boyunca hangi fare desteği türlerinin (kalıcı imleç, stilus veya dokunma girdisi) mevcut olduğunu keşfetmeyle ilgili bilgi edinmek için bkz. "Girdi türlerini keşfetme" sayfa 564.

Daha fazla Yardım konusu

flash.ui.Mouse

flash.events.MouseEvent

"Dokunmatik, çoklu dokunmatik ve hareket girdisi" sayfa 587

Fare girişi yakalama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Fare tıklatmaları etkileşim işlevlerini tetiklemek için kullanılabilecek fare olayları oluşturur. SWF dosyasının herhangi bir yerinde oluşan fare olaylarını dinlemek için Sahne alanına bir olay dinleyici ekleyebilirsiniz. Ayrıca Sahne alanı üzerindeki InteractiveObject öğesinden miras alan nesnelere (Sprite veya MovieClip gibi) de bir nesne tıklandığında tetiklenecek olay dinleyici ekleyebilirsiniz.

Kabarcık sıralaması klavye olayları için olduğu gibi fare olayları için de geçerlidir. Aşağıdaki örnekte `square` Sahne alanının alt öğesi olduğundan kareye tıklandığında olay hem `square` hareketli grafiğinden hem de Sahne alanı nesnesinden gönderilir.

```
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 + "]");
}
```

Bir önceki örnekte fare olayının tıklatma ile ilgili konum bilgisi içerdiğine dikkat edin. `localX` ve `localY` özellikleri görüntüleme zincirindeki en alt öğenin tıklanma konumunu içerir. Örneğin, `square` öğesinin sol üst köşesine tıklandığında `square` öğesinin kayıt noktası olan [0,0] yerel koordinatları rapor edilir. Ayrıca `stageX` ve `stageY` özellikleri Sahne alanı tıklatmasının genel koordinatlarına başvurur. Aynı tıklatma bu koordinatlar için [50,50] rapor eder çünkü `square` öğesi bu koordinatlara taşınmıştır. Bu koordinat çiftlerinin her ikisini de kullanıcı etkileşimine nasıl yanıt vermek istediğinize bağlı olarak kullanabilirsiniz.

Not: *Tam ekran modunda, fare kilidini kullanmak için uygulamayı yapılandırabilirsiniz. Fare kilidi, imleci devre dışı bırakır ve sınırsız fare hareketini etkinleştirir. Daha fazla bilgi için bkz. "Tam ekran modunda çalışma" sayfa 169.*

MouseEvent nesnesi ayrıca `altKey`, `ctrlKey`ve `shiftKey` Boolean özelliklerini de içerir. Bu özellikleri fare ile tıklanırken Alt, Ctrl veya Shift tuşlarına da basılıp basılmadığını kontrol etmek için kullanabilirsiniz.

Sprite'ları sahne alanı etrafında sürükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sprite sınıfının `startDrag()` yöntemini kullanarak kullanıcının, bir Sprite nesnesini sahne alanı çevresinde sürüklemesini sağlayabilirsiniz. Aşağıdaki kod bu işleve bir örnektir:

```
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);
}
```

Ayrıntılar için, "Konum değiştirme" sayfa 176 başlığı altındaki sürükleyip bırakma etkileşimi oluşturmayla ilgili bölüme bakın.

AIR uygulamasında sürükleme ve bırakma

Adobe AIR uygulamasında kullanıcılara, verileri uygulamanıza sürüklemek ve uygulamadan çıkarmak için sürükle ve bırak desteği sağlanmıştır. Ayrıntılar için bkz. "AIR uygulamasında taşıma ve bırakma" sayfa 614.

Fare imlecini özelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Fare imleci (fare işaretçisi) Sahne alanı üzerindeki herhangi bir görüntüleme nesnesi için gizlenebilir veya değiştirilebilir. Fare imlecini gizlemek için `Mouse.hide()` yöntemini çağırın. `Mouse.hide()` yöntemini çağırıp `MouseEvent.MOUSE_MOVE` olayı için Sahne alanını dinleyerek ve bir görüntüleme nesnesinin (özel imleciniz) koordinatlarını olayın `stageX` ve `stageY` özelliklerine ayarlayarak imleci özelleştirebilirsiniz. Aşağıdaki örnek bu görevin oluşturulması için temel kodu gösterir:

```
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;
}
```

Fare girdisi örneği: WordSearch

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnek fare olayları işlenerek gerçekleştirilen kullanıcı etkileşimini gösterir. Kullanıcılar bir ızgara üzerindeki rastgele seçilmiş harfleri ızgara içerisinde yatay veya dikey olarak hareket ettirerek ve aynı harfi yalnız bir kere kullanarak mümkün olduğu kadar çok sözcük türetir. Bu örnek aşağıdaki ActionScript 3.0 özelliklerini gösterir:

- Dinamik olarak bir içerik ızgarası oluşturma
- Fare olaylarını yanıtlama
- Kullanıcı etkileşimi tabanlı bir puan elde etme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. WordSearch uygulama dosyalarını Samples/WordSearch klasörü içinde bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
WordSearch.as	Uygulamanın temel işlevini sağlayan sınıf.
WordSearch.fla veya WordSearch.mxml	Flex (MXML) veya Flash (FLA) için ana uygulama dosyası.
dictionary.txt	Türetilen sözcüklere puan verilip verilmeyeceğini ve sözcüklerin doğru yazılıp yazılmadığını belirlemek için kullanılan dosya.

Bir sözlük yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sözcük türetme ile ilgili bir oyun oluşturmak için bir sözlük gereklidir. Örnek, satır başı karakteri ile ayrılmış kelimelerin bulunduğu dictionary.txt adlı bir metin dosyası içerir. `words` adlı bir dizi oluşturduktan sonra `loadDictionary()` işlevi bu dosyayı ister ve başarılı şekilde yükledikten sonra dosya uzun bir dize haline gelir. Bu dizeyi `split()` yöntemiyle satır başı (karakter kodu 10) veya yeni satır (karakter kodu 13) örneklerinin her birini atarak bir sözcük dizisine ayrıştırabilirsiniz. Ayrıştırma `dictionaryLoaded()` işlevinde gerçekleşir:

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

Kullanıcı arabirimini oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sözcükler saklandıktan sonra kullanıcı arabirimini oluşturabilirsiniz. Biri türetilen sözcüğü göndermek diğeri de yazılmakta olan kelimeyi silmek için olmak üzere iki Button örneği oluşturun. Her iki durum için de düğmenin yayınladığı `MouseEvent.CLICK` olayını dinleyip bir işlev çağırarak kullanıcı girişine yanıt vermeniz gerekir. `setupUI()` işlevinde bu kod iki düğme üzerinde dinleyici oluşturur:

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

Bir oyun tahtası oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Oyun tahtası rastgele seçilmiş harflerden oluşan bir ızgaradır. `generateBoard()` işlevinde, iç içe iki döngüden oluşan iki boyutlu bir ızgara oluşturulur. İlk döngü sıraları artırırken ikinci döngü o sırada bulunan toplam sütun sayısını artırır. Bu sıra ve sütunlar ile oluşturulan hücrelerin her biri tahta üzerinde bir harfi temsil eden bir düğme içerir.

```
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);
        }
    }
}
```

Sadece bir satırda `MouseEvent.CLICK` olayı için bir dinleyici eklenmiş olmasına rağmen `for` döngüsünde olduğundan Button örneklerinin her birine atanmıştır. Ayrıca her düğmeye sıra ve sütun konumundan derlenmiş bir ad verilmiştir ve bu şekilde kodun sonraki bölümlerinde düğmelerin sırasına ve sütununa dağa kolay başvurulabilir.

Kullanıcı girişinden sözcük oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sözcükler alt alta veya yan yana olan harfler seçilerek türetilebilir ancak harflerin tümü yalnızca bir kez kullanılabilir. Her tıklatmayla bir fare olayı üretilir ve bu noktada kullanıcının türettiği sözcük daha önce tıklanan harflerden düzgün şekilde devam ettiğinden emin olmak için kontrol edilmelidir. Eğer devam etmiyorsa önceki sözcük silinir ve yeni sözcüğe başlanır. Bu kontrol, `isLegalContinuation()` yönteminde oluşur.

```
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));
}
```

String sınıfının `charAt()` ve `indexOf()` yöntemleri hem tıklanmakta olan hem de daha önce tıklanmış olan düğmelerin uygun sıralarını ve sütunlarını getirir. Bir sıra veya sütun değişmemişse veya değişen sıra veya sütun öncekinden sadece bir farklıysa `isLegalContinuation()` yöntemi `true` değerini döndürür. Oyunun kurallarını değiştirmek ve çapraz sözcük türetmeye izin vermek isterseniz değişmeyen sıra veya sütunun işaretlerini kaldırabilirsiniz, bu durumda son satır aşağıdaki şekilde görünecektir:

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol -
currButtonCol) <= 1));
```

Gönderilen sözcükleri kontrol etme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Oyun kodunu tamamlamak için, gönderilen sözcükleri kontrol etme ve puan hesaplama mekanizmaları gereklidir. `searchForWord()` yöntemi her ikisini de içerir:

```
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;
}
```

Bu işlevin döngüsü sözlükteki tüm sözcükleri kapsar. Kullanıcının türettiği sözcüğün sözlükte bir karşılığı varsa sözlük içindeki pozisyonu döndürülür. Ardından `submitWord()` yöntemi yanıtı kontrol eder ve pozisyon geçerliyse puanı günceller.

Özelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Sınıf başlangıcında birkaç sabit vardır. Oyunu bu değişkenlerin değerleriyle oynayarak özelleştirebilirsiniz. Örneğin `TOTAL_TIME` değişkeninin değerini yükselterek oyun süresini değiştirebilirsiniz. Ayrıca sözcük bulmayı kolaylaştırmak için `PERCENT_VOWELS` değişkenini değerini de artırabilirsiniz.

Bölüm 32: Dokunmatik, çoklu dokunmatik ve hareket girdisi

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Flash Platform'un dokunmatik olay işleme özelliklerine dokunmatik cihazlarda tek bir temas noktasından veya çoklu temas noktalarından alınan girdiler de dahildir. Flash çalışma zamanları ise bir hareket yaratma hareketiyle çoklu dokunma noktalarını birleştiren olayları işler. Diğer bir deyişle, Flash çalışma zamanları iki girdi türünü yorumlar:

Dokunmatik parmak, stilus veya başka bir araçla dokunmatik aygıtta tek noktalı bir aygıtla girdi. Bazı aygıtlar parmak veya stilusla aynı anda birden fazla temas noktasını destekler.

Çoklu Dokunma aynı anda birden fazla temas noktasıyla girdi.

Hareket Bir aygıt veya işletim sisteminin bir ya da daha fazla dokunma olayına yanıt olarak yorumladığı girdi. Örneğin, bir kullanıcı iki parmağını aynı anda döndürür ve aygıt ya da işletim sistemi bu dokunma girdisini bir döndürme hareketi olarak yorumlar. Bazı el hareketleri bir parmak ya da dokunma noktasıyla gerçekleştirilirken, bazı hareketler birden fazla dokunma noktası gerektirir. Aygıt veya işletim sistemi, girdiye atanacak hareket tipini belirler.

Hem dokunma hem de hareket girdisi, kullanıcının aygıtına bağlı olarak çoklu dokunma girdisi olabilir. ActionScript dokunma olaylarını, hareket olaylarını ve çoklu girdi için ayrı takip edilen dokunma olaylarını işleyecek API'ler sağlar.

Not: *Dokunma ve hareket olaylarının dinlenmesi, bilgi işlem aygıtına ve işletim sistemine bağlı olarak işleme kaynaklarının önemli bir kısmını tüketebilir (saniye başına birkaç kare oluşturmaya eşittir) Dokunma ve hareketin sağladığı ekstra işlevselliğe gerçekten ihtiyacınız yoksa fare olaylarını kullanmak genellikle daha iyidir. Dokunma ve hareket olaylarını kullanmadığınızda, özellikle kaydırma, döndürme veya yakınlaştırma işleminde olduğu gibi bu tür olaylar hızlı bir şekilde gönderilirken, oluşan grafik değişikliklerinin miktarını azaltmayı göz önünde bulundurun. Örneğin, kullanıcı bir bileşeni yakınlaştırma hareketi kullanarak yeniden boyutlandırırken bileşenin içindeki animasyonu durdurabilirsiniz.*

Daha fazla Yardım konusu

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 (Mobil Telefonda Dokunma Olayları ve Hareketler)

Mike Jones: Virtual Game Controllers (Sanal Oyun Denetleyicileri)

Dokunma girdisinin temelleri

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Flash Platform dokunma girdisini destekleyen bir ortamda çalışırken, InteractiveObject örnekleri dokunma olaylarını dinleyip işleyicileri çağırabilir. Genelde dokunma, çoklu dokunma ve hareket olaylarını ActionScript'teki diğer olaylar gibi işlersiniz (ActionScript ile olay işleme hakkında temel bilgiler edinmek için bkz. "Olayları işleme" sayfa 126).

Ancak, Flash çalışma zamanının bir dokunma veya hareket yorumlaması için, çalışma zamanının dokunma veya çoklu dokunma girdisini destekleyen bir donanım ve yazılımda çalışıyor olması gerekir. Farklı dokunmatik ekran tiplerini karşılaştıran bir çizelge için bkz. "Girdi türlerini keşfetme" sayfa 564. Bunun yanında, çalışma zamanı bir kap uygulamasında (tarayıcı gibi) çalışıyorsa, bu kap girdiyi çalışma zamanına geçirir. Bazı durumlarda, geçerli donanım ve işletim sistemi ortamı çoklu dokunmayı destekler, ancak Flash çalışma zamanını içeren tarayıcı girdiyi yorumlar ve çalışma zamanına geçirmez. Ya da girdiyi bir bütün olarak yok sayabilir.

Aşağıdaki şemada girdinin kullanıcıdan çalışma zamanına akışı gösterilmektedir:

Girdinin kullanıcıdan Flash Platform çalışma zamanına akışı

Neyse ki, dokunmatik uygulamalar geliştirmeye yönelik ActionScript API'si, çalışma zamanı ortamında dokunma veya çoklu dokunma girdilerine verilen desteği belirleyecek sınıf, yöntem ve özelliklere sahiptir. Dokunma girdisi desteğini belirlemek için kullandığınız API, dokunma olayı işlemek için "keşif API'sidir".

Önemli kavramlar ve terimler

Aşağıdaki referans listesi dokunma olayı işleme uygulamaları yazmaya ilişkin önemli terimler içermektedir:

Keşif API'si Çalışma zamanı ortamının dokunma olayları ve farklı girdi modları desteğini sınamada kullanılan yöntem ve özellikler.

Dokunma olayı Tek temas noktası kullanarak dokunmatik bir aygıt üzerinde gerçekleştirilen girdi eylemi.

Dokunma noktası Tek dokunma olayında temas noktası. Bir aygıt hareket girdisini desteklemese bile, aynı anda birden fazla dokunma noktasını destekleyebilir.

Dokunma dizisi Tek bir dokunuşun ömrünü ifade eden olaylar dizisi. Bu olaylara bir başlangıç, sıfır veya daha fazla hamle ve bir son dahildir.

Çoklu dokunma olayı Birden fazla temas noktası kullanarak (birden fazla parmak gibi) dokunmatik bir aygıt üzerinde gerçekleştirilen girdi eylemi.

Hareket olayı Birtakım karmaşık hareketleri izleyerek dokunmatik bir aygıt üzerinde gerçekleştirilen girdi eylemi. Örneğin, bir hareket iki parmakla ekrana dokunuyor ve döndürmeyi göstermek amacıyla parmakları soyut bir çemberin çevresinde aynı anda hareket ettiriyor.

Aşamalar Olay akışındaki belirgin zaman noktaları (başlangıç ve son gibi)

Stilus Dokunmatik ekranla etkileşimi sağlayan bir araç. Bir stilus insan parmağından daha fazla kesinlik sağlar. Bazı aygıtlar yalnızca belli stilus tiplerinin girdilerini tanır. Stilus girdisini tanıyan aygıtlar aynı anda birden fazla temas noktasını veya parmak temasını da tanıyabilir.

Bastır-tıklat Kullanıcının bir parmağıyla dokunmatik aygıta basıp, diğer parmağı veya işaret aygıtıyla tıklattığı özel bir çoklu dokunma girdisi türüdür. Bu hareket çoğu zaman çoklu dokunma uygulamalarındaki fare sağ tıklatmasını taklit etmek için kullanılır.

Dokunma girdisi API yapısı

ActionScript girdisi API'si, dokunma girdisi işlemenin Flash çalışma zamanı donanım ve yazılım ortamına bağlı olmasına hitap etmeyi amaçlar. Dokunma girdisi API'si öncelikli olarak üç dokunmatik uygulama geliştirme gereksinimine hitap eder: keşif, olaylar ve aşamalar. Siz uygulamayı geliştirirken hedef aygıt bilinmiyor olsa bile Bu API'leri kullanıcı için öngörülebilir ve duyarlı bir deneyim oluşturmak üzere koordine edin.

Keşif

Keşif API'si çalışma zamanındaki donanım ve yazılım ortamını test edebilme becerisini sağlar. Çalışma zamanı tarafından yerleştirilen değerler, Flash çalışma zamanının geçerli bağlamında mevcut olan dokunma girdilerini belirler. Ayrıca, keşif özellikleri ve yöntemlerinin koleksiyonunu kullanarak uygulamanızı fare olaylarına tepki verecek şekilde ayarlayın (ortamın bazı dokunma girdilerini desteklemediği durumlarda dokunma olaylarının yerine). Daha fazla bilgi için bkz. "Dokunma desteği keşfi" sayfa 590.

Olaylar

ActionScript diğer olayları olduğu gibi dokunma girdisi olaylarını da olay dinleyicileri ve olay işleyicileriyle yönetir. Ancak, dokunma girdisi olayını işlemede şunlar da hesaba katılmalıdır:

- Bir dokunma, aygıt veya işletim sistemi tarafından dokunma dizisi veya hep beraber bir hareket olarak birçok farklı şekilde yorumlanabilir.
- Tek dokunmalı bir dokunmatik aygıt (parmak, stilus veya işaret aygıtıyla) her zaman bir fare olayı da gönderir. Fare olayını MouseEvent sınıfındaki olay türleriyle işleyebilirsiniz. Ya da uygulamanızı yalnızca dokunma olaylarına yanıt verecek şekilde tasarlayabilirsiniz. Ya da her ikisine de yanıt veren bir uygulama tasarlayabilirsiniz.
- Bir uygulama aynı anda birden fazla dokunma olayına yanıt verirken, her birini ayrı ayrı işler.

Tipik olarak, keşif API'sini uygulamanızın işlediği olayları şartlı olarak işlemek ve bu olayların nasıl işlendiğini görmek için kullanın. Uygulama çalışma zamanı ortamını bildiği zaman, kullanıcının uygulamayla etkileşim kurmasıyla birlikte uygun işleyiciyi çağırabilir veya doğru olay nesnesini belirleyebilir. Ya da uygulama geçerli ortamda bazı girdilerin işlenemeyeceğini belirtip, kullanıcıya bir alternatif veya bilgiler sunar. Daha fazla bilgi edinmek için bkz. "Touch olayını işleme" sayfa 591 ve "Gesture olayını işleme" sayfa 595.

Aşamalar

Dokunma ve çoklu dokunma uygulamalarında, dokunma olayı nesneleri kullanıcı etkileşimi aşamalarını takip edecek özellikler içerir. Kullanıcı girdisinin başlangıç, güncelleme veya bitiş aşaması gibi aşamalarını kullanıcıya geribildirimde bulunmak üzere işleyecek ActionScript yazın. Olay aşamalarına kullanıcı, ekrana dokunduğunda ve dokunma noktasını taşıdığında görsel nesneler değişecek şekilde yanıt verin. Ya da bir hareket değişim geçirdikçe belli özelliklerini takip etmek için aşamaları kullanın.

Dokunma noktası olayları için, kullanıcının bir interaktif nesneye ne kadar uzun süre dayandığını takip edin. Bir uygulama aynı anda ayrı ayrı birden fazla dokunma noktası aşamasını takip edip, her birini uygun şekilde işleyebilir.

Bir hareket için, hareketin meydana geldiği sırada dönüştürülmesine ilişkin bilgileri yorumlayın. Temas noktası ekran üzerinde hareket ettikçe, temas noktasının (veya birden fazla) koordinatlarını izleyin.

Dokunma desteği keşfi

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Uygulamanızın işlediği dokunma girdisinin kapsamını belirlemek için Multitouch sınıfı özelliklerini kullanın. Ardından, ActionScript'inizin işlediği olaylar için destek olup olmadığını görmek üzere ortamı test edin. Spesifik olarak, öncelikle uygulamanızın dokunma girdisi tipini belirleyin. Seçenekler şunlardır: dokunma noktası, hareket veya hiçbiri (tüm dokunma girdilerini fare tıklatmaları ve fare olayı işleyicileri gibi yorumla) Ardından, Multitouch sınıfının özellik ve yöntemlerini kullanarak çalışma zamanının uygulamanız tarafından istenen dokunma olaylarını desteklediğinden emin olun. Çalışma zamanı ortamının dokunma girdisi türlerini desteklediğini (örneğin, el hareketlerini yorumlayabildiğini) ve uygun şekilde yanıt verebildiğini sınayın.

Not: *Multitouch sınıfı özellikleri statik özellikler olup, herhangi bir sınıfın örneklerine ait değildir. Bunları Multitouch.property söz dizimiyle kullanın; örneğin:*

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

Girdi tipini ayarlayın

Flash çalışma zamanı yorumlanacak dokunma girdisinin türünü bilmelidir; çünkü bir dokunma olayının birden fazla öğesi veya aşaması olabilir. Dokunmatik bir ekrana bir parmak dokunduğunda, çalışma zamanı bir dokunma olayı gönderiyor mu? Ya da bir hareketi mi bekliyor? Ya da çalışma zamanı bu dokunuşu bir fare olayı gibi mi izliyor? Dokunma girdisini destekleyen bir uygulama, Flash çalışma zamanı için işlediği dokunma olaylarının türünü belirlemelidir. Çalışma zamanı için dokunma girdisi türünü belirlemek için `Multitouch.inputMode` özelliğini kullanın. Girdi modu aşağıdaki üç seçenekten biri olabilir:

Yok Dokunma olayları için özel bir işlem yapılmaz. Ayar: `Multitouch.inputMode=MultitouchInputMode.NONE` ve girdiyi işlemek için MouseEvent sınıfını kullanın.

Tek dokunma noktaları Tüm dokunma girdileri ayrı ayrı yorumlanır ve tüm dokunma noktaları izlenip işlenebilir. Ayar: `Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT` girdiyi işlemek için TouchEvent sınıfını kullanın.

Hareket girdisi Aygıt ve işletim sistemi, girdiyi ekran üzerindeki karmaşık bir parmak hareketleri biçimi gibi yorumlar. Aygıt ve işletim sistemi, hareketi topluca tek bir hareket girdisi olayı olarak atar. Ayar: `Multitouch.inputMode=MultitouchInputMode.GESTURE` ve girdiyi işlemek için TransformGestureEvent, PressAndTapGestureEvent veya GestureEvent sınıflarını kullanın.

Bir dokunma olayını işlemeden önce girdi türünü belirlemek için `Multitouch.inputMode` özelliğini kullanan bir örnek için bkz. "Touch olayını işleme" sayfa 591.

Dokunma girdisi desteği testi

Multitouch sınıfının diğer özellikleri, uygulamanızın geçerli ortamın dokunma desteğine hassas ayarını yapacak değerleri sunar. Flash çalışma zamanı, aynı anda izin verilen dokunma noktası veya mevcut hareket sayısının değerlerini verir. Çalışma zamanı, uygulamanızın ihtiyaç duyduğu dokunma olayı işlemesini desteklemeyen bir ortamsa, kullanıcıya bir alternatif sunun. Örneğin, fareyle olay işleme veya geçerli ortamda hangi özelliklerin mevcut olup olmadığıyla ilgili bilgiler verin

API'yi klavye, dokunma ve fare desteği için de kullanabilirsiniz, bkz. "Girdi türlerini keşfetme" sayfa 564.

Uyumluluk testi hakkında daha fazla bilgi edinmek için bkz. "Sorun Giderme" sayfa 599.

Touch olayını işleme

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

ActionScript'te temel dokunma olayları da fare olayları gibi diğer olayları işlediğiniz şekilde işlenir. TouchEvent sınıfı bölümündeki olay türü sabitleriyle tanımlanan bir dizi dokunma olayını dinleyebilirsiniz.

Not: *Çoklu dokunma noktası girdisi için (bir aygıta birden fazla parmakla dokunma gibi), ilk temas noktası bir fare olayı ve dokunma olayı gönderir.*

Basit bir dokunma olayını işlemek için:

1. `flash.ui.Multitouch.inputMode` özelliğini `MultitouchInputMode.TOUCH_POINT` özelliğine ayarlayarak uygulamanızın dokunma olaylarını işlemesini sağlayın.
2. Sprite veya TextField gibi özelliklerini InteractiveObject sınıfından alan bir sınıf örneğine olay dinleyicisi ekleyin.
3. İşlenecek dokunma olayının türünü belirtin.
4. Olaya yanıt olarak bir işlem gerçekleştirmesi için bir olay işleyici işlevi çağırın.

Örneğin, dokunmatik bir ekranda mySprite üzerindeki kareye dokunulduğunda aşağıdaki kod bir mesaj görüntüler:

```
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);
}
```

Touch Olayı özellikleri

Bir olay meydana geldiğinde, olay nesnesi oluşturulur. TouchEvent nesnesi, dokunma olayının yeri ve şartları hakkında bilgiler içerir. Bu bilgileri almak için olay nesnesinin özelliklerinden faydalanabilirsiniz.

Örneğin, aşağıdaki kod bir TouchEvent nesnesi *evt* oluşturur ve ardından metin alanında olay nesnesinin `stageX` özelliğini (dokunmanın meydana geldiği Sahne alanının x koordinatı) gösterir:

```
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);
}
```

Olay nesnesiyle kullanılabilir özellikler için bkz. TouchEvent sınıfı.

Not: *Tüm TouchEvent özellikleri tüm çalışma zamanı ortamlarında desteklenmez. Örneğin, tüm dokunmatik aygıtlar, kullanıcının dokunmatik ekrana uyguladığı basınç miktarını algılayamaz. Dolayısıyla, `TouchEvent.pressure` özelliği bu aygıtlarda desteklenmez. Uygulamanızın çalıştığından emin olmak için belli özelliklerin desteğini sınamayı deneyin ve daha fazla bilgi edinmek için "Sorun Giderme" sayfa 599 bölümüne bakın.*

Dokunma olayı aşamaları

Fare olaylarında yaptığınız gibi dokunma olaylarını da InteractiveObject üzerinde ve dışındaki çeşitli sahnelerde takip edin. Dokunma olaylarını bir dokunma etkileşiminin başlangıcı, ortası ve sonunda izleyin. TouchEvent sınıfı `touchBegin`, `touchMove` ve `touchEnd` olaylarını işleyecek değerleri sunar.

Örneğin, kullanıcıya bir görüntüleme nesnesine dokunduğunda ve nesneyi hareket ettirdiğinde görsel geribildirimde bulunmak için `touchBegin`, `touchMove` ve `touchEnd` olaylarını kullanabilirsiniz:

```
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;
}
```

Not: *Başlangıç dokunma dinleyicisi mySprite'a eklenirken, dokunma olayını taşıyıp sona erdirecek olay eklenmez. Kullanıcının parmağı ve işaret aygıtı görüntüleme nesnesinden düz ilerlerse, Sahne dokunma olayını dinlemeye başlar.*

Dokunma Noktası Kimliği

`TouchEvent.touchPointID` özelliği, dokunma girdisine yanıt veren uygulamaları yazarken gerekli bir unsurdur. Flash her dokunma noktasına benzersiz bir `touchPointID` değeri atar. Dokunma girdisinin aşamalarına veya hareketine bir uygulamadan her yanıt geldiğinde, olayı işlemeden önce `touchPointID` değerini kontrol edin. Sprite sınıfının dokunma girdisi sürükleme yöntemleri, doğru girdi örneğinin işlenebilmesi için parametre olarak `touchPointID` özelliğini kullanır. `touchPointID` özelliği, bir olay işleyicinin doğru dokunma noktasına yanıt vermesini sağlar. Aksi takdirde, olay işleyici öngörülemez davranışlar oluşturarak aygıt üzerindeki her türlü dokunma olayı türü örneğine (örneğin, tüm `touchMove` olaylarını) yanıt verir. Bu özellik bilhassa kullanıcının nesneleri sürüklemesi sırasında önemlidir.

Dokunma dizisinin tamamını yönetmek için `touchPointID` özelliğini kullanın. Bir dokunma dizisinde her biri aynı `touchPointID` değerine sahip bir `touchBegin` olayı, sıfır veya daha fazla `touchMove` olayı ve bir `touchEnd` olayı bulunur.

Aşağıdaki örnek, dokunmatik taşıma olayına yanıt vermeden önce doğru `touchPointID` değerini test etmek için bir *touchMoveID* değişkeni belirler. Aksi takdirde öteki dokunmatik girdi de olay işleyicisini tetikler. Görüntüleme nesnesinin değil, sahnedeki taşıma ve bitiş aşamalarının dinleyicilerini fark edin. Kullanıcının dokunuşu görüntüleme nesnesi sınırlarını aşarsa sahne, taşı veya bitiş aşamalarını dinler.

```
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;
}
```

Dokun-sürükle

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Dokunma noktası girdisini destekleyen dokunmatik uygulamalara ek destek sağlamak üzere, Sprite sınıfı adresine iki yöntem eklenmiştir: `Sprite.startTouchDrag()` ve `Sprite.stopTouchDrag()`. Bu yöntemler fare olaylarında `Sprite.startDrag()` ve `Sprite.stopDrag()` gibi davranır. Ancak, `Sprite.startTouchDrag()` ve `Sprite.stopTouchDrag()` yöntemlerinin her ikisinin de parametre olarak `touchPointID` aldığı gözden kaçırılmamalıdır.

Çalışma zamanı, bir dokunma olayı için olaya `touchPointID` değerini atar. Ortamın aynı anda birden fazla dokunma noktasını desteklemesi durumunda (el hareketlerini işlemese bile), belli bir dokunma noktasına yanıt vermek için bu değeri kullanın. `touchPointID` özelliği hakkında daha fazla bilgi almak için bkz. "Dokunma Noktası Kimliği" sayfa 593.

Aşağıdaki kod, bir dokunma olayı için basit bir sürüklemeye başla olayı işleyicisini ve sürüklemeyi durdur olayı işleyicisini göstermektedir. *bg* değişkeni, *mySprite* içeren bir görüntüleme nesnesidir:

```
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");
}
```

Aşağıda ise sürüklemeyi dokunma olayı aşamalarıyla birleştiren daha gelişmiş bir örnek gösterilmektedir:

```
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;
}
```

Gesture olayını işleme

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Hareket olaylarını da basit dokunma olayları gibi işleyin. TransformGestureEvent sınıfı, GestureEvent sınıfı ve PressAndTapGestureEvent sınıfı içerisinde olay türü sabitleriyle belirlenmiş bir dizi hareket olayını dinleyebilirsiniz.

Bir hareket dokunma olayını işlemek için:

1. `flash.ui.Multitouch.inputMode` özelliğini `MultitouchInputMode.GESTURE` olarak belirleyip, uygulamanızı hareket girdilerini işlemeye ayarlayın.
2. Sprite veya TextField gibi özelliklerini InteractiveObject sınıfından alan bir sınıf örneğine olay dinleyicisi ekleyin.
3. İşlenecek hareket olayının türünü belirtin.
4. Olaya yanıt olarak bir şey yapması için bir olay işleyici işlevi çağırın.

Örneğin, dokunmatik bir ekranda *mySprite* üzerindeki kareye dokunulduğunda aşağıdaki kod bir mesaj görüntüler:

```
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);
}
```

İki parmaklı dokunma olayları da aynı şekilde işlenirken, GestureEvent sınıfını kullanır:

```
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);
}
```

Bastır-tıklat olayları da aynı şekilde işlenirken, PressAndTapGestureEvent sınıfını kullanır:

```
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);
}
```

Not: *Tüm GestureEvent, TransformGestureEvent ve PressAndTapGestureEvent olay türleri bütün çalışma zamanı ortamlarında desteklenmez. Örneğin, tüm dokunmatik aygıtlar çok parmaklı dokunuşu algılayamaz. Dolayısıyla, bu aygıtlar üzerinde InteractiveObject* `gestureSwipe` *olayları desteklenmez. Uygulamanızın çalıştığından emin olmak için belli olayların desteğini sınamayı deneyin ve daha fazla bilgi edinmek için* *"Sorun Giderme" sayfa 599 bölümüne bakın.*

Gesture Olayı özellikleri

Hareket olaylarının kapsamı, basit dokunmatik olaylarınkinden küçüktür. Bunlara da aynı şekilde, olay işleyici işlevindeki olay nesnesinden erişebilirsiniz.

Örneğin, kullanıcı bir döndürme hareketi gerçekleştirdikçe aşağıdaki kod *mySprite* öğesini döndürür. Metin alanı, son hareketten bu yana dönüş miktarını gösterir (bu kodu test ederken, değerlerin değişimini görmek için defalarca döndürün):

```
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();
}
```

Not: *Tüm TransformGestureEvent özellikleri tüm çalışma zamanı ortamlarında desteklenmez. Örneğin, tüm dokunmatik aygıtlar ekran üzerindeki bir hareketin dönüşünü algılayamaz. Dolayısıyla,* `TransformGestureEvent.rotation` *özelliği bu aygıtlarda desteklenmez. Uygulamanızın çalıştığından emin olmak için belli özelliklerin desteğini sınamayı deneyin ve daha fazla bilgi edinmek için* *"Sorun Giderme" sayfa 599 bölümüne bakın.*

Gesture aşamaları

Bunun yanında, hareket olayları aşamalardan takip edilebilir, böylece hareket meydana gelirken özellikleri takip edebilirsiniz. Örneğin, bir nesne dokunma hareketiyle taşındığında x koordinatlarını takip edebilirsiniz. Dokunma tamamlandıktan sonra, yol üzerindeki tüm noktalardan bir çizgi oluşturmak için bu değerleri kullanın. Ya da bir görüntüleme nesnesi ekran boyunca kaydırma hareketiyle sürüklendiğinde, onu görsel olarak değiştirin. Kaydırma hareketi tamamlandığında nesneyi yeniden değiştirin.

```
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;
}
}
```

Not: *Güncelleme aşamasının sıklığı, çalışma zamanının ortamına bağlıdır. Bazı işletim sistemi ve donanım kombinasyonları, güncellemelere hiç dayanmaz.*

Hareket aşaması basit hareket olayları için "tümü" şeklindedir

Bazı hareket olay nesneleri, hareket olayının aşamalarını tek tek izlemek yerine, olay nesnesinin aşama özelliğini hepsi değeriyle doldurur. Basit hareketler ve iki parmakla dokunma, olayı birden fazla aşamayla izlemez. `gestureSwipe` veya `gestureTwoFingerTap` olaylarının dinlenmesi için bir InteractiveObject olay nesnesinin `phase` özelliği, olay gönderildiğinde daima `hepsi`'dir:

```
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"
}
```

Sorun Giderme

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Dokunma girdisi için donanım ve yazılım desteği hızla değişmektedir. Bu başvuru, çoklu dokunma desteği bulunan işletim sistemi ve yazılım kombinasyonunu içeren her aygıtın bir listesini tutmaz. Ancak, uygulamanızın çoklu dokunma desteğine sahip bir aygıta konuşlandırılmış olup olmadığını belirlemek için keşif API'sini kullanma kurallarını sağlar ve ActionScript kodunuzun sorunlarını gidermek için ipuçları verir.

Flash çalışma zamanları, dokunma olaylarına aygıtın, işletim sisteminin veya içindeki yazılımların (tarayıcı gibi) çalışma zamanına geçirdiği bilgilere göre yanıt verir. Yazılım ortamına bu bağımlılık, çoklu dokunma uyumluluğunu belgelemeyi zorlaştırır. Bazı aygıtlar hareket veya dokunma hareketini başka bir aygıttan farklı bir şekilde yorumlar. Döndürme, aynı anda dönen iki parmakla mı tanımlanıyor? Bir parmakla döndürme işlemi, ekranda bir daire çiziyor mu? Donanım ve yazılım ortamına bağlı olarak, döndürme hareketi tamamen farklı bir şey olabilir. Bu yüzden aygıt, işletim sistemine kullanıcı girdisini söyler, ardından işletim sistemi bu bilgileri çalışma zamanına geçirir. Çalışma zamanı bir tarayıcının içindeyse, tarayıcı yazılımı zaman zaman hareket veya dokunma olayını yorumlar ve girdiyi çalışma zamanına geçirmez. Bu davranış "kısayol tuşları"nın davranışına benzer: Flash Player'a tarayıcı içerisinde bir şey yaptırmak için belli bir tuş kombinasyonu kullanmaya çalışırsınız ve tarayıcı bunun yerine bir menü açmaya devam eder.

Ayrı API ve sınıflar, belli işletim sistemleriyle uyumlu olmadıklarını belirtir. Ayrı API girişlerini buradan Multitouch sınıfıyla başlayarak keşfedebilirsiniz:
http://help.adobe.com/tr_TR/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html.

Yaygın hareket ve dokunma açıklamaları şunlardır:

Kaydır Bir parmağı soldan sağa veya sağdan sola taşıyın. Bazı aygıtlar iki parmakla kaydırma gerektirir.

Döndürme İki parmakla dokunun ve bir daire etrafında hareket ettirin (her ikisi de bir yüzey üzerinde aynı anda hayali bir daireyi takip ediyormuş gibi). Pivot noktası, iki parmağın dokunduğu noktaların arasındaki orta noktada belirlenir.

Kaydır Üç parmağı soldan sağa veya sağdan sola, yukarıdan aşağıya veya aşağıdan yukarıya hızla taşıyın.

Yakınlaştır İki parmakla dokunun, sonra yakınlaştırmak için bunları birbirinden ayırın ve uzaklaştırmak için birbirine yaklaştırın.

Bas ve dokun Bir parmakla taşırken veya basarken, diğer parmakla yüzeyi tıklatın.

Her aygıt, aygıtın desteklediği el hareketleri ve bir hareketin bu aygıtta nasıl gerçekleştirileceği ile ilgili ayrı belgelere sahiptir. Genellikle, işletim sistemine bağlı olarak kullanıcı, el hareketleri arasında tüm parmaklarının aygıtla temasını kesmelidir.

Uygulamanızın dokunma veya hareket olaylarına yanıt vermediğini görürseniz, aşağıdaki testleri yapın:

1 InteractiveObject sınıfından alınan bir nesne sınıfına bağlı dokunma veya hareket olayları için olay dinleyicileriniz var mı? Dokunma ve hareket olaylarını yalnızca InteractiveObjects sınıfı örnekleri dinleyebilir

2 Uygulamanızı Flash Professional CS5'te mi test ediyorsunuz? Öyleyse, uygulamayı yayınlamayı ve test etmeyi deneyin; çünkü Flash Professional etkileşimi engelleyebilir.

3 İlk olarak basit başlayın ve neyin işe yaradığını görün (aşağıdaki kod örneği `Multitouch.inputMode` API girişinden alınmıştır:

```
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);
}
```

Dikdörtgene dokunun. Bu örnek işe yararsa, ortamınızın basit bir tıklatmayı desteklediğini anlarsınız. Bundan sonra daha karmaşık bir işlem deneyebilirsiniz.

Hareket desteğini test etmek daha karmaşık bir işlemdir. Ayrı bir aygıt veya işletim sistemi herhangi bir hareket girdisini destekleyebilir veya hiçbirini desteklemeyebilir.

Aşağıda yakınlaştırma hareketi için basit bir test vardır:

```
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";
}
```

Aygıtta bir yakınlaştırma hareketi gerçekleştirin ve metin alanının `Zoom is supported` mesajıyla doldurulup doldurulmadığına bakın. Test uygulamasının herhangi bir kısmında hareket uygulayabilmeniz için olay dinleyici sahne alanına eklenir.

Aşağıda yatay kaydırma hareketi için basit bir test vardır:

```
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";
}
```

Aygıtta bir yatay kaydırma hareketi gerçekleştirin ve metin alanının `Pan is supported` mesajıyla doldurulup doldurulmadığına bakın. Test uygulamasının herhangi bir kısmında hareket uygulayabilmeniz için olay dinleyici sahne alanına eklenir.

Bazı işletim sistemi ve aygıt kombinasyonları her iki hareketi de destekler. Bazıları yalnızca bir tanesini desteklerken bazıları hiçbirini desteklemez. Emin olmak için uygulamanızın dağıtım ortamını test edin.

Bilinen Sorunlar

Aşağıda dokunma girdisiyle ilgili bilinen sorunlar verilmiştir:

1. Windows Mobile işletim sistemindeki Mobile Internet Explorer, SWF dosya içeriğini otomatik olarak yakınlaştırır:

   Bu Internet Explorer yakınlaştırma davranışı, SWF dosyasını barındıran HTML sayfasına aşağıdakiler eklenerek iptal edilir:

   ```
   <head>
   <meta name="viewport" content="width=device-width, height=device-height, initial-
   scale=1.0">
   </head>
   ```

2. Windows 7 (ve muhtemelen başka işletim sistemleri), kullanıcı el hareketleri arasında işaret aygıtını (veya parmaklarını) ekrandan kaldırmalıdır. Örneğin, bir görüntüyü döndürmek ve yakınlaştırmak için:
   - Döndür hareketini yapın.
   - Parmaklarınızı ekrandan kaldırın.
   - Parmaklarınızı ekrana geri koyun ve yakınlaştırma hareketini yapın.
3. Windows 7'de (ve büyük ihtimalle diğer işletim sistemlerinde), kullanıcı hareketi çok hızlı bir şekilde gerçekleştirirse, döndür ve yakınlaştır hareketleri her zaman bir “güncelleme” aşaması oluşturmaz.
4. Windows 7 Starter Edition çoklu dokunma olayını desteklemez. Ayrıntılar için bkz. AIR Laboratuarları Forumu: http://forums.adobe.com/thread/579180?tstart=0
5. Mac OS 10.5.3 ve üstü sürümler için, `Multitouch.supportsGestureEvents` değeri donanımın hareket olaylarını desteklemediği durumlarda bile daima `true`'dur.

Bölüm 33: Kopyalama ve yapıştırma

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Sistem panosundan ve sistem panosuna bilgi yapıştırmak için pano API'sindeki sınıfları kullanın. Adobe® Flash® Player veya Adobe® AIR® içinde çalışan bir uygulamaya veya uygulamadan aktarılabilen veri biçimleri arasında şunlar yer alır:

- Metin
- HTML biçimli metin
- Zengin Metin Biçimli veriler
- Serileştirilmiş nesneler
- Nesne başvuruları (yalnızca kaynak uygulama içinde geçerli)
- Bitmap'ler (yalnızca AIR)
- Dosyalar (yalnızca AIR)
- URL dizeleri (yalnızca AIR)

Kopyala ve yapıştır işleminin temelleri

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Kopyala ve yapıştır API'si aşağıdaki sınıfları içerir.

Paket	Sınıflar
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

Statik `Clipboard.generalClipboard` özelliği, işletim sistemi panosunu temsil eder. Clipboard sınıfı, clipboard nesnelerinden veri okuma ve bu nesnelere veri yazmaya yönelik yöntemler sağlar.

HTMLLoader sınıfı (AIR'de) ve TextField sınıfı, normal kopyalayıp yapıştırma klavye kısayolları için varsayılan davranışı uygular. Özel bileşenler için kopyalama ve yapıştırma kısayolu davranışını uygulamak üzere doğrudan bu tuş girişlerini dinleyebilirsiniz. Ayrıca tuş girişlerine dolaylı olarak yanıt vermek için, tuş eşdeğerleriyle beraber yerel menü komutlarını da kullanabilirsiniz.

Diğer uygulamaların verileri anlama ve kullanma kapasitesini artırmak için, aynı bilgilerin farklı temsilleri tek bir Clipboard nesnesinde kullanılabilir hale getirilebilir. Örneğin bir görüntü, görüntü verisi, serileştirilmiş Bitmap nesnesi veya bir dosya olarak dahil edilebilir. Bir biçimdeki verilerin oluşturulması ertelenebilir, böylece bu biçimdeki veriler okunana kadar biçim gerçekten oluşturulmaz.

Sistem panosundan okuma ve sistem panosuna yazma

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

İşletim sistemi panosunu okumak için, `Clipboard.generalClipbooard` nesnesinin `getData()` yöntemini çağırarak okunacak biçimin adında iletin:

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

Not: *Flash Player'da veya AIR'deki uygulama dışı sanal alanda çalışan içerik, yalnızca `paste` olayı için olay işleyicisinde `getData()` yöntemini çağırabilir. Başka bir deyişle, yalnızca AIR uygulaması sanal alanında çalışan kod `paste` olay işleyicisi dışında `getData()` yöntemini çağırabilir.*

Panoya yazmak için, verileri `Clipboard.generalClipboard` nesnesine bir veya daha fazla biçimde ekleyin. Aynı biçimdeki verilerin otomatik olarak üzerine yazılır. Buna rağmen, diğer biçimlerdeki ilişkili olmayan verilerin silindiğinden de emin olmak için yeni veri yazmadan önce sistem panosunu temizlemek yararlı olur.

```
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);
```

Not: *Flash Player'da veya AIR'de uygulama dışı sanal alanda çalışan içerik klavye veya fare olayı gibi bir kullanıcı olayı için ya da `copy` veya `cut` olayı için bir olay işleyicisinde `setData()` yöntemini çağırabilir. Başka bir deyişle, yalnızca AIR uygulaması sanal alanında çalışan kod `setData()` kullanıcı olay işleyicisi dışında getData() yöntemini çağırabilir.*

AIR'de HTML kopyalama ve yapıştırma

Adobe AIR 1.0 ve üstü

Adobe AIR uygulamasındaki HTML ortamı kopyalama ve yapıştırma için kendi olay kümesini ve varsayılan davranışı sağlar. Yalnızca uygulama sanal alanındaki kod çalıştırma, AIR `Clipboard.generalClipboard` nesnesini kullanarak sistem panosuna doğrudan erişebilir. Uygulama dışı bir sanal alandaki JavaScript kodu, HTML belgesindeki bir öğe tarafından gönderilen copy veya paste olaylarından birine yanıt olarak gönderilen olay nesnesini kullanarak panoya erişebilir.

Copy ve paste olayları şunları içerir: `copy`, `cut` ve `paste`. Bu olaylar için gönderilen nesne `clipboardData` özelliğiyle panoya erişim sağlar.

Varsayılan davranış

Adobe AIR 1.0 ve üstü

Varsayılan olarak AIR, klavye kısayolu veya bağlam menüsü tarafından oluşturulabilen kopyala komutuna yanıt olarak seçili öğeleri kopyalar. Düzenlenebilir bölgeler içerisinde, AIR, kes komutuna yanıt olarak metni keser veya yapıştır komutuna yanıt olarak metni imlecin bulunduğu yere veya seçime yapıştırır.

Varsayılan davranışı engellemek için, olay işleyiciniz gönderilen olay nesnesinin `preventDefault()` yöntemini çağırabilir.

olay nesnesinin clipboardData özelliğini kullanma

Adobe AIR 1.0 ve üstü

Copy veya paste olaylarından birinin sonucu olarak gönderilen olay nesnesinin `clipboardData` özelliği sayesinde, pano verilerini okuyabilir ve yazabilirsiniz.

Bir copy veya cut olayını işlerken panoya yazmak için, kopyalanacak verileri ve MIME türünü ileten `clipboardData` nesnesinin `setData()` yöntemini kullanın:

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

Yapıştırılan verilere erişmek için, veri biçiminin MIME türünü ileten `clipboardData` nesnesinin `getData()` yöntemini kullanabilirsiniz. Kullanılabilir biçimler `types` özelliği tarafından bildirilir.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

`getData()` yöntemine ve `types` özelliğine yalnızca `paste` olayı tarafından gönderilen olay nesnesinde erişilebilir.

Aşağıdaki örnek bir HTML sayfasında varsayılan kopyalama ve yapıştırma davranışının nasıl geçersiz kılınacağını gösterir. `copy` olay işleyicisi, kopyalanan metni italik yapar ve HTML metni olarak panoya yapıştırır. `cut` olay işleyicisi seçili verileri panoya kopyalar ve belgeden kaldırır. `paste` işleyicisi pano içeriklerini HTML olarak ekler ve eklemeyi kalın metin olarak biçimlendirir.

```
 <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>
```

Pano veri biçimleri

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Pano biçimleri bir Clipboard nesnesinde bulunan verileri açıklar. Flash Player veya AIR uygulaması, ActionScript veri türleri ile sistem panosu biçimleri arasında standart veri biçimlerini otomatik olarak çevirir. Ayrıca, uygulama tanımlı biçimler kullanılarak ActionScript tabanlı uygulamalar içinde ve arasında uygulama nesneleri aktarılabilir.

Bir Clipboard nesnesi aynı bilginin farklı biçimlerdeki temsillerini içerebilir. Örneğin, bir Sprite öğesini temsil eden bir Clipboard nesnesi, aynı uygulama için kullanılacak bir başvuru biçimi, Flash Player veya AIR uygulamasında çalışan başka bir uygulama tarafından kullanılacak serileştirilmiş bir biçim, bir görüntü düzenleyicisi tarafından kullanılacak bitmap biçimi ve Sprite öğesinin bir temsilinin kopyalanıp dosya sistemine sürüklenmesi için belki de PNG dosyasını kodlamak üzere ertelenmiş oluşturmayla bir dosya listesi biçimi içerebilir.

Standart veri biçimleri

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Standart biçim adlarını tanımlayan sabitler ClipboardFormats sınıfında sağlanır:

Sabit	Açıklama
TEXT_FORMAT	Metin biçimi verileri, ActionScript String sınıfına çevrilir ve bu sınıftan geri çevrilir.
HTML_FORMAT	HTML işaretlemeli metin.
RICH_TEXT_FORMAT	Zengin metin biçimi verileri, ActionScript ByteArray sınıfına çevrilir ve bu sınıftan geri çevrilir. RTF işaretlemesi, herhangi bir şekilde yorumlanmaz veya çevrilmez.
BITMAP_FORMAT	(yalnızca AIR) Bitmap-biçimi verileri, ActionScript BitmapData sınıfına veya bu sınıftan çevrilir.
FILE_LIST_FORMAT	(yalnızca AIR) Dosya biçimi verileri, ActionScript File nesnelerinden oluşan bir diziye veya bu diziden çevrilir.
URL_FORMAT	(yalnızca AIR) URL-biçimi verileri, ActionScript String sınıfına veya bu sınıftan çevrilir.

Bir AIR uygulamasında HTML içeriğindeki bir `copy`, `cut` veya `paste` olayına yanıt olarak veri kopyalayıp yapıştırırken, ClipboardFormat dizeleri yerine MIME türleri kullanılmalıdır. Geçerli veri MIME türleri şunlardır:

MIME türü	Açıklama
Metin	"text/plain"
URL	"text/uri-list"
Nokta Esaslı	"image/x-vnd.adobe.air.bitmap"
Dosya listesi	"application/x-vnd.adobe.air.file-list"

Not: *Zengin metin biçimli veriler, HTML içeriğindeki `paste` olayının sonucu olarak gönderilen olay nesnesinin `clipboardData` özelliğinden kullanılamaz.*

Özel veri biçimleri

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Nesneleri başvurular veya serileştirilmiş kopyalar olarak aktarmak için, uygulama tanımlı özel biçimler kullanabilirsiniz. Başvurular yalnızca aynı uygulama dahilinde geçerlidir. Serileştirilmiş nesneler uygulamalar arasında aktarılabilir, ancak yalnızca serileştirildiğinde ve serileştirmesi kaldırıldığında geçerliliği devam eden nesnelerle kullanılabilir. Özellikleri basit türler veya serileştirilebilir nesneler olduğunda nesneler genellikle serileştirilebilir.

Serileştirilmiş bir nesneyi Clipboard nesnesine eklemek için, *serializable* parametresini `Clipboard.setData()` yöntemini çağırırken `true` değerine ayarlayın. Biçim adı standart biçimlerden biri veya uygulamanız tarafından tanımlanan rastgele bir dize olabilir.

Aktarım modları

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Bir nesne, özel veri biçimi kullanılarak panoya yazıldığında, nesne verisi, bir başvuru olarak veya orijinal nesnenin serileştirilmiş bir kopyası olarak okunabilir. Nesnelerin başvuru olarak mı serileştirilmiş kopyalar olarak mı aktarılacağını belirleyen dört aktarım modu vardır.

Aktarım modu	Açıklama
ClipboardTransferModes.ORIGINAL_ONLY	Yalnızca bir başvuru döndürülür. Bir başvuru varsa, boş bir değer döndürülür.
ClipboardTransferModes.ORIGINAL_PREFFERED	Varsa bir başvuru döndürülür. Aksi takdirde serileştirilmiş bir kopya döndürülür.
ClipboardTransferModes.CLONE_ONLY	Yalnızca serileştirilmiş bir kopya döndürülür. Serileştirilmiş kopya yoksa, boş bir değer döndürülür.
ClipboardTransferModes.CLONE_PREFFERED	Varsa serileştirilmiş bir kopya döndürülür. Aksi takdirde bir başvuru döndürülür.

Özel veri biçimlerini okuma ve yazma

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Bir nesneyi panoya yazarken, *format* parametresi için `air:` veya `flash:` ayrılmış önekleriyle başlamayan herhangi bir dize kullanabilirsiniz. Nesneyi okumak için biçim olarak aynı dizeyi kullanın. Aşağıdaki örnek nesnelerin panoya nasıl okunacağını ve yazılacağını gösterir:

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

Pano nesnesinden serileştirilmiş bir nesneyi ayıklamak için (bırakma veya yapıştırma işleminden sonra), aynı biçim adını ve `CLONE_ONLY` veya `CLONE_PREFFERED` aktarım modlarını kullanın.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

Başvuru her zaman Clipboard nesnesine eklenir. Pano nesnesinden başvuruyu ayıklamak için (bırakma veya yapıştırma işleminden sonra), serileştirilmiş kopya yerine `ORIGINAL_ONLY` veya `ORIGINAL_PREFFERED` aktarım modlarını kullanın:

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

Başvurular yalnızca Clipboard nesnesi geçerli uygulaması kaynaklıysa geçerlidir. Başvuru varsa başvuruya erişmek için `ORIGINAL_PREFFERED` aktarım modunu, başvuru yoksa serileştirilmiş klonu kullanın.

Gecikmeli oluşturma

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Bir veri biçimini oluşturma hesaplama yönünden pahalıysa, istenilen verileri sağlayan bir işlev sağlayarak gecikmeli oluşturmayı kullanabilirsiniz. Yalnızca bırakma veya yapıştırma işlemi, ertelenmiş biçimde veri istediğinde bu işlev çağrılır.

Oluşturma işlevi bir Clipboard nesnesine `setDataHandler()` yöntemi kullanılarak eklenir. İşlevin verileri uygun biçimde döndürmesi gerekir. Örneğin, `setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)` öğesini çağırdıysanız, `writeText()` işlevi bir dize döndürmelidir.

Aynı türdeki veri biçimi bir Clipboard nesnesine `setData()` yöntemiyle eklenirse, bu veriler gecikmeli sürüme göre önceliğe sahip olacaktır (oluşturma işlevi hiçbir zaman çağrılmaz). Aynı clipboard verilerine ikinci kez erişilirse, oluşturma işlevi çağırılabilir veya çağrılmayabilir.

Not: *Mac OS X işletim sisteminde, ertelenmiş oluşturma yalnızca özel veri biçimleriyle çalışır. Standart veri biçimleriyle, oluşturma işlemi hemen çağrılır.*

Gecikmeli oluşturma işlevi kullanarak metin yapıştırma

Flash Player 10 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örnek gecikmeli oluşturma işlevinin nasıl uygulanacağını gösterir.

Kullanıcı Kopyala düğmesine bastığında, uygulama önceki pano işlemlerinden veri kalmamasını sağlamak için sistem panosunu temizler. Daha sonra `setDataHandler()` yöntemi, pano oluşturucusu olarak `renderData()` işlevini ayarlar.

Kullanıcı, hedef metin alanının bağlam menüsünden Yapıştır komutunu seçtiğinde, uygulama panoya erişir ve hedef metni ayarlar. Panodaki metin verisi biçimi, dize yerine bir işlevle ayarlanmış olduğundan, pano `renderData()` işlevini çağırır. `renderData()` işlevi metni kaynak metinde döndürür, ardından bu metin hedef metne atanır.

Kaynak metni Yapıştır düğmesine basmadan önce düzenlerseniz, düzenleme kopyala düğmesine basıldıktan sonra gerçekleşmiş bile olsa, düzenlemenin yapıştırılan metne yansıtılacağını göz önünde bulundurun. Bunun nedeni, yapıştır düğmesine basılana kadar oluşturma işlevinin kaynak metni kopyalamamasıdır. (Gecikmeli oluşturmayı gerçek bir uygulamada kullanırken, bu sorunun önüne geçmek için kaynak verileri bir şekilde saklamak veya korumak isteyebilirsiniz.)

Flash örneği

```
package {
    import flash.desktop.Clipboard;
    import flash.desktop.ClipboardFormats;
    import flash.desktop.ClipboardTransferMode;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.text.TextFieldType;
    import flash.events.MouseEvent;
    import flash.events.Event;
    public class DeferredRenderingExample extends Sprite
    {
        private var sourceTextField:TextField;
        private var destination:TextField;
        private var copyText:TextField;
        public function DeferredRenderingExample():void
        {
            sourceTextField = createTextField(10, 10, 380, 90);
            sourceTextField.text = "Neque porro quisquam est qui dolorem "
                + "ipsum quia dolor sit amet, consectetur, adipisci velit.";

            copyText = createTextField(10, 110, 35, 20);
            copyText.htmlText = "<a href='#'>Copy</a>";
            copyText.addEventListener(MouseEvent.CLICK, onCopy);

            destination = createTextField(10, 145, 380, 90);
            destination.addEventListener(Event.PASTE, onPaste);
        }
        private function createTextField(x:Number, y:Number, width:Number,
                        height:Number):TextField
        {
            var newTxt:TextField = new TextField();
            newTxt.x = x;
            newTxt.y = y;
            newTxt.height = height;
```

```
            newTxt.width = width;
            newTxt.border = true;
            newTxt.multiline = true;
            newTxt.wordWrap = true;
            newTxt.type = TextFieldType.INPUT;
            addChild(newTxt);
            return newTxt;
        }
        public function onCopy(event:MouseEvent):void
        {
            Clipboard.generalClipboard.clear();
            Clipboard.generalClipboard.setDataHandler(ClipboardFormats.TEXT_FORMAT,
                            renderData);
        }
        public function onPaste(event:Event):void
        {
            sourceTextField.text =
            Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT).toString;
        }
        public function renderData():String
        {
            trace("Rendering data");
            var sourceStr:String = sourceTextField.text;
            if (sourceTextField.selectionEndIndex >
                     sourceTextField.selectionBeginIndex)
            {
                return sourceStr.substring(sourceTextField.selectionBeginIndex,
                                    sourceTextField.selectionEndIndex);
            }
            else
            {
                return sourceStr;
            }
        }
    }
}
```

Flex örneği

```
<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>
```

Bölüm 34: Akselorometre girdisi

Flash Player 10.1 ve üstü, Adobe AIR 2 ve üstü

Accelerometer sınıfı olayları aygıtın hareket alıcısı tarafından algılanan etkinlikleri temel alarak gönderir. Bu veri aygıtın üç boyutlu bir eksen üzerindeki konumunu veya hareketini temsil eder. Aygıt hareket ettiğinde, alıcı bu hareketi algılar ve aygıtın ivme koordinatlarını döndürür. Accelerometer sınıfı akselerometrenin desteklenip desteklenmediğini sorgulayacak ve ayrıca ivme olaylarının gönderilme hızını ayarlayacak yöntemler sağlar.

Akselerometre eksenleri aygıtın fiziksel yönlendirmesine göre değil ekranın yönlendirmesine göre normalleştirilir. Aygıt ekranı yeniden yönlendirdiğinde akselerometre eksenleri de yeniden yönlendirilir. Telefonun hangi yöne döndürüldüğüne bakılmaksızın, kullanıcı telefonu normal, dik bir görüntüleme konumunda tutarken y ekseni yaklaşık olarak her zaman dikeydir. Örneğin, bir tarayıcıdaki SWF içeriği tam ekran modundayken otomatik yönlendirme kapalıysa, aygıt döndürüldükçe akselerometre eksenleri yeniden yönlendirilmez.

Daha fazla Yardım konusu

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

Akselerometre desteğini kontrol etme

Çalışma zamanı ortamının bu özelliği kullanabilme becerisini sınamak için `Accelerometer.isSupported` özelliğini kullanın:

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

Her bir API girişi için listelenen çalışma zamanı sürümleri, Accelerometer sınıfına ve bu sınıfın üyelerine erişebilir. Buna karşın, çalışma zamanındaki geçerli medya bu özelliğin kullanılabilirliğini belirler. Örneğin, Flash Player 10.1 için Accelerometer sınıfı özelliklerini kullanarak kod derleyebilirsiniz, ancak kullanıcının aygıtında Accelerometer özelliğinin kullanılabilirliğini sınamak için `Accelerometer.isSupported` özelliğini kullanmanız gerekir. `Accelerometer.isSupported` çalışma zamanında `true` ise, Accelerometer desteği o sırada mevcut demektir.

Akselerometre değişikliklerini saptama

Akselerometre sensörünü kullanmak için, Accelerometer nesnesini somutlaştırın ve gönderdiği `update` olaylarına kaydolun. `update` olayı bir Accelerometer olayı nesnesidir. Bu olay dört özelliğe sahiptir ve bunların her biri sayıdır:

- `accelerationX`—g cinsinden ölçülen, x ekseni boyunca ivme. X ekseni, aygıt dik konumdayken aygıtın solundan sağına doğru gider. (Cihazın tepesi dönükken cihaz dik konumdadır.) Aygıt sağa doğru hareket ederse, ivme pozitiftir.

- `accelerationY`—g cinsinden ölçülen, y ekseni boyunca ivme. Y ekseni, aygıt dik konumdayken aygıtın üstünden altına doğru gider. (Cihazın tepesi dönükken cihaz dik konumdadır.) Aygıt bu eksene göre yukarı doğru hareket ederse, ivme pozitiftir.
- `accelerationZ`—g cinsinden ölçülen, z ekseni boyunca ivme. Z ekseni, aygıtın önüne dikey olarak gider. Aygıtı, yüzü yukarıya dönecek şekilde hareket ettirirseniz, ivme pozitiftir. Aygıtın yüzü zemine bakıyorsa, ivme negatiftir.
- `timestamp`—Çalışma zamanının başlatılmasından olay zamanına kadar geçen milisaniye sayısı.

1 g, yerçekiminden kaynaklanan standart ivmedir ve yaklaşık 9,8 $m/sn^{2.}$ değerine karşılık gelir.

Aşağıda bir metin alanında akselerometre verilerini gösteren basit bir örnek verilmektedir:

```
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()
}
```

Bu örneği kullanmak için, `accTextField` metin alanını oluşturup bu kodu kullanmadan önce ekran listesine eklemeyi unutmayın.

Accelerometer nesnesinin `setRequestedUpdateInterval()` yöntemini çağırarak akselerometre için istenilen zaman aralığını ayarlayabilirsiniz. Bu yöntemin çağırdığı tek parametre, milisaniye cinsinden talep edilen güncelleme aralığı olan `interval` parametresidir:

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

Akselerometre güncellemeleri arasındaki asıl süre bu değerden daha büyük veya daha az olabilir. Güncelleme aralığındaki herhangi bir değişiklik tüm kayıtlı dinleyicileri etkiler. `setRequestedUpdateInterval()` yöntemini çağırmazsanız, uygulama aygıtın varsayılan aralığına göre güncelleme alır.

Akselerometre verileri bir miktar hata payına sahiptir. Verileri düzeltmek için yeni verilerin hareketli bir ortalamasını kullanabilirsiniz. Örneğin, aşağıdaki örnekte akselerometre okumaları yuvarlak bir sonuç elde etmek üzere geçerli okumayla çarpılır:

```
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));
}
```

Ancak, bu hareketli ortalama yalnızca akselerometre güncelle aralığı küçükse arzu edilir niteliktedir.

Bölüm 35: AIR uygulamasında taşıma ve bırakma

Adobe AIR 1.0 ve üstü

Kullanıcı arabirimi sürükle ve bırak hareketlerini desteklemek için, Adobe ®AIR™'de bulunan sürükle ve bırak API'sindeki sınıfları kullanın. Bu anlamda *hareket* kullanıcı tarafından gerçekleştirilen, hem işletim sistemi hem de uygulamanız tarafından aracı olunan, bilgi kopyalamak, taşımak veya bağ vermek için bir amaç ifade eden bir eylemdir. *Dışarı sürükleme* hareketi, kullanıcı bir nesneyi bileşenin veya uygulamanın dışına sürüklediğinde oluşur. *Dışarı sürükleme* hareketi, kullanıcı bir nesneyi bileşenin veya uygulamanın dışından içeri sürüklediğinde oluşur.

Sürükle ve bırak API'siyle kullanıcının uygulamalar arasında ve bir uygulama içerisindeki bileşenler arasında veri sürüklemesini sağlayabilirsiniz. Desteklenen aktarım biçimleri şunlardır:

- Bitmap'ler
- Dosyalar
- HTML biçimli metin
- Metin
- Zengin Metin Biçimli veriler
- URL'ler
- Dosya uzantıları
- Serileştirilmiş nesneler
- Nesne başvuruları (yalnızca kaynak uygulama içinde geçerli)

AIR'de sürükle ve bırak temelleri

Adobe AIR 1.0 ve üstü

Bir AIR uygulamasında sürükle ve bırakı kullanmakla ilgili hızlı bir açıklama ve kod örnekleri için Adobe Geliştirici Bağlantısında bulunan aşağıdaki hızlı başlangıç makalelerini inceleyin:

- Sürükle/bırak ve kopyala/yapıştır desteği (Flex)
- Sürükle/bırak ve kopyala/yapıştır desteği (Flash)

Sürükle ve bırak API'si aşağıdaki sınıfları içerir.

Paket	Sınıflar
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise Sürükle ve bırak API'siyle kullanılan sabitlerin tanımlandığı sabitler: • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

Sürükle ve bırak hareket sahneleri

Sürükle ve bırak hareketinin üç sahnesi vardır:

Başlatma *Kullanıcı, fare düğmesini basılı tutarak bir bileşeni veya bileşendeki öğeyi sürükleyerek sürükle ve bırak işlemini başlatır.* Sürüklenen öğenin kaynağı olan bileşen genellikle sürükleme başlatıcısı olarak belirlenir ve `nativeDragStart` ve `nativeDragComplete` olaylarını gönderir. Adobe® AIR uygulaması, `mouseDown` veya `mouseMove` olayına yanıt olarak `NativeDragManager.doDrag()` yöntemini çağırarak bir sürükleme işlemi başlatır.

Sürükleme işlemi AIR uygulamasının dışından başlatıldıysa `nativeDragStart` veya `nativeDragComplete` olaylarını gönderecek başlatıcı nesne bulunmaz.

Sürükleme *Fare düğmesini basılı tutarken kullanıcı fare imlecini başka bir bileşene, uygulamaya veya masaüstüne taşır.*Sürükleme işlemi devam ederken, başlatıcı nesne `nativeDragUpdate` olaylarını gönderir. (Ancak bu olay Linux için AIR'de gönderilmez) Kullanıcı fareyi AIR uygulamasındaki olası bir bırakma hedefinin üzerine taşıdığında, bırakma hedefi bir `nativeDragEnter` olayı gönderir. Olay işleyicisi, sürüklenen verilerin hedef tarafından kabul edilecek bir biçimde mevcut olup olmadığını belirlemek için olay nesnesini denetleyebilir, mevcutsa `NativeDragManager.acceptDragDrop()` yöntemini çağırarak kullanıcının verileri bırakmasını sağlar.

Sürükleme hareketi etkileşimli bir nesnenin üzerinde kaldığı sürece, bu nesne `nativeDragOver` olaylarını gönderir. Sürükleme hareketi etkileşimli nesneyi bıraktığında, bir `nativeDragExit` olayı gönderir.

Düşür *Kullanıcı fare düğmesini uygun bir bırakma hedefinin üzerinde serbest bırakır.* Hedef bir AIR uygulaması veya bileşeni olduğunda, hedef nesne bir `nativeDragDrop` olayı gönderir. Olay işleyicisi olay nesnesinden aktarılan verilere erişebilir. Hedef, AIR'in dışındaysa, bırakma işlemini işletim sistemi veya başka bir uygulama yapar. Her iki durumda da, başlatan nesne bir `nativeDragComplete` olayı gönderir (sürükleme AIR'in içinden başlatıldıysa).

NativeDragManager sınıfı hem içeri hem de dışarı sürükleme hareketlerini kontrol eder. NativeDragManager sınıfının tüm üyeleri statiktir, bu sınıfın bir örneğini oluşturmayın.

Clipboard nesnesi

Bir uygulamanın veya bileşenin içine ya da dışına sürüklenen veriler bir Clipboard nesnesinde bulunmaktadır. Tek bir Clipboard nesnesi, başka bir uygulamanın verileri anlayıp kullanabilmesi olasılığını arttırmak için aynı bilgilerin farklı temsillerini kullanılabilir hale getirebilir. Örneğin bir görüntü, görüntü verisi, serileştirilmiş Bitmap nesnesi veya bir dosya olarak dahil edilebilir. Bir biçimde veri oluşturma işlemi, veriler okunana kadar çağrılmayan bir oluşturma işlevine ertelenebilir.

Bir sürükleme hareketi başladıktan sonra Clipboard nesnesine yalnızca `nativeDragEnter`, `nativeDragOver` ve `nativeDragDrop` olayları için olay işleyicisinin içinden erişilebilir. Sürükleme hareketi bittikten sonra Clipboard nesnesi okunamaz veya yeniden kullanılamaz.

Bir uygulama nesnesi başvuru olarak ve serileştirilmiş bir nesne olarak aktarılabilir. Başvurular yalnızca kaynak uygulama içinde geçerlidir. Serileştirilmiş nesne aktarımları AIR uygulamaları arasında geçerlidir, ancak yalnızca serileştirildiğinde ve serileştirmesi kaldırıldığında geçerliliği devam eden nesnelerle kullanılabilir. Serileştirilen nesneler dize tabanlı veri aktarım biçimi olan ActionScript 3 (AMF3) için Action Message Format'a dönüştürülür.

Flex çerçevesiyle çalışma

Çoğu durumda, Flex uygulamaları oluştururken Adobe® Flex™ sürükle ve bırak API'sini kullanmak daha yararlıdır. AIR'de bir Flex uygulaması çalıştırıldığında (AIR NativeDragManager öğesini dahili şekilde kullanır), Flex çerçevesi eşdeğer bir özellik kümesi sağlar. Flex ayrıca, bir uygulama veya bileşen daha kısıtlayıcı bir tarayıcı ortamında çalışırken, daha sınırlı bir özellik setini korur. AIR sınıfları, AIR çalışma zamanı ortamının dışında çalışan bileşenlerde veya uygulamalarda kullanılamaz.

Dışarı sürükleme hareketini destekleme

Adobe AIR 1.0 ve üstü

Dışarı sürükleme hareketini desteklemek için, `mouseDown` olayına yanıt olarak bir Clipboard nesnesi oluşturmalı ve `NativeDragManager.doDrag()` yöntemine göndermelisiniz. Ardından uygulamanız, kullanıcı hareketi tamamladığında veya iptal ettiğinde ne yapacağını belirlemek için başlatan nesnedeki `nativeDragComplete` olayını dinleyebilir.

Verileri aktarım için hazırlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Verileri veya bir nesneyi sürüklemeye hazırlamak için, bir Clipboard nesnesi oluşturun ve bir veya birden fazla biçimde aktarılacak bilgileri ekleyin. Otomatik olarak yerel pano biçimlerine çevrilebilecek verileri iletmek için standart veri biçimlerini ve nesneleri iletmek için uygulama tanımlı biçimleri kullanabilirsiniz.

Belirli bir biçime aktarılacak bilgileri dönüştürmek hesaplama yönünden pahalıysa, dönüştürmeyi gerçekleştirmesi için bir işleyici işlevinin adını verebilirsiniz. Yalnızca alıcı bileşen veya uygulama ilişkilendirilen biçimi okursa işlev çağrılır.

Pano biçimleri hakkında daha fazla bilgi edinmek için bkz. “Pano veri biçimleri” sayfa 605.

Aşağıdaki örnek birçok biçimde bitmap içeren bir Clipboard nesnesinin nasıl oluşturulacağını gösterir: bir Bitmap nesnesi, bir yerel bitmap biçimi ve bitmap'in ilk yüklendiği dosyayı içeren bir dosya listesi biçimi:

```
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;
}
```

Dışarı sürükleme işlemi başlatma

Adobe AIR 1.0 ve üstü

Bir sürükleme işlemi başlatmak için, mouse down olayına yanıt olarak `NativeDragManager.doDrag()` yöntemini çağırın. `doDrag()` yöntemi, aşağıdaki parametreleri alan statik bir yöntemdir:

Parametre	Açıklama
initiator	Sürükleme işleminin kaynaklandığı ve `dragStart` ve `dragComplete` olaylarını gönderen nesne. Başlatıcı etkileşimli bir nesne olmalıdır.
clipboard	Aktarılacak verileri içeren Clipboard nesnesi. Clipboard nesnesine, sürükleme ve bırakma sırası boyunca gönderilen NativeDragEvent nesnelerinde başvurulur.
dragImage	(İsteğe bağlı) Sürükleme sırasında görüntülenecek bir BitmapData nesnesi. Görüntü bir `alpha` değeri belirtebilir. (Not: Microsoft Windows görüntüleri sürüklemek için her zaman sabit alfa soldurma uygular.)
offset	(İsteğe bağlı) Sürükleme görüntüsünün fare etkin noktasından uzaklığını belirten Point nesnesi. Sürükleme görüntüsünü fare imlecine göre yukarı ve sola taşımak için negatif koordinatlar kullanın. Uzaklık sağlanmamışsa, sürükleme görüntüsünün sol üst köşesi farenin etkin noktasında konumlandırılmıştır.
actionsAllowed	(İsteğe bağlı) Sürükleme işlemi için hangi eylemlerin (kopyala, taşı veya bağla) geçerli olduğunu belirten bir NativeDragOptions nesnesi. Hiç argüman sağlanmamışsa, tüm eylemlere izin verilir. Olası bir sürükleme hedefinin izin verilen eylemlerin hedef bileşenin amacına uygun olup olmadığını kontrol etmesini sağlamak için NativeDragEvent nesnelerinde DragOptions nesnesine başvurulur. Örneğin “trash” bileşeni yalnızca taşıma eylemine izin veren sürükleme hareketlerini kabul edebilir.

Aşağıdaki örnek bir dosyadan yüklenen bitmap nesnesi için sürükleme işleminin nasıl başlatılacağını anlatır. Örnek, bir görüntü yükler ve bir `mouseDown` olayında sürükleme işlemini başlatır.

```
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);
    }
}
}
```

Dışarı sürükleme aktarımını tamamlama

Adobe AIR 1.0 ve üstü

Kullanıcı sürüklenen öğeyi fare düğmesini serbest bırakarak bıraktığında, başlatıcı nesne bir `nativeDragComplete` olayı gönderir. Olay nesnesinin `dropAction` özelliğini kontrol edip uygun eylemi gerçekleştirebilirsiniz. Örneğin eylem `NativeDragAction.MOVE`, olduğunda, kaynak öğeyi orijinal konumundan kaldırabilirsiniz. Kullanıcı, imleç uygun bir bırakma hedefinin dışındayken, fare düğmesini serbest bırakarak sürükleme hareketini iptal edebilir. Sürükleme yöneticisi iptal edilmiş bir hareket için `dropAction` özelliğini `NativeDragAction.NONE` olarak ayarlar.

İçeri sürükleme hareketini destekleme

Adobe AIR 1.0 ve üstü

İçeri sürükleme hareketini desteklemek için, uygulamanız (genelde uygulamanızın görsel bir bileşeni) `nativeDragEnter` veya `nativeDragOver` olaylarına yanıt vermelidir.

Genel bir bırakma işleminin adımları

Adobe AIR 1.0 ve üstü

Bırakma işlemi için tipik olay sırası aşağıdaki gibidir:

1. Kullanıcı clipboard nesnesini bir bileşenin üzerine sürükler.
2. Bileşen bir `nativeDragEnter` olayı gönderir.
3. `nativeDragEnter` olay işleyici, kullanılabilir veri biçimlerini ve izin verilen eylemleri kontrol etmek için olay nesnesini inceler. Bileşen bırakmayı işleyebilirse, `NativeDragManager.acceptDragDrop()` öğesini çağırır.
4. NativeDragManager fare imlecini, nesnenin bırakıldığını gösterecek şekilde değiştirir.
5. Kullanıcı nesneyi bileşenin üzerine bırakır.
6. Alıcı bileşen bir `nativeDragDrop` olayı gönderir.
7. Alıcı bileşen, istenen biçimdeki verileri olay nesnesinin içindeki Clipboard nesnesinden okur.
8. Sürükleme hareketi bir AIR uygulamasının içinden kaynaklandıysa, başlatan etkileşimli nesne bir `nativeDragComplete` olayı gönderir. Hareket AIR uygulamasının dışından kaynaklandıysa, geribildirim gönderilmez.

Bir içeri sürükleme hareketini onaylama

Adobe AIR 1.0 ve üstü

Kullanıcı bir pano öğesini görsel bir bileşenin sınırlarının içine sürüklediğinde, bileşen `nativeDragEnter` ve `nativeDragOver` olaylarını gönderir. Bileşenin pano öğesini kabul edip edemeyeceğini belirlemek için, bu olayların işleyicileri, olay nesnesinin `clipboard` ve `allowedActions` özelliklerini kontrol edebilir. Olay işleyici, bileşenin bırakma işlemini kabul edebileceğini bildirmek için alıcı bileşene bir başvuru ileterek `NativeDragManager.acceptDragDrop()` yöntemini çağırmalıdır. Birden fazla kayıtlı olay dinleyicisi `acceptDragDrop()` yöntemini çağırırsa, listedeki son işleyici önceliğe sahiptir. Fare kabul eden nesnenin sınırlarından çıkıp `nativeDragExit` olayını tetikleyene kadar, `acceptDragDrop()` çağrısı geçerliliğini korur.

`doDrag()` öğesine iletilen `allowedActions` parametresinde birden fazla eyleme izin veriliyorsa, kullanıcı bir değiştirici tuşunu basılı tutarak izin verilen eylemlerden hangilerini gerçekleştirmeyi düşündüğünü belirtebilir. Sürükleme yöneticisi, kullanıcıya, bırakma işlemini tamamlasaydı hangi eylemin gerçekleşeceğini göstermek üzere imleç görüntüsünü değiştirir. İstenen eylem, NativeDragEvent nesnesinin `dropAction` özelliği tarafından bildirilir. Sürükleme hareketi için eylem kümesi yalnızca tavsiye amaçlıdır. Aktarımda yer alan bileşenlerin uygun davranışı uygulaması gereklidir. Örneğin bir taşıma eylemini tamamlamak için, sürükleme başlatıcısı sürüklenen öğeyi sürükleyebilir ve bırakma hedefi bunu ekleyebilir.

Sürükleme hedefiniz NativeDragManager sınıfının `dropAction` özelliğini ayarlayarak, sürükleme eylemini üç olası eylemden biriyle sınırlayabilir. Kullanıcı klavyeyi kullanarak farklı bir eylem seçmeyi denerse, NativeDragManager *unavailable* imleci görüntüler. Hem `nativeDragEnter` hem de `nativeDragOver` olayları için, işleyicilerdeki `dropAction` özelliğini ayarlayın.

Aşağıdaki örnek, bir `nativeDragEnter` veya `nativeDragOver` olayı için olay işleyicisini gösterir. Sürüklenen pano metin biçiminde veriler içeriyorsa, bu işleyici yalnızca içeri sürükleme hareketini kabul eder.

```
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
    }
}
```

Bırakmayı tamamlama

Adobe AIR 1.0 ve üstü

Kullanıcı sürüklenen öğeyi hareketi kabul eden etkileşimli bir nesnenin üzerine bıraktığında, etkileşimli nesne bir `nativeDragDrop` olayı gönderir. Bu olayın işleyicisi verileri olay nesnesinin `clipboard` özelliğinden ayıklayabilir.

Pano uygulama tanımlı bir biçim içerdiğinde, Clipboard nesnesinin `getData()` yöntemine iletilen `transferMode` parametresi sürükleme yöneticisinin bir başvuruyu mu, yoksa nesnenin serileştirilmiş bir sürümünü mü döndürdüğünü belirler.

Aşağıdaki örnek `nativeDragDrop` olayı için bir olay işleyicisini gösterir:

```
 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));
}
```

Olay işleyicisinden çıkıldıktan sonra, Clipboard nesnesi geçerliliğini kaybeder. Nesneye veya verilerine erişme girişimi, hata oluşmasına neden olur.

Bir bileşenin görünümünü güncelleme

Adobe AIR 1.0 ve üstü

Bir bileşen görünümünü NativeDragEvent olaylarına dayalı olarak güncelleyebilir. Aşağıdaki tablo tipik bir bileşenin farklı olaylara yanıt olarak yapacağı değişiklikleri açıklar:

Olay	Açıklama
nativeDragStart	Başlatan etkileşimli nesne `nativeDragStart` olayını, sürükleme hareketinin bu etkileşimli nesneden kaynaklandığına dair görsel geribildirim sağlamak için kullanabilir.
nativeDragUpdate	Başlatan etkileşimli nesne, nativeDragUpdate olayını, hareket sırasında durumunu güncellemek için kullanabilir. (Bu olay Linux için AIR'de bulunmaz)
nativeDragEnter	Olası alıcı etkileşimli nesne bu olayı odağı kendine çekmek veya bırakma işlemini kabul edip edemeyeceğini görsel olarak belirtmek için kullanabilir.

Olay	Açıklama
nativeDragOver	Olası alıcı etkileşimli nesne bu olayı, farenin sokak haritası görüntüsü gibi karmaşık bir bileşenin "sıcak" bölgesine girmesi gibi farenin etkileşimli nesne içerisindeki hareketine yanıt vermek için kullanabilir.
nativeDragExit	Olası alıcı etkileşimli nesne bu olayı, bir sürükleme hareketi sınırlarının dışına taşındığında durumunu geri yüklemek için kullanabilir.
nativeDragComplete	Alıcı etkileşimli nesne bu olayı, listeden bir öğeyi kaldırmak gibi ilişkilendirilen veri modelini güncellemek veya görsel durumunu geri yüklemek için kullanabilir.

İçeri sürükleme hareketi sırasında fare konumunu izleme

Adobe AIR 1.0 ve üstü

Sürükleme hareketi bir bileşenin üzerinde kaldığında, bu bileşen `nativeDragOver` olaylarını gönderir. Bu olaylar her birkaç milisaniyede bir ve farenin her hareketinde gönderilir. `nativeDragOver` olay nesnesi, farenin bileşen üzerindeki konumunu belirlemek için kullanılabilir. Fare konumuna erişimi olmak, alıcı bileşenin karmaşık olduğu, ancak alt bileşenlerden oluşmadığı durumlarda yararlı olabilir. Örneğin, uygulamanız bir sokak haritası içeren bir bitmap görüntülediyse ve kullanıcı içlerine bilgi sürüklediğinde haritadaki bölgelerin vurgulanmasını istediyseniz, harita içinde fare konumunu izlemek için `nativeDragOver` olayında bildirilen fare koordinatlarını kullanabilirsiniz.

HTML'de sürükle ve bırak

Adobe AIR 1.0 ve üstü

HTML tabanlı bir uygulamaya veya uygulamadan (veya bir HTMLLoader öğesinde görüntülenen HTML'e veya HTML'den) veri sürüklemek için, HTML drag ve drop olaylarını kullanabilirsiniz. HTML sürükle ve bırak API'si sayesinde HTML içeriğindeki DOM öğelerine ve bu öğelerden veri sürükleyebilirsiniz.

Not: *Ayrıca, HTML içeriğine sahip HTMLLoader nesnesindeki olayları dinleyerek de AIR NativeDragEvent ve NativeDragManager API'lerini kullanabilirsiniz. Ancak HTML API'si HTML DOM ile daha başarılı bir şekilde entegredir ve size, varsayılan davranışın kontrolünü sağlar.*

Varsayılan sürükle ve bırak davranışı

Adobe AIR 1.0 ve üstü

HTML ortamı metin, görüntüler ve URL'leri de içeren sürükle ve bırak hareketleri için varsayılan davranış sağlar. Varsayılan davranışı kullanarak her zaman bu tür verileri bir öğenin dışına sürükleyebilirsiniz. Ancak, bir öğeye yalnızca metin sürükleyebilirsiniz ve yalnızca sayfanın düzenlenebilir bir bölgesinde bulunan öğelere sürükleyebilirsiniz. Bir sayfanın düzenlenebilir bölgeleri arasında veya içinde metni sürüklediğinizde, varsayılan davranış bir taşıma eylemi gerçekleştirir. Düzenlenemeyen bir bölgeden veya uygulamanın dışından düzenlenebilir bölgeye metin sürüklediğinizde, varsayılan davranış bir kopyalama eylemi gerçekleştirir.

drag-and-drop olaylarını kendiniz işleyerek varsayılan davranışı geçersiz kılabilirsiniz. Varsayılan davranışı iptal etmek için, drag-and-drop olayları için gönderilen nesnelerin `preventDefault()` yöntemlerini çağırmalısınız. Seçilen eylemi gerçekleştirmek için gerekli olduğundan, verileri bırakma hedefine ekleyebilir ve sürükleme kaynağından kaldırabilirsiniz.

Varsayılan olarak kullanıcı istediği metni seçip sürükleyebilir ve görüntüleri ve bağları sürükleyebilir. `-webkit-user-select` WebKit CSS özelliğini, bir HTML öğesinin nasıl seçilebileceğini kontrol etmek için kullanabilirsiniz. Örneğin `-webkit-user-select` öğesini `none` olarak ayarlarsanız, öğe içerikleri seçilebilir olmaz ve sürüklenemez. `-webkit-user-drag` CSS özelliğini, bir öğenin bütün olarak sürüklenip sürüklenemeyeceğini kontrol etmek için de kullanabilirsiniz. Ancak öğenin içerikleri ayrı öğeler olarak kabul edilir. Kullanıcı hala metnin seçili bir kısmını sürükleyebilir. Daha fazla bilgi için bkz. "AIR'de CSS" sayfa 987.

HTML'de sürükleme ve bırakma olayları

Adobe AIR 1.0 ve üstü

Bir sürükleme işleminin kaynaklandığı başlatıcı öğe tarafından gönderilen olaylar şunlardır:

Olay	Açıklama
dragstart	Kullanıcı sürükleme hareketini başlattığında gönderilir. Bu olayın işleyicisi gerekirse olay nesnesinin preventDefault() yöntemini çağırarak sürüklemeyi engelleyebilir. Sürüklenen verilerin kopyalanabildiğini, verilere bağ verilebildiğini veya taşınabildiğini kontrol etmek için, effectAllowed özelliğini ayarlayın. Seçilen metin, görüntüler ve bağlantılar varsayılan davranış tarafından panoya konur, ancak olay nesnesinin dataTransfer özelliğini kullanarak sürükleme hareketi için farklı veriler ayarlayabilirsiniz.
drag	Sürükleme hareketi boyunca kesintisiz gönderilir.
dragend	Kullanıcı sürükleme hareketini bitirmek için fare düğmesini serbest bıraktığında gönderilir.

Sürükleme hedefi tarafından gönderilen olaylar şunlardır:

Olay	Açıklama
dragover	Sürükleme hareketi öğenin sınırları içerisinde kaldığı sürece kesintisiz gönderilir. Bu olayın işleyicisi dataTransfer.dropEffect özelliğini, kullanıcı fare düğmesini serbest bırakırsa bırakmanın kopyalama, taşıma veya bağ verme eylemlerinden hangisiyle sonuçlanacağını göstermek için ayarlamalıdır.
dragenter	Sürükleme hareketi öğenin sınırlarına girdiğinde gönderilir. Bir dragenter olay işleyicisindeki dataTransfer nesnesinin herhangi bir özelliğini değiştirirseniz, bu değişiklikler bir sonraki dragover olayı tarafından hızla geçersiz kılınır. Diğer yandan, bir dragenter ve ilk dragover olayı arasında, farklı özellikler ayarlanmışsa imlecin yanıp sönmesine neden olabilecek kısa bir gecikme yaşanır. Birçok durumda, her iki olay için de aynı olay işleyicisini kullanabilirsiniz.
dragleave	Sürükleme hareketi öğenin sınırlarından çıktığında gönderilir.
drop	Kullanıcı verileri öğeye bıraktığında gönderilir. Sürüklenen verilere yalnızca bu olayın işleyicisinde erişilebilir.

Bu olaylara yanıt olarak gönderilen olay nesnesi, mouse olayına benzerdir. (`clientX`, `clientY`) ve (`screenX`, `screenY`) gibi mouse olayı özelliklerini, fare konumunu belirlemek için kullanabilirsiniz.

Bir drag olayı nesnesinin en önemli özelliği, sürüklenen verileri içeren `dataTransfer` öğesidir. `dataTransfer` nesnesi aşağıdaki özelliklere ve yöntemlere sahiptir:

Özellik veya Yöntem	Açıklama
effectAllowed	Sürüklemenin kaynağı tarafından izin verilen efekt. Genellikle dragstart olayının işleyicisi bu değeri ayarlar. Bkz. "HTML'de sürükleme efektleri" sayfa 624.
dropEffect	Hedef veya kullanıcı tarafından seçilen efekt. `dragover` veya `dragenter` olay işleyicisinde `dropEffect` öğesini ayarlarsanız, AIR, fare imlecini, kullanıcı fare düğmesini serbest bıraktığında oluşacak efekti gösterecek şekilde günceller. `dropEffect` kümesi izin verilen efektlerden biriyle eşleşmezse, bırakma işlemine izin verilmez ve *kullanılamaz* imleci görüntülenir. En son `dragover` veya `dragenter` olayına yanıt olarak bir `dropEffect` ayarlamadıysanız, kullanıcı standart işletim sistemi değiştirici tuşlarını kullanarak izin verilen efektler arasından seçim yapabilir. Son efekt, `dragend` için gönderilen nesnenin `dropEffect` özelliği tarafından bildirilir. Kullanıcı fare düğmesini uygun hedefin dışında serbest bırakarak bırakma işlemini iptal ederse, `dropEffectnone` olarak ayarlanır.
türler	`dataTransfer` nesnesinde mevcut olan her veri biçimi için MIME türü dizeleri içeren bir dizi.
getData(mimeType)	`mimeType` parametresi tarafından belirtilen biçimdeki verileri alır. `getData()` yöntemi yalnızca `drop` olayına yanıt olarak çağrılabilir.
setData(mimeType)	`dataTransfer` öğesine mimeType parametresi tarafından belirtilen biçimdeki verileri ekler. Her MIME türü için `setData()` öğesini çağırarak birden çok biçimde veri ekleyebilirsiniz. Varsayılan sürükleme davranışı tarafından `dataTransfer` nesnesine yerleştirilen tüm veriler temizlenir. `setData()` yöntemi yalnızca `dragstart` olayına yanıt olarak çağrılabilir.
clearData(mimeType)	`mimeType` parametresi tarafından belirtilen biçimdeki tüm verileri temizler.
setDragImage(image, offsetX, offsetY)	Özel bir sürükleme görüntüsü ayarlar. `setDragImage()` yöntemi yalnızca bir dragstart olayına yanıt olarak ve bir HTML öğesi onun `-webkit-user-drag` CSS stili `element` öğesine ayarlanarak taşındığında çağrılabilir. `image` parametresi bir JavaScript öğesi veya Image nesnesi olabilir.

HTML sürükle ve bırak için MIME türleri

Adobe AIR 1.0 ve üstü

Bir HTML drag-and-drop olayının `dataTransfer` nesnesiyle kullanılacak MIME türleri şunlardır:

Veri biçimi	MIME türü
Metin	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Nokta Esaslı	"image/x-vnd.adobe.air.bitmap"
Dosya listesi	"application/x-vnd.adobe.air.file-list"

Uygulama tanımlı dizeler de dahil olmak üzere diğer MIME dizelerini de kullanabilirsiniz. Ancak diğer uygulamalar aktarılan verileri tanımayabilir veya kullanmayabilir. `dataTransfer` nesnesine beklenen biçimde veriler eklemek sizin sorumluluğunuzdadır.

Önemli: *Yalnızca uygulama sanal alanında kod çalıştırma, bırakılan dosyalara erişebilir. Uygulama dışı sanal alanda bir File nesnesinin herhangi bir özelliğini okuma veya ayarlama girişiminde bulunmak bir güvenlik hatasına neden olur. Daha fazla bilgi için bkz. "Uygulama dışı HTML sanal alanlarındaki dosya bırakmaları işleme" sayfa 628.*

HTML'de sürükleme efektleri

Adobe AIR 1.0 ve üstü

Sürükleme hareketinin başlatıcısı, `dragstart` olayının işleyicisindeki `dataTransfer.effectAllowed` özelliğini ayarlayarak izin verilen sürükleme efektlerini sınırlandırabilir. Aşağıdaki dize değerleri kullanılabilir:

Dize değeri	Açıklama
"none"	Hiçbir sürükleme hareketine izin verilmez.
"copy"	Orijinali yerinde kalarak veriler hedefe kopyalanır.
"link"	Orijinaline giden bir bağ kullanılarak veriler bırakma hedefiyle paylaşılır.
"move"	Veriler hedefe kopyalanır ve orijinal konumundan kaldırılır.
"copyLink"	Veriler kopyalanabilir veya verilere bağ verilebilir.
"copyMove"	Veriler kopyalanabilir veya taşınabilir.
"linkMove"	Verilere bağ verilebilir veya veriler taşınabilir.
"all"	Veriler kopyalanabilir, taşınabilir veya verilere bağ verilebilir. *Tümü*, varsayılan davranışı engellediğinizde oluşan varsayılan efekttir.

Sürükleme hareketinin hedefi, kullanıcı bırakmayı tamamlarsa gerçekleştirilen eylemi göstermek için `dataTransfer.dropEffect` özelliğini ayarlayabilir. Bırakma efekti izin verilen efektlerden biriyse, sistem uygun kopyala, taşı veya bağ ver imlecini görüntüler. Değilse, sistem *kullanılamaz* imlecini görüntüler. Hedef tarafından bırakma efekti ayarlanmamışsa, kullanıcı değiştirici tuşlarını kullanarak izin verilen eylemler arasından seçim yapılabilir.

Hem `dragover` hem de `dragenter` olaylarının işleyicilerindeki `dropEffect` değerini ayarlayın:

```
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";
}
```

Not: *`dragenter` için işleyicideki `dropEffect` özelliğini her zaman ayarlamanız gerekse de, sonraki `dragover` olayının özelliği varsayılan değere sıfırladığını unutmayın. Her iki olaya yanıt olarak da `dropEffect` öğesini ayarlayın.*

Bir HTML öğesinden veri sürüklemek

Adobe AIR 1.0 ve üstü

Varsayılan davranış, bir HTML sayfasındaki içeriğin çoğunun sürüklenerek kopyalanmasına izin verir. Sürüklenmesine izin verilen içeriği, `-webkit-user-select` ve `-webkit-user-drag` CSS özelliklerini kullanarak kontrol edebilirsiniz.

`dragstart` olayı için işleyicideki varsayılan dışarı sürükleme davranışını geçersiz kılın. Sürükleme hareketine kendi verilerinizi koymak için, olay nesnesinin `dataTransfer` özelliğinin `setData()` yöntemini çağırın.

Varsayılan davranışa bağlı olmadığınızda bir kaynak nesnenin hangi sürükleme efektlerini desteklediğini belirtmek için, `dragstart` olayı için gönderilen olay nesnesinin `dataTransfer.effectAllowed` özelliğini ayarlayın. İstediğiniz efekt kombinasyonunu seçebilirsiniz. Örneğin bir kaynak öğesi hem *kopyalama* hem de *bağ verme* efektlerini destekliyorsa, özelliği `"copyLink"` olarak ayarlayın.

Sürüklenen verileri ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`dataTransfer` özelliğini kullanarak sürükleme hareketi için verileri `dragstart` olayının işleyicisine ekleyin. Panoya veri koymak için, aktarılacak MIME türünü ve verileri ileterek `dataTransfer.setData()` yöntemini kullanın.

Örneğin uygulamanızda *imageOfGeorge* kimliğine sahip bir görüntü öğesi varsa, aşağıdaki dragstart olay işleyicisini kullanabilirsiniz. Bu örnek George'un bir resminin birçok veri biçimindeki temsillerini ekler, bu da diğer uygulamaların sürüklenen verileri kullanabilme olasılığını artırır.

```
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));
}
```

Not: *`dataTransfer` nesnesinin `setData()` yöntemini çağırdığınızda, varsayılan sürükle ve bırak davranışı tarafından hiç veri eklenmez.*

Bir HTML öğesine veri sürükleme

Adobe AIR 1.0 ve üstü

Varsayılan davranış metnin yalnızca sayfanın düzenlenebilir bölgelerine sürüklenmesine izin verir. Öğenin açılış etiketine `contenteditable` niteliği eklenerek bir öğenin ve alt öğelerinin düzenlenebilir hale getirilebileceğini belirtebilirsiniz. Ayrıca, belge nesnesi `designMode` özelliğini `"on"` yaparak bir belgenin tamamını düzenlenebilir hale getirebilirsiniz.

Sürüklenen verileri kabul edebilen herhangi bir öğe için `dragenter`, `dragover` ve `drop` olaylarını işleyerek, bir sayfada alternatif içeri sürükleme davranışını destekleyebilirsiniz.

İçeri sürüklemeyi etkinleştirme

Adobe AIR 1.0 ve üstü

İçeri sürükleme hareketini işlemek için, öncelikle varsayılan davranışı iptal etmeniz gerekir. Bırakma hedefleri olarak kullanmak istediğiniz herhangi bir HTML öğesindeki `dragenter` ve `dragover` olaylarını dinleyin. Bu olayların işleyicilerinde, gönderilen olay nesnesinin `preventDefault()` yöntemini çağırın. Varsayılan davranışın iptal edilmesi sayesinde, düzenlenemeyen bölgeler de bırakma işlemi alabilir.

Bırakılan verileri alma

Adobe AIR 1.0 ve üstü

`ondrop` olayı için işleyicideki bırakılan verilere erişebilirsiniz:

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

Panoya verileri okumak için, okunacak veri biçiminin MIME türünü ileterek `dataTransfer.getData()` yöntemini kullanın. Hangi veri biçimlerinin kullanılabilir olduğunu, `dataTransfer` nesnesinin `types` özelliğini kullanarak bulabilirsiniz. `types` dizisi, kullanılabilir her biçimin MIME türü dizesini içerir.

dragenter veya dragover olaylarında varsayılan davranışı iptal etiğinizde, bırakılan tüm verileri belgede uygun yere koymaktan siz sorumlu olursunuz. Bir fare konumunu bir öğenin içindeki ekleme noktasına dönüştürebilecek API mevcut değildir. Bu sınırlama, ekleme türü sürükleme hareketlerinin uygulanmasını zorlaştırabilir.

Örnek: Varsayılan HTML içeri sürükleme davranışını geçersiz kılma

Adobe AIR 1.0 ve üstü

Bu örnek, bırakılan öğede mevcut olan her veri biçimini gösteren bir tablo görüntüleyen bir bırakma hedefi uygular.

Varsayılan davranış metnin, bağların ve görüntülerin uygulama içerisinde sürüklenmesini sağlamak için kullanılır. Örnek, bırakma hedefi olarak işlev gören div öğesi için varsayılan içeri sürükleme davranışını geçersiz kılar. Düzenlenemeyen içeriğin içeri sürükleme hareketini kabul etmesini sağlamada anahtar adım, hem `dragenter` hem de `dragover` olayları için gönderilen olay nesnesinin `preventDefault()` yöntemini çağırmaktır. İşleyici, `drop` olayına yanıt olarak, aktarılan verileri HTML satır öğesine dönüştürür ve satırı görüntüleme için bir tabloya ekler.

```
 <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>
```

Uygulama dışı HTML sanal alanlarındaki dosya bırakmaları işleme

Adobe AIR 1.0 ve üstü

Uygulama dışı içerik, dosyalar bir AIR uygulamasına sürüklendiğinde ortaya çıkan File nesnelerine erişemez. Bu File nesnelerinden birini sanal alan köprüsü kullanarak uygulama içeriğine göndermek de mümkün değildir. (Nesne özelliklerine serileştirme sırasında erişilmelidir.) Ancak yine de HTMLLoader nesnesinde AIR nativeDragDrop olaylarını dinleyerek uygulamanızdaki dosyaları bırakabilirsiniz.

Normalde bir kullanıcı uygulama dışı içerik barındıran bir çerçeveye dosya bırakırsa, drop olayı alt öğeden üst öğeye yayılmaz. Ancak HTMLLoader (bir AIR uygulamasındaki tüm HTML içeriğinin kabı) tarafından gönderilen olaylar HTML olay akışının bir parçası olmadığı için, hala uygulama içeriğindeki drop olayını alabilirsiniz.

Bir dosya bırakma için olayı almak üzere, üst belge, `window.htmlLoader` tarafından sağlanan başvuruyu kullanarak HTMLLoader nesnesine bir olay dinleyicisi ekler:

```
window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

Aşağıdaki örnek, uzak sanal alana alt sayfa yükleyen bir üst belge kullanır (http://localhost/). Üst öğe HTMLLoader nesnesindeki `nativeDragDrop` olayını dinler ve dosya url'sini izler.

```
<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>
```

Alt belge, HTML `dragenter` ve `dragover` olay işleyicilerindeki Event nesnesi `preventDefault()` yöntemini engelleyerek geçerli bir bırakma hedefi sunmalıdır. Yoksa, drop olayı gerçekleşemez.

```
<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>
```

Dosya uzantılarını bırakma

Adobe AIR 2 ve üstü

Dosya vaadi, kullanıcının AIR uygulamasının dışında henüz var olmayan bir dosyayı sürüklemesini sağlayan bir sürükle ve bırak panosu biçimidir. Örneğin, uygulamanız dosya vaadini kullanarak kullanıcının bir proxy simgesini masaüstündeki bir klasöre sürüklemesine izin verebilir. Proxy simgesi, URL'de olduğu bilinen bir dosyayı veya bazı verileri temsil eder. Kullanıcı simgeyi bıraktıktan sonra çalışma zamanı, verileri indirir ve dosyayı, bırakma konumuna yazar.

URL'de erişilebilir olan dosyaları sürükleyip bırakmak için, bir AIR uygulamasındaki URLFilePromise sınıfını kullanabilirsiniz. URLFilePromise uygulaması, AIR 2 SDK'nin bir parçası olarak aircore kütüphanesinde bulunur. SDK frameworks/libs/air dizininde bulunan aircore.swc veya aircore.swf dosyasını kullanın.

Dönüşümlü olarak, kendi dosya izni mantığınızı IFilePromise arabirimini kullanarak (çalışma zamanı flash.desktop paketinde tanımlanır) uygulayabilirsiniz.

Dosya vaadi, kavramsal olarak panoda veri işleyici işlevini kullanarak gecikmeli görüntü oluşturma işlemine benzer. Dosyaları sürüklerken veya bırakırken gecikmeli görüntü oluşturma yerine dosya vaadini kullanın. Gecikmeli görüntü oluşturma tekniği, veri oluşturulurken veya indirilirken sürükleme hareketinde istenmeyen duraklamalara sebep olabilir. Gecikmeli görüntü oluşturmayı kopyala ve yapıştır işlemleri (dosya vaatlerinin desteklenmediği) için kullanın.

Dosya vaadi kullanımındaki sınırlamalar

Sürükle ve bırak panosuna yerleştirebileceğiniz diğer veri biçimleriyle karşılaştırıldığında dosya vaatleri aşağıdaki sınırlamalara sahiptir:

- Dosya vaatleri yalnızca AIR uygulamasından dışarı sürüklenebilir; AIR uygulamalarının içine bırakılamaz.
- Dosya vaatleri hiçbir işletim sistemi tarafından desteklenmez. Dosya vaatlerinin ana bilgisayar sisteminde desteklenip desteklenmediğini test etmek için `Clipboard.supportsFilePromise` özelliğini kullanın. Dosya vaadini desteklemeyen sistemlerde, dosya verilerini indirmek veya oluşturmak için alternatif bir mekanizma sağlamalısınız.
- Dosya vaatleri, kopyala ve yapıştır panosuyla (`Clipboard.generalClipboard` birlikte kullanılamaz.

Daha fazla Yardım konusu

flash.desktop.IFilePromise

air.desktop.URLFilePromise

Uzak dosyaları bırakma

Adobe AIR 2 ve üstü

URL'de mevcut dosya veya verileri temsil eden dosya izni nesneleri oluşturmak için URLFilePromise sınıfını kullanın. `FILE_PROMISE_LIST` pano biçimini kullanarak panoya bir veya daha fazla dosya izni nesnesi ekleyin. Aşağıdaki örnekte, http://www.example.com/foo.txt adresindeki bir dosya indiriliyor ve bırakma konumuna bar.txt olarak kaydediliyor. (Uzak ve yerel dosya adları eşleşmek zorunda değildir.)

```
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 );
}
```

Panoya atanmış diziye daha çok dosya izni ekleyerek kullanıcının aynı anda birden fazla dosya sürüklemesine izin verebilirsiniz. İşleme dahil edilen bazı veya tüm dosyaların bırakma konumuyla ilgili alt klasörlere yerleştirilmesi için `relativePath` özelliğinde alt dizinler de belirtebilirsiniz.

Aşağıdaki örnek, birden çok dosya izni içeren bir sürükleme işleminin nasıl başlatılacağını göstermektedir. Bu örnekte, bir html sayfası *article.html* iki bağlantılı görüntü dosyası ile birlikte panoya dosya izni olarak konuluyor. İlişkili bağlantıların korunması için görüntüler *images* alt klasörüne kopyalanır.

```
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 );
}
```

IFilePromise arabirimini uygulama

Adobe AIR 2 ve üstü

URLFilePromise nesnesi kullanılarak erişilemeyen kaynaklara dosya izni sağlamak için IFilePromise arabirimini özel bir sınıfta uygulayabilirsiniz. IFilePromise arabirimi, dosya izni bırakıldığında dosyaya yazılacak yazının AIR çalışma zamanı tarafından erişilebilmesi için kullanılan yöntem ve özellikleri tanımlar.

Bir IFilePromise uygulaması, dosya izni için veri sağlayan başka bir nesneyi AIR çalışma zamanına iletir. Bu nesne, AIR çalışma zamanının veri okumak için kullandığı IDataInput arabirimini uygulamalıdır. Örneğin, IFilePromise uygulayan URLFilePromise sınıfı, veri sağlayıcı olarak URLStream nesnesi kullanır.

AIR verileri senkronize olarak veya olmayarak okuyabilir. IFilePromise uygulaması `isAsync` özelliğine uygun değeri geri vererek hangi erişim modlarının desteklendiğini bildirir. Senkronize olmayan veri erişimi sağlanırsa, veri sağlayan nesne IEventDispatcher arabirimini uygulamalı ve `open`, `progress`, `complete` gibi gerekli olayları göndermelidir.

Dosya iznine yönelik veri sağlayıcı olarak özel bir sınıf veya aşağıdaki yerleşik sınıflardan birini kullanabilirsiniz:

- ByteArray (senkronize)
- FileStream (senkronize veya senkronize olmayan)
- Socket (senkronize)
- URLStream (senkronize olmayan)

IFilePromise arabirimini uygulamak için, aşağıdaki işlevlere ve özelliklere yönelik kod sağlamalısınız:

- `open():IDataInput(` — Vaat edilen dosyanın verilerinin okunacağı veri sağlayıcı nesneye döner. Nesne IDataInput arabirimini uygulamalıdır. Veriler senkronize olarak sağlanmıyorsa, nesne IEventDispatcher arabirimini de uygulamalıdır ve gerekli olayları göndermelidir (bkz. "Bir dosya izninde senkronize olmayan bir veri sağlayıcısı kullanma" sayfa 633).

- `get relativePath():String` — Oluşturulan dosya için dosya ismini de içeren yolu sağlar. Yol, kullanıcı tarafından sürükle ve bırak işleminde seçilen bırakma konumuyla ilişkili olarak çözümlenir. Yolun ana bilgisayar işletim sistemi için doğru ayırıcı karakteri kullandığından emin olmak için, dizin içeren yolları belirtirken`File.separator` sabitini kullanın. Yolun çalışma zamanına ayarlanmasına izin vermek için ayarlayıcı bir işlev ekleyebilir veya bir yapıcı parametresi kullanabilirsiniz.
- `get isAsync():Boolean` — Veri sağlayıcı nesnenin verileri senkronize olarak sağlayıp sağlamadığı konusunda AIR çalışma zamanına bilgi verir.
- `close():void` — Veriler tamamen okunduğunda (veya bir hata daha fazla okunmasını engellediğinde) çalışma zamanı tarafından çağırılır. Bu işlevi kaynakları temizlemek için kullanabilirsiniz.
- `reportError( e:ErrorEvent ):void` — Verileri okumada bir hata oluştuğunda çalışma zamanı tarafından çağırılır.

Tüm IFilePromise yöntemleri çalışma zamanı tarafından dosya iznini içeren sürükle ve bırak işlemleri sırasında çağırılır. Genel olarak, uygulama mantığınız bu yöntemlerden herhangi birini doğrudan çağırmamalıdır.

Bir dosya izninde senkronize olan veri sağlayıcısı kullanma

Adobe AIR 2 ve üstü

IFilePromise arabirimini kullanmanın en kolay yolu ByteArray veya senkronize FileStream gibi senkronize bir veri sağlayıcı nesne kullanmaktır. Aşağıdaki örnekte bir ByteArray nesnesi oluşturulur, veri ile doldurulur ve `open()` yöntemi çağırıldığında geri döndürülür.

```
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 );
        }
    }
}
```

Uygulamada senkronize dosya vaatlerinin sınırlı kullanım alanı vardır. Veri miktarı az ise, kolaylıkla geçici dizinde bir dosya oluşturup, sürükle ve bırak panosuna normal dosya liste dizisi ekleyebilirsiniz. Diğer yandan, veri miktarı çoksa veya veriyi oluşturmak hesaplama açısından verimli değilse, uzun senkronize olan bir işlem gereklidir. Uzun senkronize işlemler kullanıcı arabirimi güncellemelerini belirli bir süre için engelleyebilir ve uygulamanızın yanıt vermiyormuş gibi görünmesine neden olabilir. Bu sorundan kaçınmak için, zamanlayıcı tarafından yönlendirilen ve senkronize olmayan bir veri sağlayıcısı oluşturabilirsiniz.

Bir dosya izninde senkronize olmayan bir veri sağlayıcısı kullanma

Adobe AIR 2 ve üstü

Senkronize olmayan bir veri sağlayıcı nesnesi kullandığınızda IFilePromise`isAsync` özelliği `true` olmalı ve `open` yöntemiyle dönen nesne IEventDispatcher arabirimini uygulamalıdır. Yerleşik nesnelerin veri sağlayıcısında kullanılabilmesi için çalışma zamanı çeşitli alternatif olayları dinler. Örneğin, `progress` olayları FileStream ve URLStream nesneleri tarafından gönderilirken, `socketData` olayları Socket nesneleri tarafından gönderilirler. Çalışma zamanı tüm bu nesneler için uygun olayları dinler.

Aşağıdaki olaylar, veri sağlayıcı nesnesinden veri okuma işlemini yürütür:

- Event.OPEN — Çalışma zamanına veri kaynağının hazır olduğunu bildirir.
- ProgressEvent.PROGRESS — Çalışma zamanına verinin geçerli olduğunu bildirir. Çalışma zamanı veri sağlayıcı nesnesinden geçerli veri miktarını okur.
- ProgressEvent.SOCKET_DATA — Çalışma zamanına verinin geçerli olduğunu bildirir. `socketData` olayları soket temelli nesneler tarafından gönderilir. Diğer nesne türleri için bir `progress` olayı göndermelisiniz. (Çalışma zamanı, veri okunabilir olduğunda her iki olayın da bunu saptamasını dinler.)
- Event.COMPLETE — Çalışma zamanına verilerin okunduğunu bildirir.
- Event.CLOSE — Çalışma zamanına tüm verilerin okunduğunu bildirir. (Çalışma zamanı bu amaçla hem `close` ve `complete` öğesini dinler.)
- IOErrorEvent.IOERROR — Çalışma zamanına veri okuma sırasında bir hata oluştuğunu bildirir. Çalışma zamanı dosya oluşturma işlemini iptal eder ve IFilePromise `close()` yöntemini çağırır.

- SecurityErrorEvent.SECURITY_ERROR — Çalışma zamanına bir güvenlik hatası oluştuğunu bildirir. Çalışma zamanı dosya oluşturma işlemini iptal eder ve IFilePromise `close()` yöntemini çağırır.
- HTTPStatusEvent.HTTP_STATUS — Çalışma zamanı tarafından `httpResponseStatus` ile birlikte geçerli verinin hata mesajı yerine istenilen içeriği gösterdiğinden emin olmak için kullanılır (404 sayfası gibi). Bu olayı HTTP protokolüne bağlı nesneler göndermelidir.
- HTTPStatusEvent.HTTP_RESPONSE_STATUS — Çalışma zamanı tarafından `httpStatus` ile birlikte geçerli verilerin arzu edilen içeriği gösterdiğinden emin olmak için kullanılır. Bu olayı HTTP protokolüne bağlı nesneler göndermelidir.

Veri sağlayıcı bu olayları aşağıdaki sırada göndermelidir:

1. `open` olayı
2. `progress` veya `socketData` olayları
3. `complete` veya `close` olayı

***Not:** Hazır nesneler, FileStream, Socket ve URLStream uygun olayları otomatik olarak gönderir.*

Aşağıdaki örnek, özel, senkronize olmayan veri sağlayıcısı kullanarak dosya izni oluşturur. Veri sağlayıcı sınıfı ByteArray öğesini genişletir (IDataInput desteği için) ve IEventDispatcher arabirimini uygular. Her zamanlayıcı olayında, nesne bir yığın veri oluşturur ve çalışma zamanına verinin geçerli olduğunu haber vermek için ilerleme olayı gönderir. Yeterli veri oluşturulduğunda, nesne tamamlanmış bir olay gönderir.

```
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 );
    }
}
}
```

Not: *Çünkü örnekteki AsyncDataProvider sınıfı ByteArray öğesini genişlettiği için, EventDispatcher öğesini de genişletebilir. IEventDispatcher arabirimini uygulamak için, sınıf, dahili EventDispatcher nesnesini kullanır ve IEventDispatcher yöntemi çağrılarını dahili nesneye yönlendirir. EventDispatcher öğesini de genişletebilir ve IDataInput öğesini uygulayabilirsiniz (veya her iki arabirimi de uygulayabilirsiniz).*

Senkronize olmayan IFilePromise uygulaması senkronize olan uygulamayla aynıdır. Temel farklar `isAsync` öğesinin `true` değerini döndürmesi ve `open()` yönteminin senkronize olmayan bir veri nesnesini döndürmesidir:

```
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 );
        }
    }
}
```

Bölüm 36: Menülerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Web tabanlı Flex ve Flash Player uygulamalarında bağlam menüsünü değiştirmek için bağlam menüsü API'sindeki sınıfları kullanın.

Adobe® AIR®'de Uygulama, pencere, bağlam menülerini ve açılır menüleri tanımlamak için yerel menü API'lerindeki sınıfları kullanın.

Temel menü işlemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir AIR uygulamasında yerel menüler oluşturmayla ilgili hızlı bir açıklama ve kod örnekleri için Adobe Geliştirici Bağlantısında bulunan aşağıdaki hızlı başlangıç makalelerini inceleyin:

- Bir AIR uygulamasına yerel menüler ekleme (Flex)
- Bir AIR uygulamasına yerel menüler ekleme (Flash)

Yerel menü sınıfları, uygulamanızın çalıştığı işletim sisteminin yerel menü özelliklerine erişmenizi sağlar. NativeMenu nesneleri uygulama menüleri (Mac OS X'te bulunan), pencere menüleri (Windows ve Linux'ta bulunan), bağlam menüleri ve açılır menüler için kullanılabilir.

AIR'nin dışında, Flash Player'ın bir kullanıcı uygulamanızdaki bir nesneye sağ tıklattığında veya command düğmesine basarak tıklattığında otomatik olarak görüntülediği bağlam menüsünü kullanabilirsiniz. (Bir otomatik bağlam menüsü AIR uygulamaları için görüntülenmez.)

Menü sınıfları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Menü sınıfları şunları içerir:

Paket	Sınıflar
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

Menü çeşitleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

AIR, aşağıdaki menü türlerini destekler:

Bağlam menüleri Bağlam menüleri, SWF içeriğindeki etkileşimli bir nesnenin veya HTML içeriğindeki bir belge öğesinin sağ tıklatılması veya command tuşuna basılı tutularak tıklatılması yoluyla açılır.

Flash Player çalışma zamanında, bir bağlam menüsü otomatik olarak görüntülenir. ContextMenu ve ContextMenuItem sınıflarını menüye kendi komutlarınızı eklemek için kullanabilirsiniz. Ayrıca yerleşik komutların bazılarını, ancak hepsini değil kaldırabilirsiniz.

AIR çalışma zamanında, NativeMenu veya ContextMenu sınıflarından birini kullanarak bir bağlam menüsü oluşturabilirsiniz. AIR'de HTML içeriğinde, HTML öğelerine bağlam menüleri eklemek için Webkit HTML ve JavaScript API'lerini kullanabilirsiniz.

Uygulama menüleri (Yalnızca AIR) Uygulama menüsü, tüm uygulamada geçerli olan genel bir menüdür. Uygulama menüleri Mac OS X'te desteklenir, ancak Windows veya Linux'ta desteklenmez. Mac OS X'te, işletim sistemi otomatik olarak bir uygulama menüsü oluşturur. Standart menülere öğeler ve alt menüler eklemek için AIR menü API'lerini kullanabilirsiniz. Varolan menü komutlarını yönetmek için dinleyiciler ekleyebilirsiniz. Veya varolan öğeleri kaldırabilirsiniz.

Pencere menüleri (Yalnızca AIR) Pencere menüsü tek bir pencereyle ilişkilendirilmiştir ve başlık menüsünün altında görüntülenir. Menüler, bir NativeMenu nesnesi oluşturma ve bu nesneyi NativeWindow nesnesinin `menu` özelliğine atama yoluyla bir pencereye eklenebilir. Pencere menüleri Windows ve Linux işletim sistemlerinde desteklenir, ancak Mac OS X'te desteklenmez. Yerel pencere menüleri yalnızca sistem kromu olan pencerelerle kullanılabilir.

Dock ve sistem tepsisi simgesi menüleri (Yalnızca AIR) Bu simge menüleri bağlam menülerine benzer ve Mac OS X'te yer alan bir uygulama simgesine veya görev çubuğundaki Windows ve Linux bildirim alanlarına atanır. Dock ve sistem tepsisi simge menüleri, NativeMenu sınıfını kullanır. Mac OS X'te, menüdeki öğeler standart işletim sistemi öğelerinin üzerine eklenmiştir. Windows ve Linux'ta standart menü bulunmaz.

Açılır menüler (Yalnızca AIR) AIR açılır menüsü, bağlam menüsüne benzer, ancak belirli bir uygulama nesnesi veya bileşeni ile ilişkilendirilmiş olmayabilir. Açılır menüler, herhangi bir NativeMenu nesnesinin `display()` yönteminin çağrılması yoluyla pencerenin herhangi bir yerinde görüntülenebilir.

Özel menüler Yerel menüler tamamen işletim sistemiyle çizilir ve bu nedenle Flash ve HTML oluşturma modellerinin dışında yer alır. Yerel menüler kullanmak yerine, her zaman kendi özel, yerel olmayan menülerinizi MXML, ActionScript veya JavaScript (AIR'de) kullanarak oluşturabilirsiniz. Bu tip menüler uygulama içeriğinin içinde tamamen oluşturulmuş olmalıdır.

Flex menüleri Adobe® Flex™ çerçevesi, bir dizi Flex menü bileşeni sağlar. Flex menüleri işletim sistemiyle değil, çalışma zamanı ile çizilir ve *yerel* menüler değillerdir. Flex menü bileşeni, sistem kromu içermeyen Flex pencereleri için kullanılabilir. Flex menü bileşenini kullanmanın bir diğer faydası da, menüleri MXML biçiminde tanıtıcı biçimde belirtebiliyor olmanızdır. Flex Çerçevesi kullanıyorsanız, pencere menüleri için yerel sınıflar yerine Flex menü sınıflarını kullanın.

Varsayılan menüler (Yalnızca AIR)

Aşağıdaki varsayılan menüler işletim sistemi veya yerleşik bir AIR sınıfı tarafından sağlanır:

- Mac OS X'teki uygulama menüsü
- Mac OS X'teki dock simgesi menüsü
- HTML içeriğinde seçilen metin ve görüntüler için bağlam menüsü

- TextField nesnesinde (veya TextField'ı genişleten bir nesnede) seçilen metin için bağlam menüsü

Bağlam menüleri hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SWF içeriğinde, InteractiveObject nesnesinden miras alınan her nesnenin `contextMenu` özelliğine bir menü nesnesi atanarak, bu nesneye bir bağlam menüsü verilebilir. Varsayılan olarak İleri, Geri, Yazdır, Kalite, ve Yakınlaştır komutları dahil edilmiştir. AIR çalışma zamanında, `contextMenu` öğesine atanan menü nesnesi NativeMenu veya type ContextMenu tipinde olabilir. Flash Player çalışma zamanında, yalnızca ContextMenu sınıfı kullanılabilir.

ContextMenu ve ContextMenuItem sınıflarını kullanırken yerel menü olaylarını veya bağlam menüsü olaylarını dinleyebilirsiniz; her ikisi de gönderilir. ContextMenuEvent nesnesinin özelliklerinin sağladığı bir fayda contextMenuOwner öğesinin menünün ekleneceği nesneyi ve `mouseTarget` öğesinin menüyü açmak için tıklatılan nesneyi tanımlamasıdır. Bu bilgi NativeMenuEvent nesnesinde bulunmaz.

Aşağıdaki örnek bir hareketli görüntü oluşturur ve basit bir düzenleme bağlam menüsü ekler:

```
 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");}
```

Not: *Tarayıcı ortamında görüntülenen SWF içeriğinin aksine, AIR'deki bağlam menüleri yerleşik komutlar içermez.*

Bir Flash Player bağlam menüsünü özelleştirme

Bir tarayıcıda veya projektörde, SWF içeriğindeki bağlam menüleri her zaman yerleşik öğeler içerir. Menüden Ayarlar ve Hakkında komutları hariç tüm bu varsayılan komutları kaldırabilirsiniz. Bunun için Sahne alanı `showDefaultContextMenu` özelliğine `false` değerini vermeniz gerekir.

Bir görüntüleme nesnesine özel bir bağlam menüsü oluşturmak için yeni bir ContextMenu sınıfı örneği oluşturun, `hideBuiltInItems()` yöntemini çağırın ve bu örneği DisplayObject örneğinin `contextMenu` özelliğine atayın. Aşağıdaki örnek dinamik olarak çizilmiş olan ve rastgele bir renk verme komutu içeren bağlam menüsüne sahip bir kare sağlar:

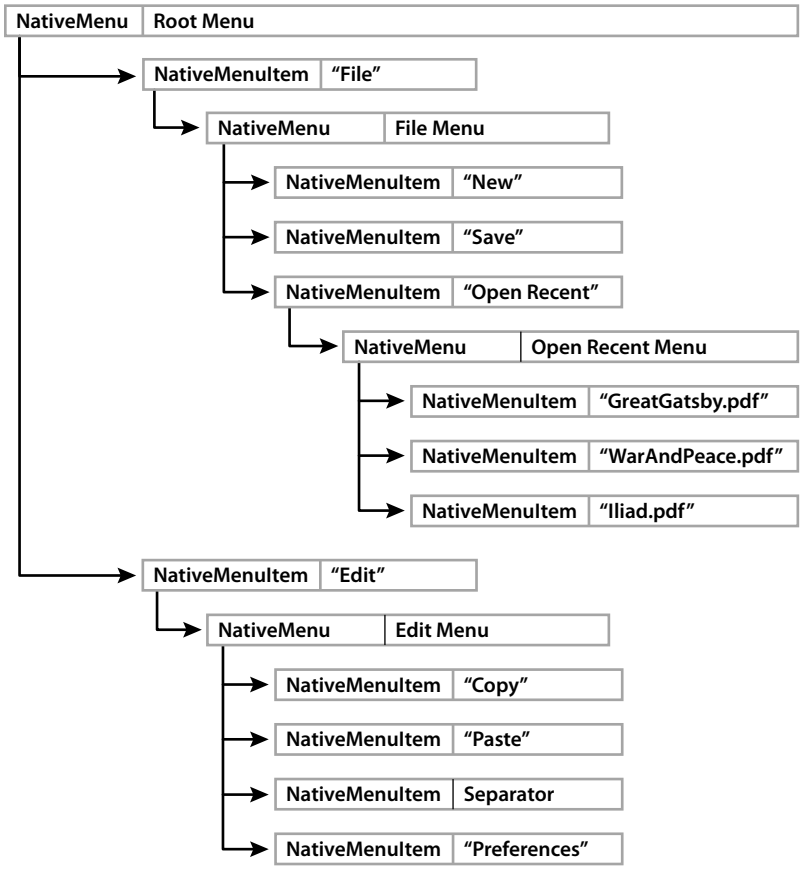
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

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);
}
```

Yerleşik menü yapısı (AIR)

Adobe AIR 1.0 ve üstü

Yerleşik menüler yapıları gereği hiyerarşiktir. NativeMenu nesneleri, alt NativeMenuItem nesneleri içerebilir. Alt menüleri temsil eden NativeMenuItem nesneleri, buna karşılık NativeMenu nesnelerini içerebilir. Yapıdaki üst veya kök düzey menü nesneleri, uygulama ve pencere menülerine ilişkin menü çubuğunu temsil eder. (Bağlam menüleri, simge menüleri ve açılır menüler menü çubuğu içermez.)

Aşağıdaki diyagram, tipik bir menünün yapısını gösterir. Kök menü, menü çubuğunu temsil eder ve *Dosya* alt menüsü ile *Düzen* alt menüsüne başvuruda bulunan iki menü öğesi içerir. Bu yapıdaki Dosya alt menüsü iki komut öğesinin yanı sıra *Son Kullanılan Menüyü Aç* alt menüsüne başvuruda bulunan ve kendisi üç öğe içeren bir öğeyi içerir. Düzen alt menüsü üç komut ve bir ayırıcı içerir.



Alt menünün tanımlanması, hem NativeMenu hem de NativeMenuItem nesnesini gerektirir. NativeMenuItem nesnesi, üst menüde görüntülenen etiketi tanımlar ve kullanıcının alt menüyü açmasını sağlar. NativeMenu nesnesi, alt menüdeki öğeler için kap işlevi görür. The NativeMenuItem nesnesi, NativeMenuItem `submenu` özelliği aracılığıyla NativeMenu nesnesine başvuruda bulunur.

Bu menüyü oluşturan bir kod örneği görüntülemek için bkz. "Yerel menü örneği: Pencere ve uygulama menüsü (AIR)" sayfa 650.

Menü olayları

Adobe AIR 1.0 ve üstü

NativeMenu ve NativeMenuItem nesneleri `preparing`, `displaying` ve `select` olaylarını gönderir:

Preparing: Nesne bir kullanıcı etkileşimine başlamak üzere olduğunda, menü ve menü öğeleri kayıtlı dinleyicilerine bir `preparing` olayı gönderir. Etkileşim bir menü açmayı veya klavye kısayoluyla bir öğe seçmeyi de içerir.

Not: *`preparing` olayı yalnızca Adobe AIR 2.6 ve sonrasında mevcuttur.*

Görüntüleme: Menü görüntülenmeden hemen önce, menü ve menü öğeleri tüm kayıtlı dinleyicilere bir `displaying` olayı gönderir.

`preparing` ve `displaying` olayları kullanıcıya gösterilmeden önce menü içeriğini veya öğe görüntüsünü güncellemeniz için fırsat verir. Örneğin, "Son Kullanılanları Aç" menüsünün `displaying` olayına ilişkin dinleyicide, menü öğelerini son görüntülenen belgelerin güncel listesini yansıtacak biçimde değiştirebilirsiniz.

Klavye kısayolu `preparing` olayını tetikleyen bir menü öğesini kaldırırsanız, menü etkileşimi başarıyla iptal edilir ve `select` olayı gönderilmez.

Olayın `target` ve `currentTarget` özellikleri dinleyicinin kayıtlı olduğu öğenin nesneleridir: menünün kendisi veya menü öğelerinden biri.

`preparing` olayı `displaying` olayından önce gönderilir. Genellikle ikisini birden değil bir olayı ya da diğerini dinlersiniz.

Seç: Kullanıcı bir komut öğesini seçtiğinde öğe kayıtlı dinleyicilere bir `select` olayı gönderir. Alt menü ve ayırıcı öğeleri seçilemez ve bu nedenle hiçbir zaman `select` olayı gönderemez.

`select` olayı, bir menü öğesinden bu öğeyi içeren menüye doğru ve kök menüye kadar baloncuk oluşturur. `Select` olaylarını doğrudan bir öğe üzerinde veya daha yüksek sesle menü yapısında dinleyebilirsiniz. Menüde `select` olayını dinlediğinizde, olayın `target` özelliğini kullanarak seçilen öğeyi tanımlayabilirsiniz. Olay menü hiyerarşisi boyunca kabarcık oluştururken olay nesnesinin `currentTarget` özelliği geçerli menü nesnesini tanımlar.

Not: *ContextMenu ve ContextMenuItem nesneleri `select`, `preparing` ve `displaying` olaylarının yanı sıra `menuItemSelect` ve `menuSelect` olaylarını gönderir.*

Yerleşik menü komutlarına ilişkin anahtar eşdeğerleri (AIR)

Adobe AIR 1.0 ve üstü

Menü komutuna bir tuş eşdeğeri (bazen hızlandırıcı olarak adlandırılır) atayabilirsiniz. Tuşa veya tuş bileşimine basıldığında, menü öğesi kayıtlı dinleyicilere bir `select` olayı gönderir. Komutun çağrılması için, öğeyi içeren menü uygulama menüsünün veya etkin pencerenin bölümü olmalıdır.

Tuş eşdeğerleri, ana anahtarı temsil eden bir dize ve ayrıca basılması gereken bir dizi değiştirici tuş olmak üzere iki bölümden oluşur. Ana anahtarı atamak için, menü öğesi `keyEquivalent` özelliğini bu tuş için tek karakter dizesi olarak ayarlayın. Büyük harf kullandığınızda, shift tuşu değiştirici dizisine otomatik olarak eklenir.

Mac OS X'te, varsayılan değiştirici (`Keyboard.COMMAND`) komut tuşudur. Windows ve Linux'ta, ctrl tuşudur (`Keyboard.CONTROL`). Bu varsayılan tuşlar değiştirici dizisine otomatik olarak eklenir. Farklı değiştirici tuşlar atamak için `keyEquivalentModifiers` özelliğine istenen tuş kodlarını içeren yeni bir dizi atayın. Varsayılan dizinin üzerine yazılır. Varsayılan değiştiricileri kullansanız da kendi değiştirici dizinizi atasanız da, `keyEquivalent` özelliğine atadığınız dize büyük harf olduğunda shift tuşu eklenir. Değiştirici tuşlar için kullanılacak tuş kodlarına ilişkin sabitler Keyboard sınıfında tanımlanır.

Atanan tuş eşdeğer dizesi, otomatik olarak menü öğesi adının yanında görüntülenir. Biçim kullanıcının işletim sistemine ve sistem tercihlerine bağlıdır.

Not: *Windows işletim sisteminde bir tuş değiştirici dizisine `Keyboard.COMMAND` değerini atadığınızda, menüde tuş eşdeğeri görüntülenmez. Ancak, menü komutunu etkinleştirmek için ctrl tuşu kullanılmalıdır.*

Aşağıdaki örnek, menü öğesine ilişkin tuş eşdeğeri olarak `Ctrl+Shift+G` tuşlarını atamaktadır:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

Bu örnek, doğrudan değiştirici dizisini ayarlayarak tuş eşdeğeri olarak `Ctrl+Shift+G` tuşlarını atar:

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

Not: *Tuş eşdeğerleri yalnızca uygulama ve pencere menüleri için tetiklenir. Bağlam menüsüne veya açılır menüye bir tuş eşdeğeri atadığınızda, tuş eşdeğeri menü etiketinde görüntülenir, ancak ilişkili menü komutu hiçbir zaman çağrılmaz.*

Anımsatıcılar (AIR)

Adobe AIR 1.0 ve üstü

Anımsatıcılar, menülere ilişkin işletim sistemi klavye arabiriminin bir parçasıdır. Linux, Mac OS X ve Windows kullanıcıların klavye yoluyla menüleri açmasına ve komutları seçmesine olanak verir, ancak arada küçük farklar vardır.

Mac OS X'te kullanıcı menünün ilk veya iki harfini yazar ve ardından return tuşuna basar. `mnemonicIndex` özelliği yok sayılır.

Windows'ta yalnızca tek bir harf önemlidir. Varsayılan olarak, önemli harf etiketteki ilk karakterdir. Ancak menü öğesine bir anımsatıcı atadığınızda, önemli karakter belirlenen harf olarak değişir. Bir menüdeki iki öğe aynı önemli karaktere sahip olduğunda (anımsatıcı atanmış olup olmaması önemli değildir) kullanıcının menü ile klavye etkileşimi biraz değişir. Kullanıcı, menüyü veya komutu seçmek üzere tek bir harfe basmak yerine, harfe istenen öğeyi vurgulamak için gerekli olduğu kadar basmalı ve daha sonra seçimi tamamlamak için Enter tuşunu kullanmalıdır. Tutarlı davranışı korumak için, pencere menülerine ilişkin bir menüdeki her öğeye benzersiz bir anımsatıcı atamanız tavsiye edilir.

Linux'ta varsayılan bir anımsatıcı sunulmaz. Bir anımsatıcı sağlamak için menü öğesinin `mnemonicIndex` özelliği için bir değer belirlemelisiniz.

Anımsatıcı karakteri etiket dizesinde dizin olarak belirleyin. Etiketteki ilk karakterin dizini 0'dır. Bu nedenle, etiketlenen menü öğesi “Format” için “r” anımsatıcısını kullanmak üzere `mnemonicIndex` özelliğini 2'ye eşitlemelisiniz.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

Menü öğesi durumu

Adobe AIR 1.0 ve üstü

Menü öğeleri iki durum özelliğine sahiptir, `checked` ve `enabled`:

checked Öğe etiketinin yanında bir onay işareti görüntülemek için `true` özelliğini belirleyin.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

enabled Komutun etkin olup olmadığını kontrol etmek için değeri `true` ve `false` arasında değiştirin. Etkin olmayan öğeler silik bir görselliğe sahiptir ve `select` olayları göndermez.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

Menü öğesine nesne ekleme

Adobe AIR 1.0 ve üstü

NativeMenuItem sınıfının `data` özelliği, her öğede rastgele bir nesneye başvuruda bulunabilmenizi sağlar. Örneğin, "Son Kullanılanları Aç" menüsünde her bir menü öğesine her belge için File nesnesi atayabilirsiniz.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

Yerel menüler oluşturma (AIR)

Adobe AIR 1.0 ve üstü

Bu başlık, AIR tarafından desteklenen çeşitli türlerdeki yerel menülerin nasıl oluşturulduğunu açıklar.

Root menu nesnesi oluşturma

Adobe AIR 1.0 ve üstü

Menü kökü olarak görev yapacak bir NativeMenu nesnesi oluşturmak için NativeMenu yapıcısını kullanın:

```
var root:NativeMenu = new NativeMenu();
```

Kök menü, uygulama ve pencere menüleri için menü çubuğunu temsil eder ve yalnızca alt menüleri açan öğeleri içermelidir. Bağlam menüsü ve açılır menüler menü çubuğu içermez. Bu nedenle kök menü, alt menülerin yanı sıra komutlar ve ayırıcı da içermelidir.

Menü oluşturulduktan sonra, menü öğelerini ekleyebilirsiniz. Öğeleri belirli bir dizine menu nesnesinin `addItemAt()` yöntemini kullanarak eklemediğiniz sürece, öğeler menüye eklendikleri sıraya göre görüntülenir.

Menüyü uygulama, pencere, simge veya bağlam menüsü olarak atayın veya aşağıdaki bölümlerde gösterildiği gibi açılır menü olarak görüntüleyin:

Uygulama menüsünü ayarlama veya pencere menüsü

Kodunuzun hem uygulama menülerine (Mac OS'de desteklenen) hem de pencere menülerine (diğer işletim sistemlerinde desteklenen) uyum sağlaması önemlidir.

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

Not: *Mac OS, her uygulama için standart öğeler içeren bir menü tanımlar. NativeApplication nesnesinin* `menu` *özelliğine yeni bir NativeMenu nesnesinin atanması, standart menüyü değiştirir. Değiştirmek yerine, standart menüyü de kullanabilirsiniz.*

Adobe Flex tüm platformlar arasında çalışan menüleri kolayca oluşturmak için bir FlexNativeMenu sınıfını sağlar. Flex Framework kullanıyorsanız, NativeMenu sınıfı yerine FlexNativeMenu sınıflarını kullanın.

Etkileşimli bir nesnede bağlam menüsünü ayarlama

```
interactiveObject.contextMenu = root;
```

Dock simgesi menüsünü ayarlama veya sistem tepsisi simgesi menüsü

Kodunuzun hem uygulama menülerine (Mac OS'de desteklenen) hem de pencere menülerine (diğer işletim sistemlerinde desteklenen) uyum sağlaması önemlidir.

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

Not: *Mac OS X, uygulama dock simgesi için standart bir menü tanımlar. DockIcon nesnesinin menu özelliğine yeni bir NativeMenu atadığınızda, bu menüdeki öğeler standart öğelerin üzerinde görüntülenir. Standart menü öğelerini kaldıramaz, değiştiremez veya bu öğelere erişemezsiniz.*

Menüyü açılır menü olarak görüntüleme

```
root.display(stage, x, y);
```

Daha fazla Yardım konusu

Çapraz platformda AIR uygulamaları geliştirme

Alt menü oluşturma

Adobe AIR 1.0 ve üstü

Alt menü oluşturmak için, üst menüye bir NativeMenuItem nesnesi ekleyin ve daha sonra öğenin `submenu` özelliğine alt menüyü tanımlayan NativeMenu nesnesini atayın. AIR, alt menü öğeleri ve bu öğelere ilişkin menü nesneleri oluşturmak için iki yol sunar:

`addSubmenu()` yöntemiyle bir menü öğesi ve buna ilişkin menü nesnesini tek bir adımda oluşturabilirsiniz:

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

Menü öğesini oluşturabilir ve menü nesnesini `submenu` özelliğine ayrıca da atayabilirsiniz:

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

Menü komutu oluşturma

Adobe AIR 1.0 ve üstü

Menü komutu oluşturmak için menüye bir NativeMenuItem nesnesi ve menü komutunu uygulayan işleve başvuran bir olay dinleyicisi ekleyin:

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

`Select` olayını komut öğesinin üzerinde dinleyebilir (örnekte gösterildiği gibi) veya `select` olayını üst menü nesnesinde dinleyebilirsiniz.

Not: *Alt menüleri ve ayırıcı satırları temsil eden menü öğeleri, `select` olayları göndermez ve bu nedenle komut olarak kullanılamaz.*

Menü ayırıcı satırı oluşturma

Adobe AIR 1.0 ve üstü

Ayırıcı satır oluşturmak için, yapıcıdaki `isSeparator` parametresini `true` olarak ayarlayarak bir NativeMenuItem oluşturun. Daha sonra ayırıcı öğeyi doğru yerdeki menüye ekleyin:

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

Ayırıcı için belirtilen bir etiket varsa, görüntülenmez.

HTML'deki (AIR) içerik menüleri hakkında

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi kullanılarak görüntülenen HTML içeriğinde, `contextmenu` olayı bir bağlam menüsü görüntülemek için kullanılabilir. Varsayılan olarak bağlam menüsü, kullanıcı seçilen metinde bağlam menüsü olayını çağırdığında görüntülenir (metni sağ tıklatma veya command tuşuna basılı tutarak tıklatma yoluyla). Varsayılan menünün açılmasını önlemek için, `contextmenu` olayını dinleyin ve event nesnesinin `preventDefault()` yöntemini çağırın:

```
function showContextMenu(event){
    event.preventDefault();
}
```

Özel bağlam menüsünü DHTML tekniklerini kullanarak veya bir AIR yerel bağlam menüsünü görüntüleyerek görüntüleyebilirsiniz. Aşağıdaki örnek, HTML `contextmenu` olayına yanıt olarak menü `display()` yöntemini çağırma yoluyla yerel bir bağlam menüsünü görüntüler:

```
<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>
```

Açılır pencere yerel menülerini görüntüleme (AIR)

Adobe AIR 1.0 ve üstü

Menü `display()` yöntemini çağırarak herhangi bir NativeMenu nesnesini bir pencerenin üzerinde rastgele bir zamanda ve yerde görüntüleyebilirsiniz. Yöntem, sahneye başvuru gerektirir; bu nedenle yalnızca uygulama sanal alanındaki içerik menüyü açılır biçimde görüntüleyebilir.

Aşağıdaki yöntem, fare tıklatmasına karşılık olarak `popupMenu` adlı bir NativeMenu nesnesi tarafından tanımlanan menüyü görüntüler:

```
private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

Not: *Menünün doğrudan bir olaya karşılık olarak görüntülenmesi gerekmez. Herhangi bir yöntem `display()` işlevini çağırabilir.*

Menü olaylarını işleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı menüyü veya bir menü öğesini seçtiğinde, menü olayları gönderir.

Menü sınıfları için olay özeti

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Menü olaylarını işlemek için menülere veya ayrı ayrı öğelere olay dinleyicileri ekleyin.

Nesne	Gönderilen olaylar
NativeMenu (AIR)	Event.PREPARING (Adobe AIR 2.6 ve üstü) Event.DISPLAYING Event.SELECT (alt öğelerden ve alt menülerden yayılan)
NativeMenuItem (AIR)	Event.PREPARING (Adobe AIR 2.6 ve üstü) Event.SELECT Event.DISPLAYING (üst menüden yayılan)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT (AIR)

Menü olaylarını seç

Adobe AIR 1.0 ve üstü

Bir menü öğesini tıklatmayı işlemek için, NativeMenuItem nesnesine `select` olayına ilişkin bir olay dinleyicisi ekleyin:

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

`Select` olayı mevcut menülere baloncuk oluşturduğundan, bir üst menüdeki select olaylarını da dinleyebilirsiniz. Menü düzeyinde dinleme sırasında, hangi menü komutunun seçildiğini belirlemek için olay nesnesinin `target` özelliğini kullanabilirsiniz. Aşağıdaki örnek, seçilen komutun etiketini gösterir:

```
 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");
}
```

ContextMenuItem sınıfını kullanıyorsanız, `select` olayını veya `menuItemSelect` olayını dinleyebilirsiniz. `menuItemSelect` olayı size bağlam menüsünü içeren nesne hakkında ek bilgi verir, ancak mevcut menülere kabarcık oluşturmaz.

Menü olaylarını görüntüleme

Adobe AIR 1.0 ve üstü

Menü açılışını işlemek için, menü görüntülenmeden önce gönderilen `displaying` olayı için dinleyici ekleyebilirsiniz. Menüyü güncellemek için, örneğin öğeler ekleyerek veya kaldırarak ya da ayrı ayrı öğelerin etkin veya işaretli durumlarını güncelleyerek görüntüleme olayını kullanabilirsiniz. Ayrıca `menuSelect` olayını bir ContextMenu nesnesinden dinleyebilirsiniz.

AIR 2.6 ve üstünde bir menü görüntüleme veya klavye kısayoluyla öğe seçmeye karşılık olarak bir menüyü güncellemek için `preparing` olayını kullanabilirsiniz.

Yerel menü örneği: Pencere ve uygulama menüsü (AIR)

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, “Yerleşik menü yapısı (AIR)” sayfa 641 içinde gösterilen menüyü oluşturur.

Menü, hem yalnızca pencere menülerini destekleyen Windows'ta, hem de yalnızca uygulama menülerini destekleyen Mac OS X'te çalışmak üzere tasarlanmıştır. MenuExample sınıfı yapıcısı, ayrım yapmak için NativeWindow ve NativeApplication sınıflarının statik `supportsMenu` özelliğini kontrol eder. Yapıcı, `NativeWindow.supportsMenu` öğesi `true` olduğunda pencere için bir NativeMenu nesnesi oluşturur, daha sonra Dosya ile Düzen alt menülerini oluşturur ve ekler. `NativeApplication.supportsMenu` öğesi `true`, yapıcı Dosya ve Düzen menülerini oluşturarak bunları Mac OS X işletim sisteminin sağladığı varolan menüye ekler.

Örnek ayrıca, menü olay işlemesini de gösterir. `Select` olayı öğe düzeyinin yanı sıra menü düzeyinde de işlenir. Seçilen öğeyi içeren menüden kök menüye kadarki zincirde bulunan her bir menü `select` olayını yanıtlar. `displaying` olayı "Son Kullanılanları Aç" menüsüyle birlikte kullanılır. Menü açılmadan hemen önce, menüdeki öğeler en son kullanılan Belgeler dizisinden (bu örnekte aslında değişmeyen) yenilenir. Bu örnekte gösterilmese de, `displaying` olaylarını ayrı ayrı öğelerde de dinleyebilirsiniz.

```
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;
        }
    }
}
```

Bölüm 37: AIR'de görev çubuğu simgeleri

Adobe AIR 1.0 ve üstü

Çoğu işletim sistemi, Mac OS X dock'u gibi uygulamayı temsil edecek bir simge içeren bir görev çubuğu sağlar. Adobe® AIR® `NativeApplication.nativeApplication.icon` özelliği aracılığıyla uygulama görev çubuğu simgesiyle etkileşik kurmak için bir arabirim sağlar.

- Sistem tepsisini ve dock öğelerini kullanma (Flex)
- Sistem tepsisi ve dock simgelerini kullanma) (Flash)

Daha fazla Yardım konusu

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

Görev çubuğu simgeleri hakkında

Adobe AIR 1.0 ve üstü

AIR, `NativeApplication.nativeApplication.icon` nesnesini otomatik olarak oluşturur. Nesne türü işletim sistemine bağlı olarak DockIcon veya SystemTrayIcon olur. `NativeApplication.supportsDockIcon` ve `NativeApplication.supportsSystemTrayIcon` özelliklerini kullanarak, AIR'in geçerli işletim sisteminde bu InteractiveIcon alt sınıflarından hangisini desteklediğini belirleyebilirsiniz. InteractiveIcon temel sınıfı, simge için kullanılan görüntüyü değiştirmek için kullanabileceğiniz `width`, `height` ve `bitmaps` özelliklerini sağlar. Ancak, DockIcon veya SystemTrayIcon öğelerine özgü özelliklere yanlış işletim sisteminde erişilmesi, bir çalışma zamanı hatası oluşturur.

Bir simge için kullanılan görüntüyü ayarlamak veya değiştirmek için, bir veya daha çok görüntü içeren bir dizi oluşturun ve bu diziyi `NativeApplication.nativeApplication.icon.bitmaps` özelliğine atayın. Görev çubuğu simgelerinin boyutu farklı işletim sistemlerinde farklı olabilir. Ölçeklemeye bağlı görüntü bozulmasını önlemek için `bitmaps` dizisine görüntülerin birden çok boyutunu ekleyebilirsiniz. Birden çok görüntü sağladığınızda, AIR gerektiğinde görüntüyü ölçekleyerek görev çubuğu simgesinin geçerli görüntü boyutuna en yakın boyutu seçer. Aşağıdaki örnek, iki görüntüyü kullanarak görev çubuğu simgesine ilişkin görüntüyü ayarlar:

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

Simge görüntüsünü değiştirmek için `bitmaps` özelliğine yeni görüntüyü veya görüntüleri içeren bir dizi atayın. Simgeyi, bir `enterFrame` or `timer` olayına yanıt olarak görüntüyü değiştirerek hareketlendirebilirsiniz.

Simgeyi Windows ve Linux'ta bildirim alanından kaldırmak veya Mac OS X'te varsayılan simge görünümünü geri yüklemek için `bitmaps` öğesini boş bir dizi olarak ayarlayın:

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

Dock simgeleri

Adobe AIR 1.0 ve üstü

AIR, `NativeApplication.supportsDockIcontrue` olduğunda dock simgelerini destekler. `NativeApplication.nativeApplication.icon` özelliği, dock üzerindeki uygulama simgesini temsil eder. (Pencere dock simgesini değil.)

Not: *AIR, Mac OS X altındaki dock üzerinde pencere simgelerini değiştirmeyi desteklemez. Ayrıca, uygulama dock simgesindeki değişiklikler yalnızca uygulama çalışırken uygulanır — uygulama iptal edildiğinde simge normal görünümüne geri döner.*

Dock simgesi menüleri

Adobe AIR 1.0 ve üstü

Komutlar içeren bir NativeMenu nesnesi oluşturarak ve bu nesneyi `NativeApplication.nativeApplication.icon.menu` özelliğine atayarak standart dock menüsüne komutlar ekleyebilirsiniz. Menüdeki öğeler, standart dock simgesi menü öğelerinin üzerinde görüntülenir.

Dock'u zıplatma

Adobe AIR 1.0 ve üstü

`NativeApplication.nativeApplication.icon.bounce()` yöntemini çağırarak dock simgesini zıplatabilirsiniz. `bounce() priority` parametresini bilgi amaçlı olarak ayarlarsanız, simge bir kez zıplar. Parametreyi kritik olarak belirlerseniz, simge kullanıcı uygulamayı etkinleştirene kadar zıplar. `Priority` parametresine ilişkin sabitler NotificationType sınıfında tanımlanır.

Not: *Uygulama zaten etkinse, simge zıplamaz.*

Dock simgesi olayları

Adobe AIR 1.0 ve üstü

Dock simgesi tıklatıldığında, NativeApplication nesnesi bir `invoke` olayı gönderir. Uygulama çalışmıyorsa, sistem uygulamayı başlatır. Aksi halde, `invoke` olayı çalışan uygulama örneğine iletilir.

Sistem Tepsisi simgeleri

Adobe AIR 1.0 ve üstü

Şu anda Windows ve çoğu Linux dağıtımlarında olduğu gibi `NativeApplication.supportsSystemTrayIcontrue` değerinde olduğunda, AIR sistem tepsisi simgelerini destekler. Windows ve Linux'ta, sistem tepsisi simgeleri görev çubuğunun bildirim alanında görüntülenir. Simge, varsayılan olarak görüntülenir. Simgeyi görüntülemek için, simgenin `bitmaps` özelliğine BitmapData nesneleri içeren bir dizi atayın. Simge görüntüsünü değiştirmek için `bitmaps` öğesine yeni görüntüleri içeren bir dizi atayın. Simgeyi kaldırmak için `bitmaps` öğesini `null` değerine ayarlayın.

Sistem tepsisi simge menüleri

Adobe AIR 1.0 ve üstü

Bir NativeMenu nesnesi oluşturup bu nesneyi `NativeApplication.nativeApplication.icon.menu` özelliğine atayarak menüye sistem tepsisi simgesi ekleyebilirsiniz. (İşletim sistemi, varsayılan menü sağlamaz.) Sistem tepsisi simge menülerine simgeyi sağ tıklatarak erişin.

Sistem tepsisi simge araç ipuçları

Adobe AIR 1.0 ve üstü

Araç ipucu özelliğini ayarlayarak simgeye araç ipucu ekleyebilirsiniz:

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

Sistem tepsisi simge olayları

Adobe AIR 1.0 ve üstü

NativeApplication.nativeApplication.icon özelliği tarafından başvurulan SystemTrayIcon nesnesi, `click`, `mouseDown`, `mouseUp`, `rightClick`, `rightMouseDown` ve `rightMouseUp` olayları için bir ScreenMouseEvent gönderir. Bu olayları simge menüsüyle birlikte, görünür pencere olmadığında kullanıcıların uygulamanızla etkileşim kurmasına izin vermek için kullanabilirsiniz.

Örnek: Pencere içermeyen bir uygulama oluşturma

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, bir sistem tepsisi simgesi içeren, ancak görünür pencere içermeyen bir AIR uygulaması oluşturur. (Uygulama tanımlayıcısında uygulamanın `görünür` özelliği `true` değerine ayarlanmamalıdır, yoksa uygulama başlatıldığında pencere görünür olacaktır.)

```
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];
    }
  }
}
```

Not: *Flex WindowedApplication bileşenini kullanırken, WindowedApplication etiketinin* `visible` *niteliğini* `false` *değerine ayarlamanız gerekir. Bu nitelik uygulama tanımlayıcısındaki ayarın üzerine geçer.*

Not: *Örnek, uygulamanın* `icons` *alt dizininde* `AIRApp_16.png` *ve* `AIRApp_128.png` *adlı görüntü dosyaları olduğunu varsayar. (Proje klasörünüze kopyalayabileceğiniz örnek simge dosyaları AIR SDK'da mevcuttur.)*

Pencere görev çubuğu simge ve düğmeleri

Adobe AIR 1.0 ve üstü

Pencerelerin simgelenmiş temsilleri, kullanıcıların arka plandaki veya simge durumuna küçültülmüş pencerelere kolayca erişebilmeleri için, genellikle dock veya görev çubuğunun pencere alanında görüntülenir. Mac OS X dock'u, simge durumuna küçültülen her pencerenin yanı sıra, uygulamanız için de bir simge görüntüler. Microsoft Windows ve Linux görev çubukları, uygulamanızda yer alan normal türdeki her pencere için program simgesi ve başlığı içeren bir düğme görüntüler.

Görev çubuğu pencere düğmesini vurgulama

Adobe AIR 1.0 ve üstü

Bir pencere arka planda olduğunda, pencereyle ilişkin ilgilendiği bir olayın gerçekleştiğini kullanıcıya bildirebilirsiniz. Mac OS X'te, kullanıcıyı uygulama dock simgesini zıplatarak bilgilendirebilirsiniz ("Dock'u zıplatma" sayfa 655 bölümünde açıklandığı gibi). Windows ve Linux'ta, NativeWindow örneğinin `notifyUser()` yöntemini çağırarak pencere görev çubuğu düğmesini vurgulayabilirsiniz. Yönteme iletilen `type` parametresi, bildirimin aciliyetini belirler.

- `NotificationType.CRITICAL`: kullanıcı pencereyi ön plana getirene kadar, pencere simgesi yanıp söner.
- `NotificationType.INFORMATIONAL`: pencere simgesi renk değiştirerek vurgulanır.

  Not: *Linux'ta yalnızca bilgilendirici bildirim türü desteklenir.* `notifyUser()` *işlevine herhangi bir türdeki değerin iletilmesi aynı etkiyi yaratacaktır.*

  Aşağıdaki ifade, bir pencerenin görev çubuğu düğmesini vurgular:

  ```
  stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
  ```

  İşletim sisteminde pencere düzeyinde bildirimi desteklemeyen`NativeWindow.notifyUser()` yönteminin çağrılması etkisizdir. Pencere bildiriminin desteklenip desteklenmediğini belirlemek için `NativeWindow.supportsNotification` özelliğini kullanın.

Görev çubuğu düğmeleri veya simgeleri içermeyen pencereler oluşturma

Adobe AIR 1.0 ve üstü

Windows işletim sisteminde, *utility* veya *lightweight* türüyle oluşturulan pencereler görev çubuğunda görünmez. Görünür olmayan pencereler görev çubuğunda da görüntülenmez.

İlk pencerenin *normal* türünde olması gerektiğinden, görev çubuğunda görünen herhangi bir pencere içermeyen bir uygulama oluşturmak için ilk pencereyi kapatmalı veya görünmez olarak bırakmalısınız. Uygulamayı iptal etmeden uygulamanızdaki tüm pencereleri kpatmak için, son pencereyi kapatmadan önce NativeApplication nesnesinin `autoExit` özelliğini `false` olarak ayarlayın. İlk pencerenin görünür olmasını engellemek için, uygulama tanımlayıcı dosyasının `<initalWindow>` öğesine `<visible>false</visible>` özelliğini ekleyin (ve `visible` özelliğini `true` olarak ayarlamayın veya pencerenin `activate()` yöntemini çağırmayın).

Uygulama tarafından açılan yeni pencerelerde, pencere yapıcısına iletilen NativeWindowInitOption nesnesinin `type` özelliğini `NativeWindowType.UTILITY` veya `NativeWindowType.LIGHTWEIGHT` olarak ayarlayın.

Mac OS X'te, simge durumuna küçültülen pencereler dock görev çubuğunda görüntülenir. Pencereyi simge durumuna küçültmek yerine gizlediğinizde, küçültülen simgenin görüntülenmesini engelleyebilirsiniz. Aşağıdaki örnek bir `nativeWindowDisplayState` değişiklik olayını dinler ve pencere simge durumuna küçültüldüğünde bu olayı iptal eder. Bunun yerine işleyici, pencerenin `visible` özelliğini `false` olarak ayarlar:

```
private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

Mac OS X dock'unda bir pencere simge durumuna küçültülmüşse, `visible` özelliğini `false` olarak ayarladığınızda dock simgesi kaldırılmaz. Kullanıcı yine de, pencerenin yeniden görünmesi için simgeyi tıklatabilir.

Bölüm 38: Dosya sistemiyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash® Player FileReference sınıfı yoluyla temel dosya okuma ve yazma olanaklarını sağlar.f Güvenlik sebeplerinden dolayı, Flash Player'da çalışırken kullanıcı her zaman siz bir dosyayı okuyabilir veya yazabilir hale gelmeden önce izin vermelidir.

Adobe® AIR® ana bilgisayarın dosya sistemine Flash Player'da olabileceğinden daha eksiksiz erişim sağlar. AIR dosya sistemi API'sini kullanarak dizinlere ve dosyalara erişebilir ve bunları yönetebilir, oluşturabilir, dosyalara veri yazabilir, vb. işlemler gerçekleştirebilirsiniz.

Daha fazla Yardım konusu

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

FileReference sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

FileReference nesnesi, istemci veya sunucu makinesinde bir veri dosyasını temsil eder. FileReference sınıfının yöntemleri, uygulamanızın yerel olarak veri dosyalarını yükleyip kaydetmesine ve uzak sunuculara ve uzak sunuculardan dosya verilerini aktarmasına olanak sağlar.

FileReference sınıfı, veri dosyalarını yüklemeye, aktarmaya ve kaydetmeye yönelik iki farklı yaklaşım sunar. Kullanıma sunulduğu günden bu yana, FileReference sınıfına `browse()` yöntemi, `upload()` yöntemi ve `download()` yöntemi eklenmiştir. Kullanıcının dosya seçmesini sağlamak için `browse()` yöntemini kullanın. Dosya verilerini uzaktaki bir sunucuya aktarmak için `upload()` yöntemini kullanın. Sunucudan aynı verilere ulaşmak ve yerel bir dosyaya kaydetmek için `download()` yöntemimi kullanın. Flash Player 10 ve Adobe AIR 1.5 ile başlandığında, FileReference sınıfına`load()` ve `save()` yöntemleri de eklenir. `load()` ve `save()` yöntemleri, yerel dosyalara doğrudan erişmenizi ve yerel dosyaları kaydetmenizi sağlar. Bu yöntemlerin kullanımı, URLLoader ve Loader sınıflarındaki eşdeğer adlara sahip yöntemlere benzerdir.

Not: *FileReference sınıfını ve FileStream sınıfını genişleten File sınıfı dosyalarla ve yerel dosya sistemiyle çalışmaya yarayan ek işlevler sağlar. File ve FileStream sınıfları yalnızca AIR tarafından desteklenir ve Flash Player tarafından desteklenmez.*

FileReference sınıfı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Her FileReference nesnesi, yerel makinede tek bir veri dosyasını temsil eder. FileReference sınıfının özellikleri, dosyanın boyutu, türü, adı, dosya adı uzantısı, oluşturucusu, oluşturma tarihi ve değiştirme tarihi hakkında bilgileri içerir.

Not: *`creator` özelliği yalnızca Mac OS'de desteklenmektedir. Diğer tüm platformlar `null` değerini döndürür.*

Not: *`extension` özelliği yalnızca Adobe AIR uygulamasında desteklenmiştir.*

FileReference sınıfının bir örneğini iki şekilde oluşturabilirsiniz:

- Şu kodda gösterildiği gibi, `new` operatörünü kullanın:

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- Kullanıcının karşıya yükleme işlemi için bir veya daha fazla dosya seçmesini sağlayan bir iletişim kutusunu açan `FileReferenceList.browse()` yöntemini çağırın. Kullanıcı bir veya daha fazla dosyayı başarıyla seçerse, bu, FileReference nesnelerinin bir dizisini oluşturur.

Bir FileReference nesnesi oluşturduktan sonra aşağıdakileri yapabilirsiniz:

- Kullanıcının yerel dosya sisteminden tek bir dosya seçmesini sağlayan bir iletişim kutusunu açan `FileReference.browse()` yöntemini çağırabilirsiniz. Bu genellikle, `FileReference.upload()` veya `FileReference.load()` yönteminin daha sonra çağırılmasından önce yapılır. Dosyayı uzaktaki bir sunucuya yüklemek için`FileReference.upload()` yöntemini çağırın. Yerel bir dosyayı açmak için `FileReference.load()` yöntemini çağırın.
- `FileReference.download()` yöntemini çağırın. `download()` yöntemi kullanıcının yeni dosyayı kaydetmesi için bir konum seçmesini sağlayan bir iletişim kutusu açar. Daha sonra bu sunucudan verileri indirir ve yeni dosyaya kaydeder.
- `FileReference.load()` yöntemini çağırın. Bu yöntem, `browse()` yöntemini kullanarak önceden seçilmiş olan bir dosyadan veri yüklemeye başlar. `browse()` işlemi tamamlanıncaya kadar (kullanıcı bir dosya seçer) `load()` yöntemi çağrılamaz.
- `FileReference.save()` yöntemini çağırın. Bu yöntem bir iletişim kutusu açar ve kullanıcıdan yerel dosya sisteminde tek bir dosya konumu seçmesini ister. Daha sonra verileri belirtilen konuma kaydeder.

Not: *Herhangi bir noktada yalnızca bir iletişim kutusu açılabildiğinden, aynı anda yalnızca bir `browse()`, `download()` veya `save()` eylemi gerçekleştirebilirsiniz.*

FileReference nesnesinin, `name`, `size` veya `modificationDate` gibi özellikleri aşağıdakilerden biri gerçekleşmedikçe tanımlanmaz:

- `FileReference.browse()` yöntemi veya `FileReferenceList.browse()` yöntemi çağrılıp kullanıcı, iletişim kutusunu kullanarak bir dosya seçinceye kadar.
- `FileReference.download()` yöntemi çağrılıp kullanıcı iletişim kutusunu kullanarak yeni bir dosya konumu belirtinceye kadar.

Not: *İndirme işlemi gerçekleştirilirken, indirme tamamlanmadan önce yalnızca `FileReference.name` özelliği doldurulur. Dosya indirildikten sonra, tüm özellikler kullanılabilir.*

`FileReference.browse()`, `FileReferenceList.browse()`, `FileReference.download()`, `FileReference.load()` veya `FileReference.save()` yöntemlerine yapılan çağrılar çalıştırılıyorken, birçok oynatıcı olayları gönderme ve kodu çalıştırma işlemleri de dahil olmak üzere SWF dosyasını oynatmaya devam eder.

Yükleme ve indirme işlemleri için SWF dosyası, ilke dosyası tarafından belirtilen etki alanları da dahil olmak üzere yalnızca kendi etki alanındaki dosyalara erişebilir. Sunucu, yüklemeyi başlatan SWF dosyasıyla aynı etki alanında değilse, sunucuya dosyayı içeren bir ilke dosyası koymanız gerekir.

Bkz. FileReference.

Dosyalardan veri yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`FileReference.load()` yöntemi yerel bir dosyadan belleğe veri yüklemenizi sağlar.

Not: *Öncelikle kodunuzun kullanıcının yüklenecek bir dosya seçmesini sağlamak için `FileReference.browse()` yöntemini çağırması gerekir. Bu kısıtlama uygulama güvenlik sanal alanında Adobe AIR'de çalışan içeriğe etki etmez.*

`FileReference.load()` yöntemi çağrıldıktan hemen sonra döndürülür, ancak yüklenen veriler kullanılabilir duruma hemen gelmez. FileReference nesnesi yükleme işleminin her adımında dinleyici yöntemlerini çağırmak için olaylar gönderir.

FileReference nesnesi, yükleme işlemi sırasında şu olayları gönderebilir.

- `open` olayı (`Event.OPEN`): Yükleme işlemi başladığında gönderilir.
- `progress` olayı (`ProgressEvent.PROGRESS`): Dosyadan veri baytları okundukça düzenli aralıklarla gönderilir.
- `complete` olayı (`Event.COMPLETE`): Yükleme işlemi başarıyla tamamlandığında gönderilir.
- `ioError` olayı (`IOErrorEvent.IO_ERROR`): Dosyadan veri açılırken veya okunurken bir girdi/çıktı hatası oluştuğundan yükleme işlemi başarısız olduğunda gönderilir.

FileReference nesnesi tamamlanmış olayı gönderdikten sonra, yüklenen verilere FileReference nesnesinin `data` özelliğinden ByteArray olarak erişilebilir.

Şu örnek, kullanıcıdan bir dosya seçmesinin ve sonra o dosyadaki verileri belleğe yüklemesinin nasıl istendiğini göstermektedir:

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

Örnek kod öncelikle `fileRef` adlı FileReference nesnesini oluşturur ve ardından `browse()` yöntemini çağırır. `browse()` yöntemi kullanıcıdan bir dosya seçmesini isteyen bir iletişim kutusu açar. Bir dosya seçildiğinde kod, `onFileSelected()` yöntemini görevlendirir. Bu yöntem `progress` ve `complete` olayları için dinleyiciler ekler ve daha sonra FileReference nesnesinin `load()` yöntemini çağırır. Örnekteki diğer işleyici yöntemleri, yükleme işleminin ilerlemesiyle ilgili bildiri göndermek için mesajları verir. Yükleme işlemi tamamlandığında uygulama, `trace()` yöntemini kullanarak yüklenen dosyanın içindekileri görüntüler.

Adobe AIR'de, FileStream sınıfı bir yerel dosyadan veri okumaya yarayan ek işlev sağlar. Bkz. "Dosyaları okuma ve dosyalara yazma" sayfa 696

Yerel dosyalara veri kaydetme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`FileReference.save()` yöntemi yerel bir dosyaya veri kaydetmenizi sağlar. Kullanıcının, yeni bir dosya adı ve dosyanın kaydedileceği bir konum girmesini sağlayan bir iletişim kutusunu açarak başlar. Kullanıcı, dosya adını ve konumunu seçtikten sonra veriler yeni dosyaya yazılır. Dosya başarıyla kaydedildiğinde, FileReference nesnesinin özellikleri, yerel dosyanın özellikleriyle doldurulur.

Not: *Kodunuzun `FileReference.save()` yöntemini yalnızca fare tıklatması veya tuş basışı olayı gibi bir kullanıcı olayına yanıt olarak çağırması gerekir. Aksi takdirde bir hata atılır. Bu kısıtlama uygulama güvenlik sanal alanında Adobe AIR'de çalışan içeriğe etki etmez.*

`FileReference.save()` yöntemi, çağrıldıktan hemen sonra döndürülür. FileReference nesnesi daha sonra dosya kaydetme işleminin her adımında dinleyici yöntemlerini çağırmak için olaylar gönderir.

FileReference nesnesi, dosya kaydetme işlemi sırasında şu olayları gönderir:

- `select` olayı (`Event.SELECT`): Kullanıcı, kaydedilecek yeni dosyanın konumunu ve dosya adını belirttiğinde gönderilir.
- `cancel` olayı (`Event.CANCEL`): Kullanıcı, iletişim kutusunda İptal düğmesini tıklattığında gönderilir.
- `open` olayı (`Event.OPEN`): Kaydetme işlemi başladığında gönderilir.
- `progress` olayı (`ProgressEvent.PROGRESS`): Dosyaya veri baytları kaydedildikçe düzenli aralıklarla gönderilir.
- `complete` olayı (`Event.COMPLETE`): Kaydetme işlemi başarıyla tamamlandığında gönderilir.
- `ioError` olayı (`IOErrorEvent.IO_ERROR`): Dosyaya veri kaydetmeye çalışılırken bir girdi/çıktı hatası oluştuğundan kaydetme işlemi başarısız olduğunda gönderilir.

`FileReference.save()` yönteminin `data` parametresinde iletilen nesne türü, verilerin dosyaya nasıl yazılacağını belirler:

- Bu bir String değeriyse, UTF-8 kodlamasının kullanıldığı bir metin dosyası olarak kaydedilir.
- XML nesnesiyse, tüm biçimlendirmesi korunmuş şekilde XML biçiminde bir dosyaya yazılır.
- ByteArray nesnesi ise, içerikleri herhangi bir dönüştürme olmadan doğrudan dosyaya yazılır.
- Başka bir nesne olması durumunda, `FileReference.save()` yöntemi nesnenin `toString()` yöntemini çağırır ve sonuçta elde edilen String değeri bir UTF-8 metin dosyasına kaydedilir. Nesnenin `toString()` yöntemi çağrılamazsa, bir hata fırlatılır.

Eğer `data` parametresinin değeri `null` ise, hata fırlatılır.

Aşağıdaki kod önceki bir örneği `FileReference.load()` yöntemi için genişletir. Bu örnek verileri dosyadan okuduktan sonra, kullanıcıdan bir dosya adı ister ve verileri yeni bir dosyaya kaydeder:

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

Verilerin tamamı dosyadan yüklendiğinde kod `onComplete()` yöntemini çağırır. `onComplete()` yöntemi yükleme olayları için dinleyicileri kaldırır ve sonra `saveFile()` yöntemini çağırır. `saveFile()` yöntemi`FileReference.save()` yöntemini çağırır. `FileReference.save()` yöntemi, kullanıcının dosyanın kaydedilmesi için yeni bir dosya adı ve konumu seçmesini sağlayan yeni bir iletişim kutusu açar. Kalan olay dinleyicisi yöntemleri, dosya kaydetme işleminin tamamlanıncaya kadar ilerlemesini izler.

Adobe AIR'de, FileStream sınıfı bir yerel dosyaya veri yazmaya yarayan ek işlev sağlar. Bkz. "Dosyaları okuma ve dosyalara yazma" sayfa 696

Bir sunucuya dosyalar yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir sunucuya dosya yüklemek için öncelikle kullanıcının bir veya daha fazla dosya seçmesine olanak sağlamak üzere `browse()` yöntemini çağırın. Daha sonra, `FileReference.upload()` yöntemi çağrıldığında, seçili dosya sunucuya aktarılır. Kullanıcı `FileReferenceList.browse()` yöntemini kullanarak birden çok dosya seçerse, Flash Player, seçilen dosyaların `FileReferenceList.fileList` adında bir dizisini oluşturur. Daha sonra dosyaların her birini ayrı ayrı yüklemek için `FileReference.upload()` yöntemini kullanabilirsiniz.

Not: *`FileReference.browse()` yönteminin kullanılması yalnızca tekli dosyalar yüklemenize olanak sağlar. Kullanıcının birden çok dosya yüklemesini sağlamak için, `FileReferenceList.browse()` yöntemini kullanın.*

Varsayılan olarak, sistem dosyası seçicisi iletişim kutusu, kullanıcıların yerel bilgisayardan herhangi türde bir dosya seçmesine olanak sağlar. Geliştiriciler, FileFilter sınıfını kullanıp dosya filtresi örneklerinin bir dizisini `browse()` yöntemine ileterek bir veya daha fazla özel dosya türü filtresi belirtebilir:

```
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);
```

Kullanıcı, sistem dosyası seçicisinde dosyaları seçip Aç düğmesini tıklattığında, `Event.SELECT` olayı gönderilir. Yüklenecek dosyayı seçmek için `FileReference.browse()` yöntemi kullanılırsa, aşağıdaki kod, dosyayı web sunucusuna gönderir.

```
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");
}
```

`POST` veya `GET` yöntemlerini kullanarak değişkenler göndermek için `URLRequest.method` ve `URLRequest.data` özelliklerini kullanarak `FileReference.upload()` yöntemiyle sunucuya veri gönderebilirsiniz.

`FileReference.upload()` yöntemini kullanarak bir dosyayı yüklemeyi denediğinizde, aşağıdaki olaylar gönderilir:

- `open` olayı (`Event.OPEN`): Yükleme işlemi başladığında gönderilir.
- `progress` olayı (`ProgressEvent.PROGRESS`): Dosyadan veri baytları karşıya yüklendikçe düzenli aralıklarla gönderilir.
- `complete` olayı (`Event.COMPLETE`): Yükleme işlemi başarıyla tamamlandığında gönderilir.
- `httpStatus` olayı (`HTTPStatusEvent.HTTP_STATUS`): Bir HTTP hatası nedeniyle karşıya yükleme işlemi başarısız olduğunda gönderilir.
- `httpResponseStatus` olayı (`HTTPStatusEvent.HTTP_RESPONSE_STATUS`): `upload()` veya `uploadUnencoded()` yöntemine yapılan bir çağrı HTTP üzerinden verilere erişmeye çalışıyorsa ve Adobe AIR, istek için durum kodunu algılayıp döndürebiliyorsa gönderilir.
- `securityError` olayı (`SecurityErrorEvent.SECURITY_ERROR`): Güvenlik ihlali nedeniyle karşıya yükleme işlemi başarısız olduğunda gönderilir.
- `uploadCompleteData` olayı (`DataEvent.UPLOAD_COMPLETE_DATA`): Başarılı bir karşıya yükleme işleminin ardından sunucudan veri alındıktan sonra gönderilir.

- `ioError` olayı (`IOErrorEvent.IO_ERROR`): Aşağıdaki nedenlerden herhangi birinden dolayı karşıya yükleme işlemi başarısız olduğunda gönderilir:
  - Flash Player dosya okuma, yazma veya iletme işlemi yaparken girdi/çıktı hatası oluştuğunda.
  - SWF, kimlik doğrulaması (örn. kullanıcı adı ve şifre) gerektiren bir sunucuya dosya yüklemeye çalıştığında. Yükleme sırasında, Flash Player kullanıcıların şifre girmesi için bir araç sağlamadığında.
  - `url` parametresi geçersiz bir protokol içerdiğinde. `FileReference.upload()` yönteminin, HTTP veya HTTPS kullanması gerektiğinde.

Flash Player, kimlik doğrulaması gerektiren sunucular için tam destek sunmaz. Yalnızca tarayıcı eklentisi veya Microsoft ActiveX® denetimi kullanan bir tarayıcıda çalışan SWF dosyaları, kullanıcıdan kimlik doğrulaması için ve daha sonra da indirme işlemi için kullanıcı adı ve şifre girmesini istemek üzere bir iletişim kutusu sağlayabilir. Eklenti veya ActiveX denetimi kullanılarak yapılan yüklemeler ya da bağımsız veya harici oynatıcı kullanılarak yapılan yükleme/indirme işlemleri için dosya aktarımı başarısız olur.

Flash Player uygulamasından bir dosya yükleme işlemini kabul etmek üzere ColdFusion'da sunucu komut dosyası oluşturmak için, aşağıdakine benzer kodu kullanabilirsiniz:

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#"
nameconflict="OVERWRITE" />
```

Bu ColdFusion kodu, Flash Player tarafından gönderilen dosyayı yükleyip ColdFusion şablonuyla aynı dizine kaydederek aynı ada sahip dosyanın üzerine yazar. Önceki kod, dosya yüklemesini kabul etmek için gerekli minimum kod miktarını gösterir; bu komut dosyasının üretim ortamında kullanılmaması gerekir. Kullanıcıların yalnızca kabul edilen dosya türlerini (örn. tehlikeli olabilecek bir sunucu tarafı komut dosyası yerine bir görüntü) yüklediğinden emin olmak için veri doğrulaması eklemek idealdir.

Aşağıdaki kod, PHP kullanarak dosya yüklemelerini gösterir ve veri doğrulaması içerir. Komut dosyası, yükleme dizininde yüklenmiş dosyaların sayısını 10 olarak sınırlar, dosyanın 200 KB'tan küçük olduğundan emin olur ve yalnızca JPEG, GIF veya PNG dosyalarının dosya sistemine yüklenip kaydedilmesine izin verir.

```
<?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;
    }
}
?>
```

`POST` veya `GET` istek yöntemini kullanarak yükleme komut dosyasına ek değişkenler iletebilirsiniz. Yükleme komut dosyanıza ek `POST` değişkenler göndermek için, aşağıdaki kodu kullanabilirsiniz:

```
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");
}
```

Önceki örnek, uzak sunucu tarafı komut dosyasına ileteceğiniz yeni bir URLVariables nesnesi oluşturur. Önceki ActionScript sürümlerinde, sorgu dizesindeki değerleri ileterek, sunucu yükleme komut dosyasına değişkenleri iletebiliyordunuz. ActionScript 3.0, `POST` veya `GET` yöntemini kullanarak veri iletmenize olanak sağlayan bir URLRequest nesnesini kullanarak uzak komut dosyasına değişkenler iletmenizi sağlar; bu da daha büyük veri kümelerinin iletilmesini daha kolay ve net duruma getirir. Değişkenlerin `GET` veya `POST` istek yöntemi kullanılarak iletilip iletilmediğini belirmek için, `URLRequest.method` özelliğini sırayla `URLRequestMethod.GET` veya `URLRequestMethod.POST` olarak ayarlayabilirsiniz

ActionScript 3.0, ayrıca önceki örnekte (`Filedata` varsayılan değerini `Custom1` olarak değiştiren) gösterildiği gibi, `upload()` yöntemine ikinci bir parametre sağlayarak varsayılan `Filedata` yükleme dosyası alan adını da geçersiz kılmanıza olanak sağlar.

`upload()` yöntemine üçüncü parametre olarak `true` değeri iletip bu varsayılanı geçersiz kılabilseniz de, varsayılan olarak Flash Player test yüklemesi göndermeyi denemez. Test yüklemesinin amacı, gerçek dosya yüklemesinin ve gerekiyorsa sunucu kimlik doğrulamasının başarılı olup olmayacağını kontrol etmektir.

Not: *Test yüklemesi şu anda yalnızca Windows tabanlı Flash Player uygulamalarında gerçekleşir.*

Dosya yüklemeyi işleyen sunucu komut dosyasının, aşağıdaki öğeleri içeren bir HTTP `POST` isteğini beklemesi gerekir:

- `multipart/form-data` değerine sahip `Content-Type`
- `name` niteliği " `Filedata`" öğesine ve `filename` niteliği orijinal dosyanın adına ayarlanmış şekilde `Content-Disposition`. `FileReference.upload()` yönteminde `uploadDataFieldName` parametresi için bir değer ileterek özel bir `name` niteliği belirtebilirsiniz.
- Dosyanın ikili içeriği.

Aşağıda örnek bir HTTP `POST` isteği verilmiştir:

```
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,,,)
```

Aşağıdaki örnek HTTP `POST` isteği, `api_sig`, `api_key` ve `auth_token` olmak üzere üç `POST` değişkeni gönderir ve özel bir `"photo"` yükleme verisi alan adı değerini kullanır:

```
POST /handler.asp HTTP/1.1
Accept: text/*
Content-Type: multipart/form-data;
boundary=----------Ij5ae0ae0KM7GI3KM7ei4cH2ei4gL6
User-Agent: Shockwave Flash
Host: www.mydomain.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache

------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Filename"

sushi.jpg
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_sig"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="api_key"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="auth_token"

XXXXXXXXXXXXXXXXXXXXXXXX
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="photo"; filename="sushi.jpg"
Content-Type: application/octet-stream

(actual file data,,,)
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data; name="Upload"

Submit Query
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7--
```

Sunucudan dosyaları indirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcıların `request` ve `defaultFileName` olmak üzere iki parametre alan `FileReference.download()` yöntemini kullanarak bir sunucudan dosya indirmesine olanak sağlayabilirsiniz. Birinci parametre, indirilecek dosyanın URL'sini içeren URLRequest nesnesidir. İkinci parametre isteğe bağlı olup dosya indirme iletişim kutusunda görüntülenen varsayılan bir dosya adı belirtmenizi sağlar. İkinci parametreyi (`defaultFileName`) çıkarırsanız, belirtilen URL'deki dosya adı kullanılır.

Aşağıdaki kod, SWF belgesiyle aynı dizinden index.xml adındaki bir dosyayı indirir:

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

Varsayılan adı index.xml yerine currentnews.xml olarak ayarlamak için, aşağıdaki kod parçasının gösterdiği gibi, `defaultFileName` parametresini belirtin:

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

Sunucunun dosya adı kullanışlı değilse veya sunucu tarafından geliştirilmediyse bir dosyayı yeni bir isim vermek faydalı olabilir. Dosyayı doğrudan indirmek yerine sunucu tarafı komut dosyası kullanarak indirdiğinizde, `defaultFileName` parametresini açıkça belirtmeniz de iyi olur. Örneğin, kendisine iletilen URL değişkenlerini esas alarak belirli dosyaları indiren sunucu tarafı komut dosyanız varsa, `defaultFileName` parametresini belirtmeniz gerekir. Aksi takdirde, indirilen dosyanın varsayılan adı, sunucu tarafı komut dosyanızın adıdır.

Ayrıştırılacak sunucu komut dosyasının URL'sine parametreler ekleyip `download()` yöntemi kullanılarak veriler sunucuya gönderilebilir. Aşağıdaki ActionScript 3.0 komut parçası, ColdFusion komut dosyasına iletilen parametreleri esas alarak bir belgeyi indirir:

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

Aşağıdaki kod, URL değişkeninin değerine bağlı olarak sunucudan iki dosyadan birini indiren download.cfm ColdFusion komut dosyasını gösterir:

```
<cfparam name="URL.id" default="1" />
<cfswitch expression="#URL.id#">
    <cfcase value="2">
        <cfcontent type="text/plain" file="#ExpandPath('two.txt')#" deletefile="No" />
    </cfcase>
    <cfdefaultcase>
        <cfcontent type="text/plain" file="#ExpandPath('one.txt')#" deletefile="No" />
    </cfdefaultcase>
</cfswitch>
```

FileReferenceList sınıfı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

FileReferenceList sınıfı, kullanıcının sunucu tarafı komut dosyasına yüklemek üzere bir veya daha çok dosya seçmesine olanak sağlar. Dosya yükleme, kullanıcının seçtiği her dosyada çağrılması gereken `FileReference.upload()` yöntemi tarafından işlenir.

Aşağıdaki kod, iki FileFilter nesnesi (`imageFilter` ve `textFilter`) oluşturur ve bunları bir dizide `FileReferenceList.browse()` yöntemine iletir. Bu da, işletim sistemi dosyası iletişim kutusunun dosya türleri için iki olası filtre görüntülemesine neden olur.

```
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.");
}
```

FileReferenceList birden çok dosya seçmenize olanak verse de, FileReferenceList sınıfı kullanılarak kullanıcının bir veya daha fazla dosyayı seçip yüklemesine olanak sağlanması, dosya seçmek için `FileReference.browse()` öğesinin kullanılmasıyla aynıdır. Birden çok dosya yüklemek için aşağıdaki kodun gösterdiği gibi `FileReference.upload()` öğesini kullanarak seçilen dosyaların her birini yüklemeniz gerekir:

```
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");
}
```

Dizideki FileReference nesnelerinin her birine `Event.COMPLETE` olayı eklendiğinden, Flash Player uygulaması her bir dosyanın yüklemesi bitince `completeHandler()` yöntemini çağırır.

AIR dosya sistemi API'sini kullanma

Adobe AIR 1.0 ve üstü

Adobe AIR dosya sistemi API'sine aşağıdaki sınıflar dahildir:

- File
- FileMode
- FileStream

Dosya sistemi API'si şunları (ve daha fazlasını) yapmanıza olanak verir:

- Dosyalar ve dizinleri kopyala, oluştur, sil, ve taşı
- Dosyalar ve dizinler hakkında bilgi alın
- Dosya oku ve yaz

AIR dosyası temel bilgileri

Adobe AIR 1.0 ve üstü

AIR sisteminde dosyayla çalışma hakkında çabuk açıklama ve kod örnekleri için aşağıdaki Adobe Geliştirici Bağlantısı'ndaki hızlı başlatma makalelerine bakın:

- Metin dosyası düzenleyicisi oluşturma (Flash)
- Metin dosyası düzenleyicisi oluşturma (Flex)
- Dizin arama uygulaması oluşturma) (Flex)
- XML tercihler dosyasından okuma ve yazma (Flex)
- Dosyaları ve verileri sıkıştırma) (Flex)

Adobe AIR hem dosyalara hem de klasörlere erişmek, bunları oluşturmak ve yönetmek için kullanabileceğiniz sınıflar sağlar. flash.filesystem paketinde bulunan bu sınıflar şu şekilde kullanılır:

Dosya sınıfları	Açıklama
File	File nesnesi bir dosya veya dizin yolunu temsil eder. Bir file nesnesini, dosya veya klasörle etkileşim başlatan bir dosyaya veya klasöre işaretçi oluşturmak için kullanırsınız.
FileMode	FileMode sınıfı, FileStream sınıfının `open()` ve `openAsync()` yöntemlerinin `fileMode` parametresinde kullanılan dize sabitlerini tanımlar. Bu yöntemlerin `fileMode` parametresi, dosya açıldıktan sonra FileStream nesnesi tarafından kullanılabilen, aralarında yazma, okuma, ekleme ve güncelleme bulunan özellikleri belirler.
FileStream	FileStream nesnesi dosyaları okuma ve yazma için açmada kullanılır. Yeni veya varolan bir dosyaya işaret eden bir File nesnesi oluşturduğunuzda, o işaretçiyi FileStream nesnesine iletirsiniz, böylece dosyayı açabilir ve veri okuyabilir ya da yazabilirsiniz.

File sınıfındaki bazı yöntemlerin hem senkronize hem de senkronize olmayan sürümleri vardır:

- `File.copyTo()` ve`File.copyToAsync()`
- `File.deleteDirectory()` ve `File.deleteDirectoryAsync()`
- `File.deleteFile()` ve `File.deleteFileAsync()`
- `File.getDirectoryListing()` ve `File.getDirectoryListingAsync()`
- `File.moveTo()` ve `File.moveToAsync()`
- `File.moveToTrash()` ve `File.moveToTrashAsync()`

Ayrıca FileStream işlemlerinin senkronize veya senkronize olmayan şekilde çalışmasını FileStream nesnesinin dosyayı açma şekli belirler: `open()` yöntemini çağırarak veya `openAsync()` yöntemini çağırarak.

Senkronize olmayan sürümler arka planda çalışan işlemleri başlatmanızı ve tamamlandığında (veya hata olayı oluştuğunda) olayları göndermenizi sağlar. Bu senkronize olmayan işlemler arka planda sürerken, başka kod çalışabilir. İşlemlerin senkronize olmayan sürümleriyle, işlevi çağıran File veya FileStream nesnesinin `addEventListener()` yöntemini kullanarak olay dinleyicisi işlevleri oluşturmalısınız.

Senkronize sürümler sayesinde, olay dinleyicileri ayarlamaya dayanmayan daha basit kod yazabilirsiniz. Ancak, senkronize bir yöntem çalışırken başka kod çalışamadığı için, görüntüleme nesnesi oluşturma ve animasyon gibi önemli işlemler geciktirilebilir.

AIR'de File nesneleriyle çalışma

Adobe AIR 1.0 ve üstü

File nesnesi dosya sistemindeki bir dosyaya veya dizine işaretçidir.

File sınıfı FileReference sınıfını genişletir. AIR'nin yanı sıra Adobe® Flash® Player'da da mevcut olan FileReference sınıfı, bir dosyaya işaretçiyi temsil eder. File sınıfı güvenlik nedenleriyle Flash Player'da açık olmayan özellikleri ve yöntemleri ekler (tarayıcıda çalışan bir SWF dosyasında).

File sınıfı hakkında

Adobe AIR 1.0 ve üstü

File sınıfını aşağıdaki işlemler için kullanabilirsiniz:

- Kullanıcı dizini, kullanıcının documents dizini, uygulamanın başlatıldığı dizin ve uygulama dizini de dahil olmak üzere özel dizinlerin yolunu alma
- Dosyaları ve dizinleri kopyalama
- Dosyaları ve dizinleri taşıma
- Dosyaları ve dizinleri silme (veya çöp kutusuna taşıma)
- Bir dizinde bulunan dosyaları ve dizinleri listeleme
- Geçici dosyalar ve klasörler oluşturma

Bir File nesnesi dosya yoluna işaret ettikten sonra, FileStream sınıfını kullanarak dosya verilerini okumak ve yazmak için dosya yolunu kullanabilirsiniz.

File nesnesi henüz var olmayan bir dosyanın veya dizinin yoluna işaret edebilir. Böyle bir File nesnesini dosya veya dizin oluştururken kullanabilirsiniz.

File nesnelerinin Yolları

Adobe AIR 1.0 ve üstü

Her File nesnesi, yolunu tanımlayan iki özelliğe sahiptir:

Özellik	Açıklama
`nativePath`	Bir dosyanın platforma özgü yolunu belirtir. Örneğin Windows'ta bir yol "c:\Sample directory\test.txt" olurken Mac OS'de "/Sample directory/test.txt" olabilir. `nativePath` özelliği, Windows'ta dizin ayırıcı karakteri olarak ters eğik çizgi (\) karakterini ve Mac OS ile Linux'ta eğik çizgi (/) karakterini kullanır.
`url`	Bir dosyaya işaret etmek için, dosya URL şemasını kullanabilir. Örneğin Windows'ta bir yol "file:///c:/Sample%20directory/test.txt" olurken Mac OS'de "file:///Sample%20directory/test.txt" olabilir. Çalışma zamanı `file` öğesinin yanında başka özel URL şemaları da içerir ve bunlar "Desteklenen AIR URL şemaları" sayfa 684 bölümünde anlatılmaktadır

File sınıfı, Mac OS, Windows ve Linux'ta standart dizinlere işaret etmek üzere statik özellikler içerir. Bu özellikler şunları içerir:

- `File.applicationStorageDirectory`, yüklü olan her AIR uygulamasına özgü bir depolama dizini. Bu dizin dinamik uygulama varlıklarını ve kullanıcı tercihlerini saklamak için uygun bir konumdur. Büyük miktardaki verileri başka bir alanda saklamayı düşünün.

  Android ve iOS'de, uygulama kaldırıldığında veya kullanıcı uygulama verilerini silmeyi seçtiğinde uygulama depolama dizini kaldırılır, ancak diğer platformlarda bu yaşanmaz.
- `File.applicationDirectory`: uygulamanın yüklendiği dizin (yüklü olan diğer varlıklarla birlikte). Bazı işletim sistemlerinde, uygulama fiziksel bir dizin yerine tek bir pakette depolanır. Bu durumda içeriğe yerel yol kullanılarak erişilemeyebilir. Uygulama dizini salt okunurdur.
- `File.desktopDirectory`: kullanıcının masaüstü dizini. Bir platform masaüstü dizini tanımlamıyorsa dosya sistemindeki başka bir konum kullanılır.
- `File.documentsDirectory`: kullanıcının belgeler dizini. Bir platform belgeler dizini tanımlamıyorsa dosya sistemindeki başka bir konum kullanılır.
- `File.userDirectory`: kullanıcı dizini. Bir platform kullanıcı dizini tanımlamıyorsa dosya sistemindeki başka bir konum kullanılır.

Not: *Bir platform masaüstü, belgeler veya kullanıcı dizinleri için standart konumlar tanımlamıyorsa,* `File.documentsDirectory`*,* `File.desktopDirectory` *ve* `File.userDirectory` *aynı dizine başvurabilir.*

Bu özelliklerin farklı işletim sistemlerinde farklı değerleri vardır. Örneğin, Mac ve Windows'ta kullanıcı masaüstü dizini için farklı yerel yollar vardır. Ancak, `File.desktopDirectory` özelliği her platformda uygun bir dizin yolunu gösterir. Farklı platformlarda düzgün çalışan uygulamalar yazmak için, bu özellikleri uygulama tarafından kullanılan diğer dizin ve dosyaları başvuru olarak almak için temel olarak kullanın. Sonra, yolu daraltmak için `resolvePath()` yöntemini kullanın. Örneğin, bu kod uygulama saklama dizinindeki preferences.xml dosyasına işaret ediyor.

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

File sınıfı belirli bir dosya yoluna işaret etmenize izin verse de, bunu yapmak farklı platformlar üzerinde çalışmayan uygulamalara sebep olabilir. Örneğin, C:\Documents and Settings\cem\ yalnızca Windows'da çalışır. Bu sebeplerden dolayı, en iyisi File sınıfının `File.documentsDirectory` gibi statik özelliklerini kullanmaktır.

Ortak dizin konumları

Platform	Dizin türü	Tipik dosya sistemi konumu
Android	Uygulama	`/data/data/`
	Uygulama depolama	`/data/data/air.applicationID/filename/Local Store`
	Önbellek	/data/data/applicationID/cache
	Masaüstü	`/mnt/sdcard`
	Belgeler	`/mnt/sdcard`
	Geçici	`/data/data/applicationID/cache/FlashTmp.randomString`
	Kullanıcı	`/mnt/sdcard`
iOS	Uygulama	`/var/mobile/Applications/uid/filename.app`
	Uygulama depolama	`/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store`
	Önbellek	/var/mobile/Applications/uid/Library/Caches
	Masaüstü	erişilmez
	Belgeler	`/var/mobile/Applications/uid/Documents`
	Geçici	`/private/var/mobile/Applications/uid/tmp/FlashTmpNNN`
	Kullanıcı	erişilmez
Linux	Uygulama	`/opt/filename/share`
	Uygulama depolama	`/home/userName/.appdata/applicationID/Local Store`
	Masaüstü	`/home/userName/Desktop`
	Belgeler	`/home/userName/Documents`
	Geçici	`/tmp/FlashTmp.randomString`
	Kullanıcı	`/home/userName`
Mac	Uygulama	`/Applications/filename.app/Contents/Resources`
	Uygulama depolama	`/Users/`***`userName`***`/Library/Preferences/`***`applicationid`***`/Local Store` (AIR 3.2 ve öncesi) ***`path`***`/Library/Application Support/`***`applicationid`***`/Local Store` (AIR 3.3 ve üstü): Burada yol `/Users/`***`userName`***`/Library/Containers/`***`bundle-id`***`/Data` (korumalı ortam) veya `/Users/userName` (korumalı bir ortamın dışında çalıştırırken) şeklindedir.
	Önbellek	/Users/userName/Library/Caches
	Masaüstü	`/Users/`***`userName`***`/Desktop`
	Belgeler	`/Users/`***`userName`***`/Documents`
	Geçici	`/private/var/folders/JY/randomString/TemporaryItems/FlashTmp`
	Kullanıcı	`/Users/`***`userName`***

Platform	Dizin türü	Tipik dosya sistemi konumu
Windows	Uygulama	`C:\Program Files\filename`
	Uygulama depolama	`C:\Documents and settings\userName\ApplicationData\applicationID\Local Store`
	Önbellek	C:\Documents and settings\userName\Local Settings\Temp
	Masaüstü	`C:\Documents and settings\userName\Desktop`
	Belgeler	`C:\Documents and settings\userName\My Documents`
	Geçici	`C:\Documents and settings\userName\Local Settings\Temp\randomString.tmp`
	Kullanıcı	`C:\Documents and settings\userName`

Bu dizinlerin gerçek yerel yolları işletim sistemine ve bilgisayar konfigürasyonuna göre değişir. Bu tabloda gösterilen yollar tipik örneklerdir. Uygulamanızın herhangi bir platformda düzgün çalışmasını sağlamak için bu dizinlere başvururken uygun statik File sınıfı özelliklerini kullanmanız gerekir. Gerçek bir AIR uygulamasında tabloda gösterilen `applicationID` ve `filename` değerleri uygulama açıklayıcısından alınır. Uygulama açıklayıcısında bir yayıncı kimliği belirtirseniz, yayıncı kimliği bu yollarda uygulama kimliğine eklenir. `userName` değeri yüklemeyi yapan kullanıcının hesap adıdır.

File nesnesini bir dizine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini bir dizine işaret etmesi için ayarlamanın farklı yolları vardır.

Kullanıcının ana dizinine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini kullanıcının ana dizinine yönlendirebilirsiniz. Aşağıdaki kod bir File nesnesini ana dizinin AIR Test alt dizinine işaret etmesi için ayarlar:

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

Kullanıcının documents dizinine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini kullanıcının documents dizinine yönlendirebilirsiniz. Aşağıdaki kod bir File nesnesini documents dizininin AIR Test alt dizinine işaret etmesi için ayarlar:

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

Masaüstü dizinine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini masaüstüne işaret edebilirsiniz. Aşağıdaki kod bir File nesnesini masaüstünün AIR Test alt dizinine işaret etmesi için ayarlar:

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

Uygulama depolama dizinine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini uygulama depolama dizinine işaret edebilirsiniz. Her AIR uygulaması için, uygulama depolama dizinini tanımlayan ilişkilendirilmiş, benzersiz bir yol vardır. Bu dizin her uygulama ve kullanıcı için benzersizdir. Bu dizini kullanıcıya özgü, uygulamaya özgü verileri saklamak için kullanabilirsiniz (kullanıcı verileri ve tercihler dosyaları gibi). Örneğin aşağıdaki kod bir File nesnesini uygulama depolama dizininde bulunan bir tercihler dosyası olan prefs.xml dosyasına işaret eder:

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

Uygulama deposu dizini konumu genellikle kullanıcı adı ve uygulama kimliğine bağlıdır. Aşağıdaki dosya sistemi konumları uygulamanızda hata ayıklamaya yardımcı olmak için verilmiştir. Bu dizindeki dosyaları çözümlemek için her zaman `File.applicationStorage` özelliğini veya `app-storage:` URI şemasını kullanmanız gerekir:

- Mac OS'de AIR sürümüne göre değişiklik gösterir:

  AIR 3.2 ve öncesi: `/Users/user name/Library/Preferences/`*`applicationID`*`/Local Store/`

  AIR 3.3 ve üstü: *`path`*`/Library/Application Support/`*`applicationID`*`/Local Store`: Burada *`path`*`/Users/`*`username`*`/Library/Containers/`*`bundle-id`*`/Data` (korumalı ortam) veya `/Users/`*`username`* (korumalı ortamın dışında çalıştırırken) şeklindedir

  Örneğin (AIR 3.2):

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- Windows'ta—Belgeler ve Ayarlar dizininde:

  C:\Documents and Settings\user name`\Application Data\`*applicationID*`\Local Store\`

  Örneğin:

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- Linux'ta:

  `/home/`*kullanıcı adı*`/.appdata/`*applicationID*`/Local Store/`

  Örneğin:

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- Android'de şu konumda:

  /data/data/`androidPackageID`/`applicationID/Local Store`

  Örneğin:

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

Not: *Bir uygulamanın yayıncı kimliği varsa, yayıncı kimliği aynı zamanda uygulama depolama dizinine giden yolun bir parçası olarak kullanılır.*

`File.applicationStorageDirectory` ile oluşturulan bir File nesnesinin URL'si (ve `url` özelliği), aşağıda gösterildiği gibi `app-storage` URL şemasını kullanır (bkz. "Desteklenen AIR URL şemaları" sayfa 684):

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

Uygulama dizinine işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesini uygulamanın yüklendiği, uygulama dizini olarak bilinen dizine işaret edebilirsiniz. Bu dizine `File.applicationDirectory` özelliğini kullanarak başvurabilirsiniz. Bu dizini uygulama tanımlayıcı dosyasını veya uygulamayla yüklenen başka kaynakları incelemek için kullanabilirsiniz. Örneğin aşağıdaki kod bir File nesnesini uygulama dizinindeki *images* adlı bir dizine işaret eder:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

`File.applicationDirectory` ile oluşturulan bir File nesnesinin URL'si (ve `url` özelliği), aşağıda gösterildiği gibi `app` URL şemasını kullanır (bkz. "Desteklenen AIR URL şemaları" sayfa 684):

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

Not: *Android'de uygulama paketi içindeki dosyalara `nativePath` aracılığıyla erişilmez. `nativePath` özelliği boş bir dizedir. Uygulama dizinindeki dosyalara erişmek için her zaman yerel yol yerine URL'yi kullanın.*

Önbellek dizinine işaret etme

Adobe AIR 3.6 ve üstü

`File.cacheDirectory` özelliğini kullanarak işletim sisteminin geçici veya önbellek dizinine bir File nesnesi işaret edebilirsiniz. Bu dizinde, uygulama tarafından çalıştırılması gerekmeyen ve silinmeleri durumunda kullanıcı için sorun oluşturmayacak veya veri kaybına neden olmayacak geçici dosyalar bulunur.

Çoğu işletim sisteminde önbellek dizini geçici bir dizindir. iOS'ta önbellek dizini, uygulama kütüphanesinin Önbellekler dizinine karşılık gelir. Bu dizindeki dosyalar, çevrimiçi depolama alanına yedeklenmez ve aygıtın kullanılabilir depolama alanı çok düşükse işletim sistemi tarafından olanak dahilinde silinebilir. Daha fazla bilgi için bkz. "Dosya yedeklemeyi ve önbelleğe almayı denetleme" sayfa 684.

Dosya sistemi köküne işaret etme

Adobe AIR 1.0 ve üstü

`File.getRootDirectories()` yöntemi bir Windows bilgisayarındaki C: ve bağlanmış birimler gibi tüm kök birimleri listeler. Bu yöntem, Mac OS ve Linux'ta her zaman bilgisayar için benzersiz bir kök dizin ("/" dizini) döndürür. `StorageVolumeInfo.getStorageVolumes()` yöntemi monte edilmiş depolama birimleri hakkında daha ayrıntılı bilgi sağlar (bkz. "Depolama birimleriyle çalışma" sayfa 695).

Not: *Dosya sisteminin kökü Android'de okunabilir değildir. Dizine yerel yol ile başvuran bir File nesnesi olan "/" öğesi döndürülür, ancak bu nesnenin özellikleri doğru değerlere sahip değildir. Örneğin, `spaceAvailable` her zaman 0 değerine sahiptir.*

Açık bir dizine işaret etme

Adobe AIR 1.0 ve üstü

Aşağıdaki örnekte olduğu gibi, File nesnesinin `nativePath` özelliğini ayarlayarak File nesnesini açık bir dizine işaret edebilirsiniz (Windows'ta):

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

Önemli: Bu şekilde açık bir yola işaret etmek farklı platformlar üzerinde çalışmayan koda sebep olabilir. Örneğin, önceki örnek yalnızca Windows'da çalışır. Platformlar arası çalışan bir dizini bulmak için File nesnesinin `File.applicationStorageDirectory` gibi statik özelliklerini kullanabilirsiniz. Daha sonra bir göreceli yola gitmek için `resolvePath()` (bir sonraki kısma bakın) yöntemini kullanabilirsiniz.

Göreceli yollara gitme

Adobe AIR 1.0 ve üstü

Belirli bir yola göreceli olan bir yol edinmek için `resolvePath()` yöntemini kullanabilirsiniz. Aşağıdaki kod bir File nesnesini kullanıcının ana dizininin "AIR Test" alt dizinine işaret etmesi için ayarlar:

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

Ayrıca, aşağıda gösterildiği gibi File nesnesini URL dizesine bağlı bir dizine işaret etmek için de nesnenin `url` özelliğini kullanabilirsiniz:

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

Daha fazla bilgi için bkz. "Dosya yollarını değiştirme" sayfa 683.

Kullanıcının bir dizin seçmek için göz atmasını sağlama

Adobe AIR 1.0 ve üstü

File sınıfı, kullanıcının nesneye atamak için bir dizin seçebileceği sistem iletişim kutusu sağlayan `browseForDirectory()` yöntemini içerir. `browseForDirectory()` yöntemi senkronize değildir. File nesnesi kullanıcı bir dizin seçip Aç düğmesini tıklatırsa bir `select` olayı veya kullanıcı İptal düğmesini tıklatırsa bir `cancel` olayı gönderir.

Örneğin aşağıdaki kod kullanıcının bir dizin seçmesine olanak tanır ve seçilirse dizin yolunu çıkarır:

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

Not: *Android'de `browseForDirectory()` yöntemi desteklenmez. Bu yöntemi çağırma işleminin bir etkisi yoktur; hemen bir cancel olayı gönderilir. Kullanıcıların dizin seçmesine izin vermek için bunun yerine özel, uygulama tanımlı bir iletişim kutusu kullanın.*

Uygulamanın çağrıldığı dizine işaret etme

Adobe AIR 1.0 ve üstü

Uygulama çağrıldığında gönderilen InvokeEvent nesnesinin `currentDirectory` özelliğini işaretleyerek bir uygulamanın çağrıldığı dizin konumunu alabilirsiniz. Ayrıntılar için bkz. "Komut satırı argümanlarını yakalama" sayfa 888.

File nesnesini bir dosyaya işaret etme

Adobe AIR 1.0 ve üstü

Bir File nesnesinin işaret ettiği dosyayı ayarlamanın farklı yolları vardır.

Açık dosya yoluna işaret etme

Adobe AIR 1.0 ve üstü

Önemli: Açık bir yola işaret etmek farklı platformlar üzerinde çalışmayan koda sebep olabilir. Örneğin C:/foo.txt yolu yalnızca Windows'da çalışır. Platformlar arası çalışan bir dizini bulmak için File nesnesinin `File.applicationStorageDirectory` gibi statik özelliklerini kullanabilirsiniz. Daha sonra bir göreceli yola gitmek için `resolvePath()` (bkz. "Dosya yollarını değiştirme" sayfa 683) yöntemini kullanabilirsiniz.

Aşağıda gösterildiği gibi File nesnesini URL dizesine bağlı bir dosyaya veya dizine işaret etmek için de nesnenin `url` özelliğini kullanabilirsiniz:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

Ayrıca URL'yi aşağıda olduğu gibi `File()` yapıcı işlevine de iletebilirsiniz:

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

`url` özelliği her zaman URL'nin URI kodlu sürümünü döndürür (örneğin boşluklar `"%20` ile değiştirilir):

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

Ayrıca açık bir yol ayarlamak için File nesnesinin `nativePath` özelliğini de kullanabilirsiniz. Örneğin aşağıdaki kod Windows yüklü bir bilgisayarda çalıştırıldığında, File nesnesini, C: sürücüsünün AIR Test alt dizinindeki test.txt dosyasına ayarlar:

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

Ayrıca bu yolu aşağıda olduğu gibi `File()` yapıcı işlevine de iletebilirsiniz:

```
var file:File = new File("C:/AIR Test/test.txt");
```

`nativePath` özelliği için yol sınırlayıcı olarak eğik çizgi (/) karakterini kullanın. Windows'da, ters eğik çizgi (\) karakterini de kullanabilirsiniz, ancak bunu yapmak farklı platformlar üzerinde çalışmayan uygulamalara sebep olabilir.

Daha fazla bilgi için bkz. "Dosya yollarını değiştirme" sayfa 683.

Bir dizindeki dosyaları numaralandırma

Adobe AIR 1.0 ve üstü

Bir dizinin kök düzeyinde dosyalara ve alt dizinlere işaret eden File nesnelerinin bir dizisini almak için File nesnesinin `getDirectoryListing()` yöntemini kullanabilirsiniz. Daha fazla bilgi için bkz. "Dizinleri numaralandırma" sayfa 690.

Kullanıcının bir dosya seçmek için gözatmasını sağlama

Adobe AIR 1.0 ve üstü

File sınıfı, kullanıcının nesneye atamak için bir dosya seçebileceği sistem iletişim kutusu sağlayan aşağıdaki yöntemleri içerir.

- `browseForOpen()`
- `browseForSave()`
- `browseForOpenMultiple()`

Bu yöntemlerin tümü senkronize olmayan yöntemlerdir. Kullanıcı bir dosya seçtiğinde (veya browseForSave() durumunda hedef yol) `browseForOpen()` ve `browseForSave()` yöntemleri select olayını gönderir. `browseForOpen()` ve `browseForSave()` yöntemleriyle, seçimden sonra hedef File nesnesi seçili dosyalara işaret eder. Kullanıcı dosyaları seçtiğinde, `browseForOpenMultiple`() yöntemi bir `selectMultiple` olayı gönderir. `selectMultiple` olayı, File nesneleri dizisi olan bir `files` özelliğine sahip olan (seçili dosyalara işaret eden) FileListEvent türüdür.

Örneğin aşağıdaki kod kullanıcıya, kullanıcının bir dosya seçebileceği "Aç" iletişim kutusunu sunar:

```
 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);
}
```

Bir browse yöntemini çağırdığınızda uygulamada başka bir tarayıcı iletişim kutusu açıksa, çalışma zamanı bir Hata istisnası atar.

Not: *Android'de `browseForOpen()` ve `browseForOpenMultiple()` yöntemleri ile yalnızca görüntü, video ve ses dosyaları seçilebilir. Ayrıca, browseForSave() iletişim kutusu da kullanıcı rastgele dosya adı girebilse bile yalnızca medya dosyalarını görüntüler. Medya dosyası olmayan dosyaları açmak ve kaydetmek için bu yöntemler yerine özel iletişim kutuları kullanmayı göz önünde bulundurmalısınız.*

Dosya yollarını değiştirme

Adobe AIR 1.0 ve üstü

Ayrıca aşağıdaki örneklerde olduğu gibi, varolan bir File nesnesinin yolunu `resolvePath()` yöntemini çağırarak veya nesnenin `nativePath` ya da `url` özelliğini değiştirerek de değiştirebilirsiniz:

```
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
```

`nativePath` özelliğini kullanırken, eğik çizgi (/) karakterini dizin ayırıcı karakter olarak kullanın. Windows'da ters eğik çizgi (\) karakterini de kullanabilirsiniz, ancak bu platformlar arasında çalışmayan koda sebep olacağından bunu yapmamalısınız.

Desteklenen AIR URL şemaları

Adobe AIR 1.0 ve üstü

AIR'de bir File nesnesinin `url` özelliğini tanımlarken, aşağıdaki URL şemalarından herhangi birini kullanabilirsiniz:

URL şeması	Açıklama
file	Dosya sisteminin köküne göreceli olan bir yol belirtmek için kullanın. Örneğin: `file:///c:/AIR Test/test.txt` URL standardı bir dosya URL'sinin `file://<host>/<path>` formunu alacağını belirtir. Özel bir durum olarak `<host>`, "URL'nin yorumlandığı bilgisayar" olarak yorumlanan boş dize olabilir. Bu nedenle dosya URL'leri genellikle üç eğik çizgiye (///) sahiptir.
app	Yüklü uygulamanın kök dizinine göreceli bir yol belirtmek için kullanın (yüklü uygulama için application.xml dosyasını içeren dizin). Örneğin aşağıdaki yol, yüklü uygulamanın dizininin görüntüler alt dizinine işaret eder: `app:/images`
app-storage	Uygulama depolama dizinine göreceli bir yol belirtmek için kullanın. AIR her yüklenen uygulama için, o uygulamaya özgü verilerin saklanmasına yönelik kullanışlı bir yer olan benzersiz bir uygulama depolama dizini tanımlar. Örneğin aşağıdaki yol, uygulama depolama dizininin ayarlar alt dizininde bulunan bir prefs.xml dosyasını işaret eder: `app-storage:/settings/prefs.xml`

Dosya yedeklemeyi ve önbelleğe almayı denetleme

Adobe AIR 3.6 ve üstü, yalnızca iOS ve OS X

Aralarında en dikkat çekeni iOS ve Mac OS X olan bazı işletim sistemleri kullanıcılara uygulama dosyalarını uzak bir depolama alanına otomatik olarak yedekleme olanağı tanır. Ayrıca, iOS'ta dosyaların yedeklenip yedeklenemeyebileceği ve farklı amaçlara yönelik dosyaların nereye depolanabileceği konusunda kısıtlamalar bulunur.

Aşağıdakiler, dosya yedekleme ve depolama konusunda Apple talimatlarıyla nasıl uyum sağlanacağını özetlemektedir. Daha fazla bilgi için sonraki bölümlere bakın.

- Bir dosyayı yedeklemeye gerek olmadığını ve (yalnızca iOS) aygıt depolama alanının azalması durumunda işletim sistemi tarafından silinebileceğini belirtmek için dosyayı önbellek dizinine (`File.cacheDirectory`) kaydedin. Bu, iOS'ta tercih edilen depolama konumudur ve yeniden oluşturulabilen veya yeniden indirilebilen birçok dosya için kullanılmalıdır.

- Bir dosyayı yedeklemeye gerek olmadığını ancak işletim sistemi tarafından silinmemesi gerektiğini belirtmek için, dosyayı uygulama kütüphane dizinlerinden birine (uygulama depolama dizini gibi) (`File.applicationStorageDirectory`) veya belge dizinine (`File.documentsDirectory`) kaydedin. File nesnesinin `preventBackup` özelliğini `true` olarak ayarlayın. Bu, yeniden oluşturulabilen veya tekrar indirilebilen ancak çevrimiçi kullanım sırasında uygulamanızın düzgün çalışması için gerekli olan içerik için Apple tarafından gerekli görülür.

Yedekleme için dosyaları belirtme

Yedekleme alanından tasarruf etme ve ağ bant genişliği kullanımını azaltma amacıyla, iOS ve Mac uygulamalarına yönelik Apple talimatlarında, yalnızca kullanıcı girişli veriler veya farklı şekilde yeniden oluşturulamayan ya da indirilemeyen veriler içeren dosyaların yedekleme için tanımlanması gerektiği belirtilmektedir.

Varsayılan olarak, uygulama kütüphanesi klasörlerindeki tüm dosyalar yedeklenir. Mac OS X'te bu, uygulama depolama dizinidir. iOS'ta buna uygulama depolama dizini, uygulama dizini, masaüstü dizini, belge dizini ve kullanıcı dizini (çünkü bu dizinler iOS'ta uygulama kütüphanesine eşleştirilir) dahildir. Sonuç olarak, bu dizinlerdeki dosyalar varsayılan olarak sunucu depolama alanına yedeklenir.

Uygulamanız tarafından yeniden oluşturulabilen bu konumlardan birine bir dosya kaydediyorsanız, dosyayı bayraklayarak işletim sisteminin bunu yedeklemeyeceğini bilmesini sağlayabilirsiniz. Bir dosyanın yedeklenmemesi gerektiğini göstermek için, File nesnesinin `preventBackup` özelliğini `true` olarak ayarlayın.

Unutmayın, iOS'ta uygulama kütüphanesi klasörlerinden herhangi birindeki bir dosya (dosyanın `preventBackup` özelliği `true` ayarlanmış olsa bile), işletim sistemi tarafından silinmemesi gereken kalıcı bir dosya olarak bayraklanır.

Dosyanın önbelleğe alınmasını ve silinmesini denetleme

iOS uygulamalarına yönelik Apple talimatlarında, yeniden oluşturulabilen içeriğin (mümkün olduğunca), aygıttaki depolama alanının azalması durumunda işletim sistemi tarafından silinmesinin sağlanması gerektiği belirtilir.

iOS'ta, uygulama kütüphanesi klasörlerindeki dosyalar (uygulama depolama dizini veya belge dizini gibi) kalıcı olarak bayraklanır ve işletim sistemi tarafından silinmez.

Uygulama tarafından yeniden oluşturulabilen ve uygulama önbelleği dizinindeki depolama alanı azaldığında rahatlıkla silinebilen dosyalar kaydedin. `File.cacheDirectory` statik özelliğini kullanarak önbellek dizinine erişin.

iOS'ta önbellek dizini, uygulamanın önbellek dizinine (<Application Home>/Library/Caches) karşılık gelir. Diğer işletim sistemlerinde bu dizin, benzer bir dizine eşleştirilir. Örneğin, Mac OS X'te uygulama kütüphanesindeki Önbellekler dizinine de eşleştirilir. Android'de önbellek dizini uygulamanın önbellek dizinine eşleştirilir. Windows'ta önbellek dizini işletim sistemi geçici dizinine eşleştirilir. Hem Android'de hem de Windows'ta bu, File sınıfına `createTempDirectory()` ve `createTempFile()` yöntemlerine yapılan bir çağrıyla erişilen dizinle aynı dizindir.

İki dosya arasındaki göreceli yolu bulma

Adobe AIR 1.0 ve üstü

İki dosya arasındaki göreceli yolu bulmak için `getRelativePath()` yöntemini kullanabilirsiniz:

```
 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
```

`getRelativePath()` yönteminin ikinci parametresi olan `useDotDot` parametresi, `..` sözdiziminin üst dizinleri göstermek için sonuçlarda döndürülmesine izin verir:

```
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
```

Dosya adlarının kurallı sürümlerini alma

Adobe AIR 1.0 ve üstü

Windows ve Mac OS'de dosya ve yol adları büyük/küçük harfe duyarlı değildir. Aşağıdaki örnekte iki File nesnesi aynı dosyaya işaret etmektedir:

```
File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

Ancak belgeler ve dizin adları büyük harf içerir. Örneğin aşağıda, aşağıdaki örneklerde olduğu gibi documents dizininde AIR Test adlı bir klasör olduğunu varsayılmıştır:

```
var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

`canonicalize()` yöntemi, dosya veya dizin adı için büyük/küçük harfle yazımı doğru biçimde kullanmak üzere `nativePath` nesnesini dönüştürür. Büyük/küçük harfe duyarlı dosya sistemlerinde (Linux gibi), yalnızca büyük/küçük harfle yazım açısından farklı olan adlara sahip birden çok dosya bulunuyorsa, `canonicalize()` yöntemi, bulunan ilk dosyayla eşleşecek biçimde yolu ayarlar (dosya sistemi tarafından belirlenen düzende).

Aşağıdaki örneklerde gösterildiği gibi Windows'ta kısa dosya adlarını ("8.3" adlar) uzun dosya adlarına dönüştürmek için `canonicalize()` yöntemini de kullanabilirsiniz:

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

Paketlerle ve sembolik bağlarla çalışma

Adobe AIR 1.0 ve üstü

Birçok işletim sistemi paket dosyalarını ve sembolik bağ dosyalarını destekler:

Paketler—Mac OS'de, dizinler paketler olarak belirlenebilir ve Mac OS Finder'da dizin olarak değil, tek bir dosya olarak görünebilir.

Sembolik bağlar—Mac OS, Linux ve Windows Vista, sembolik bağları destekler. Sembolik bağlar bir dosyanın diskteki başka bir dosyaya veya dizine işaret etmesini sağlar. Sembolik bağlar başka adlara benzer olsa da, aynı değildir. Başka ad her zaman bir dosya olarak bildirilir (dizin olarak değil) ve başka ada veya kısayola okuma ya da yazma işlemi hiçbir zaman işaret ettiği orijinal dosyayı veya dizini etkilemez. Diğer yandan sembolik bağ tam olarak işaret ettiği dosya veya dizin gibi davranır. Bir dosya veya dizin olarak bildirilebilir ve sembolik bağa okuma veya yazma işlemi sembolik bağın kendisini değil, işaret ettiği dosyayı veya dizini etkiler. Ayrıca, Windows'ta birleşim noktasını (NTFS dosya sisteminde kullanılan) referans alan bir File nesnesine ilişkin `isSymbolicLink` özelliği `true` olarak ayarlanır.

File sınıfı, bir File nesnesinin bir pakete veya sembolik bağa başvurup başvurmadığını kontrol etmek için `isPackage` ve `isSymbolicLink` özelliklerini içerir.

Aşağıdaki kod kullanıcının masaüstü dizini üzerinden, paket *olmayan* alt dizinleri listeleyerek yinelenir:

```
 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);
    }
}
```

Aşağıdaki kod kullanıcının masaüstü dizini üzerinden, sembolik bağlar *olmayan* dosyaları ve dizinleri listeleyerek yinelenir:

```
 var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

`canonicalize()` yöntemi, bağın başvurduğu dosyaya veya dizine işaret etmek için sembolik bir bağın yolunu değiştirir. Aşağıdaki kod kullanıcının masaüstü dizini üzerinden yinelenir ve sembolik bağlar olan dosyalar tarafından başvurulan yolları bildirir:

```
 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);
    }
}
```

Bir birimdeki kullanılabilir alanı belirleme

Adobe AIR 1.0 ve üstü

Bir File nesnesinin `spaceAvailable` özelliği, Dosya konumunda bayt olarak kullanılabilir olan alandır. Örneğin aşağıdaki kod uygulama depolama dizininde kullanılabilir olan alanı kontrol eder:

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

File nesnesi bir dizine başvururse, `spaceAvailable` özelliği, dizinde dosyaların kullanabileceği alanı gösterir. File nesnesi bir dosyaya başvururse, `spaceAvailable` özelliği dosyanın büyütülebileceği alanı gösterir. Dosya konumu yoksa, `spaceAvailable` özelliği 0'a ayarlanır. File nesnesi sembolik bir bağa başvururse, `spaceAvailable` özelliği sembolik bağın işaret ettiği konumdaki kullanılabilir alana ayarlanır.

Genellikle bir dizin veya dosya için kullanılabilir olan alan, dizini veya dosyayı içeren birimdeki kullanılabilir alanla aynıdır. Ancak kullanılabilir alan, kotaları ve dizin başına sınırlamaları hesaba katabilir.

Bir dosyayı veya dizini bir birime eklemek genellikle dosyanın gerçek boyutundan veya dizin içeriğinin boyutundan daha fazla alan gerektirir. Örneğin işletim sistemi dizin bilgilerini saklamak için daha fazla alan gerektirebilir. Veya gerekli disk kesimleri ek alan kullanabilir. Ayrıca kullanılabilir alanlar dinamik olarak değişir. Dolayısıyla bildirilen alanın tamamını dosya depolama için ayıramazsınız. Dosya sistemine yazma hakkında bilgi için bkz. "Dosyaları okuma ve dosyalara yazma" sayfa 696.

`StorageVolumeInfo.getStorageVolumes()` yöntemi monte edilmiş depolama birimleri hakkında daha ayrıntılı bilgi sağlar (bkz. "Depolama birimleriyle çalışma" sayfa 695).

Varsayılan sistem uygulamasıyla dosya açma

Adobe AIR 2 ve üstü

AIR 2'de, işletim sisteminde dosyayı açmak için kayıtlı olan uygulamayı kullanarak bir dosyayı açabilirsiniz. Örneğin, AIR uygulaması bir DOC dosyasını onu açmak için kayıtlı olan uygulamayla açabilir. Dosyayı açmak için Dosya nesnesinin `openWithDefaultApplication()` yöntemini kullanın. Örneğin, aşağıdaki kodlar kullanıcının masaüstündeki test.doc isimli bir dosyayı DOC dosyalar için varsayılan uygulamayla açar:

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

Not: *Linux'ta bir dosyanın varsayılan uygulaması, dosya adı uzantısı ile değil, dosyanın MIME tipiyle belirlenir.*

Aşağıdaki kod kullanıcının bir mp3 dosyasına gitmesini ve mp3 dosyaları için varsayılan uygulamayla dosyayı açmasını sağlar:

```
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();
}
```

Uygulama dizinindeki dosyalar için `openWithDefaultApplication()` yöntemini kullanamazsınız.

AIR bazı dosyaları `openWithDefaultApplication()` yöntemiyle açmanızı engeller. Windows'ta AIR, EXE veya BAT gibi bazı dosya türlerini açmanızı engeller. Mac OS ve Linux'ta AIR, belirli bir uygulamada başlatılacak dosyaları açmanızı önler. (Bunlara Mac OS'taki Terminal ve AppletLauncher ve Linux'taki csh, bash veya ruby dahildir.) `openWithDefaultApplication()` yöntemini kullanarak bu dosyalardan birini açmaya çalışmak bir istisnayla sonuçlanır. Engellenen dosya türlerinin tam listesi için `File.openWithDefaultApplication()` yönteminin dil başvuru girişine bakın.

Not: *Bu sınırlama yerel bir yükleyici (genişletilmiş bir masaüstü uygulaması) kullanılarak yüklenmiş AIR uygulaması için geçerli değildir.*

Dosya sistemi bilgilerini alma

Adobe AIR 1.0 ve üstü

File sınıfı, dosya sistemi hakkında bazı yararlı bilgiler sağlayan aşağıdaki statik özellikleri içerir:

Özellik	Açıklama
`File.lineEnding`	Ana bilgisayar işletim sistemi tarafından kullanılan satır sonlandırıcı karakter dizisi. Bu, Mac OS ve Linux'ta satır besleme karakteridir. Bu, Windows'ta arkasından satır besleme karakteri gelen satır başı karakteridir.
`File.separator`	Ana bilgisayar işletim sisteminin yol bileşeni ayırıcı karakteri. Bu, Mac OS ve Linux'ta eğik çizgi (/) karakteridir. Bu, Windows'ta ters eğik çizgi (\) karakteridir.
`File.systemCharset`	Ana bilgisayar işletim sistemi tarafından dosyalar için kullanılan varsayılan kodlama. Bu, işletim sistemi tarafından kullanılan diline karşılık gelen karakter kümesiyle de ilgilidir.

`Capabilities` sınıfı da dosyalarla çalışırken yararlı olabilecek yararlı sistem bilgileri içerir:

Özellik	Açıklama
Capabilities.hasIME	Oynatıcının bir giriş yöntemi düzenleyicisi (IME) yüklü (`true`) bir sistem üzerinde mi, yüklü olmayan (`false`) bir sistem üzerinde mi çalıştığını belirtir.
Capabilities.language	Oynatıcının üzerinde çalıştığı sistemin dil kodunu belirtir.
Capabilities.os	Geçerli işletim sistemini belirtir.

Not: *Sistem karakteristiklerini belirlemek için `Capabilities.os` öğesini kullanırken dikkatli olun. Sistem karakteristiğini belirlemek için daha belirli bir özellik mevcutsa onu kullanın. Yoksa, tüm platformlarda düzgün bir şekilde çalışan kod yazma riskine sahip olursunuz. Örneğin, şu kodu göz önünde bulundurun:*

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

Bu kod Linux'ta sorunlara yol açar. Basitçe `File.separator` özelliğini kullanmak daha iyidir.

Dizinlerle çalışma

Adobe AIR 1.0 ve üstü

Çalışma zamanı size yerel dosya sistemindeki dizinlerle çalışmaya yönelik özellikler sağlar.

Dizinlere işaret eden File nesneleri oluşturma hakkında bilgi için bkz. “File nesnesini bir dizine işaret etme” sayfa 678.

Dizin oluşturma

Adobe AIR 1.0 ve üstü

`File.createDirectory()` yöntemi bir dizin oluşturmanızı sağlar. Örneğin aşağıdaki kod, kullanıcının ana dizininin alt dizini olarak AIR Test adlı bir dosya oluşturur.

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

Dizin varsa, `createDirectory()` yöntemi bir işlem gerçekleştirmez.

Ayrıca bazı modlarda FileStream nesnesi dosyaları açarken dizin oluşturur. Bir FileStream örneğini `FileMode.APPEND` veya `FileMode.WRITE` olarak ayarlı `FileStream()` yapıcı kümesinin `fileMode` parametresiyle başlattığınızda, eksik dizinler oluşturulur. Daha fazla bilgi için bkz. "Dosyaları okuma ve dosyalara yazma için iş akışı" sayfa 696.

Geçici bir dizin oluşturma

Adobe AIR 1.0 ve üstü

File sınıfı, aşağıdaki örnekte olduğu gibi Sistem için geçici bir dizin klasöründe bir dizin oluşturan `createTempDirectory()` yöntemini içerir.

```
var temp:File = File.createTempDirectory();
```

`createTempDirectory()` yöntemi otomatik olarak benzersiz bir geçici dizin oluşturur. (Yeni bir benzersiz konum belirlemekle uğraşmamış olursunuz.)

Geçici bir dizini, bir uygulama oturumu için kullanılan geçici dosyaları geçici olarak saklamak için kullanabilirsiniz. Sistem geçici dizininde yeni, benzersiz geçici dosyalar oluşturmak için bir `createTempFile()` yöntemi bulunduğunu unutmayın.

Tüm aygıtlarda otomatik olarak *silinmediğinden*, uygulamayı kapatmadan önce geçici dizini silmek isteyebilirsiniz.

Dizinleri numaralandırma

Adobe AIR 1.0 ve üstü

Bir dizindeki dosyalara ve alt klasörlere işaret eden bir File nesneleri dizisini almak için, File nesnesinin `getDirectoryListing()` veya `getDirectoryListingAsync()` yöntemini kullanabilirsiniz.

Örneğin aşağıdaki kod kullanıcının documents dizininin (alt dizinleri incelemeden) içeriğini listeler.

```
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);
}
```

Yöntemin senkronize olmayan sürümünü kullanırken, `directoryListing` olay nesnesi, dizinlerle ilgili File nesnelerinin dizisi olan bir `files` özelliğine sahiptir:

```
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);
    }
}
```

Dizinleri kopyalama ve taşıma

Adobe AIR 1.0 ve üstü

Bir dizini, bir dosyayı kopyalamak veya taşımak için kullanacağınız yöntemlerin aynılarını kullanarak kopyalayabilir veya taşıyabilirsiniz. Örneğin aşağıdaki kod bir dizini senkronize olarak kopyalar:

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

`copyTo()` yönteminin `overwrite` parametresi için true değerini belirttiğinizde, mevcut bir hedef dizindeki tüm dosyalar ve klasörler silinir ve yerine kaynak dizindeki dosyalar ve klasörler konur (hedef dosya kaynak dizinde olmasa bile).

`copyTo()` yönteminin `newLocation` parametresi olarak belirttiğiniz dizin, sonuçta oluşan dizinin yolunu belirtir; sonuçta oluşan dizini içerecek olan *üst* dizini *belirtmez*.

Ayrıntılar için bkz. "Dosyaları kopyalama ve taşıma" sayfa 693.

Dizin içeriğini silme

Adobe AIR 1.0 ve üstü

File sınıfı bir `deleteDirectory()` ve bir `deleteDirectoryAsync()` yöntemi içerir. İlki senkronize, ikincisi senkronize olmayan bir şekilde çalışan bu yöntemler dizinleri siler (bkz. "AIR dosyası temel bilgileri" sayfa 674). Her iki yöntem de `deleteDirectoryContents` parametresini içerir (Boolean değeri alır); bu parametre `true` olarak ayarlıyken (varsayılan değer `false` değeridir) yönteme yapılan çağrı boş olmayan dizinleri siler; aksi takdirde yalnızca boş dizinler silinir.

Örneğin aşağıdaki kod kullanıcının documents dizininin AIR Test alt dizinini senkronize olarak siler:

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

Aşağıdaki kod kullanıcının documents dizininin AIR Test alt dizinini senkronize olmayan bir şekilde siler:

```
 var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Ayrıca, bir dizini Sistem çöp kutusuna taşımak için kullanabileceğiniz `moveToTrash()` ve `moveToTrashAsync()` yöntemleri de bulunmaktadır. Ayrıntılar için bkz. "Bir dosyayı çöp kutusuna taşıma" sayfa 694.

Dosyalarla çalışma

Adobe AIR 1.0 ve üstü

AIR dosya API'sini kullanarak uygulamalarınıza temel dosya etkileşim özellikleri ekleyebilirsiniz. Örneğin dosyaları okuyabilir, yazabilir, kopyalayabilir ve silebilir, vb. işlemler gerçekleştirebilirsiniz. Uygulamalarınız yerel dosya sistemine erişebileceği için, önceden yapmadıysanız "AIR güvenliği" sayfa 1088 bölümüne bakın.

Not: *Bir dosya türünü AIR uygulamasıyla ilişkilendirebilirsiniz. (Böylece uygulama çift tıklatılarak açılabilir.) Ayrıntılar için bkz. "Dosya ilişkilendirmelerini yönetme" sayfa 897.*

Dosya bilgilerini alma

Adobe AIR 1.0 ve üstü

File sınıfı, bir File nesnesinin işaret ettiği dosya veya dizin hakkında bilgi sağlayan aşağıdaki özellikleri içerir:

Dosya özelliği	Açıklama
creationDate	Yerel diskteki dosyanın oluşturulma tarihi.
creator	Geçersiz—`extension` özelliğini kullanın. (Bu özellik dosyanın Macintosh oluşturucu türünü bildirir, bu yalnızca Mac OS X'ten önceki Mac OS sürümlerinde kullanılır.)
downloaded	(AIR 2 ve üstü) Başvurulan dosyanın veya dizinin indirilip indirilmediğini (İnternet'ten) belirtir. özellik yalnızca dosyaların indirildi olarak işaretlenebildiği işletim sistemlerinde anlamlıdır. • Windows XP servis paketi 2 ve sonrasında ve Windows Vista'da • Mac OS 10.5 ve sonrasında
exists	Başvurulan dosyanın veya dizinin var olup olmadığı.
extension	Son noktanın (".") arkasından gelen (noktayı içermeyen) adın bir parçası olan dosya uzantısı. Dosya adında nokta yoksa, uzantı `null` değerine sahiptir.
icon	Dosya için tanımlanan simgeleri içeren bir Icon nesnesi.
isDirectory	File nesnesi başvurusunun bir dizine yönelik olup olmadığı.
modificationDate	Yerel diskteki dosyanın veya dizinin son değiştirilme tarihi.
name	Yerel diskteki dosyanın veya dizinin adı (varsa dosya uzantısını da içeren).
nativePath	Ana bilgisayar işletim sistemi temsilcisindeki tam yol. Bkz. "File nesnelerinin Yolları" sayfa 675.
parent	File nesnesi tarafından temsil edilen klasörü veya dosyayı içeren klasör. File nesnesi dosya sisteminin kökündeki bir dosyaya veya dizine başvuruyorsa, bu özellik `null` değerindedir.

Dosya özelliği	Açıklama
size	Yerel diskteki dosyanın bayt olarak boyutu.
type	Geçersiz—`extension` özelliğini kullanın. (Macintosh'ta bu özellik dört karakterli dosya türüdür, bu tür yalnızca Mac OS X'ten önceki Mac OS sürümlerinde kullanılır.)
url	Dosyanın veya dizinin URL'si. Bkz. "File nesnelerinin Yolları" sayfa 675.

Bu özelliklerle ilgili ayrıntılı bilgi için, Adobe Flash Platform için ActionScript 3.0 Başvurusu bölümündeki File sınıfı girişine bakın.

Dosyaları kopyalama ve taşıma

Adobe AIR 1.0 ve üstü

File sınıfı dosyaları veya dizinleri kopyalamak için iki yöntem içerir: `copyTo()` ve `copyToAsync()`. File sınıfı dosyaları veya dizinleri taşımak için iki yöntem içerir: `moveTo()` ve `moveToAsync()`. `copyTo()` ve `moveTo()` yöntemleri senkronize çalışır; `copyToAsync()` ve `moveToAsync()` yöntemleri senkronize olmayan bir şekilde çalışır (bkz. "AIR dosyası temel bilgileri" sayfa 674).

Bir dosyayı kopyalamak veya taşımak için iki File nesnesi ayarlarsınız. Biri kopyalanacak veya taşınacak olan dosyaya işaret eder, bu copy veya move yöntemini çağıran nesnedir; diğeri hedef (sonuç) yoluna işaret eder.

Aşağıdaki örnek bir test.txt dosyasını kullanıcı documents dizininin AIR Test alt dizininden aynı dizindeki copy.txt adlı dosyaya taşır:

```
var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

Bu örnekte `copyTo()` yönteminin `overwrite` parametresinin (ikinci parametre) değeri `true` olarak ayarlıdır. `overwrite` değerini `true` olarak ayarlayarak varolan bir hedef dosyanın üzerine yazılır. Bu parametre isteğe bağlıdır. Bu değeri `false` (varsayılan değer) olarak ayarlarsanız, hedef dosya mevcutsa işlem bir IOErrorEvent olayı gönderir (ve dosya kopyalanmaz).

copy ve move yöntemlerinin "Senkronize olmayan" sürümleri senkronize olmayan bir şekilde çalışır. Aşağıdaki kodda olduğu gibi, görev veya hata durumlarının tamamlanmasını izlemek için `addEventListener()` yöntemini kullanın:

```
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.");
}
```

File sınıfı ayrıca `File.moveToTrash()` ve `File.moveToTrashAsync()` yöntemlerini de içerir, bu yöntemler bir dosyayı veya dizini sistem çöp kutusuna taşır.

Bir dosyayı silme

Adobe AIR 1.0 ve üstü

File sınıfı bir `deleteFile()` ve bir `deleteFileAsync()` yöntemi içerir. İlki senkronize, ikincisi senkronize olmayan bir şekilde çalışan bu yöntemler dosyaları siler (bkz. "AIR dosyası temel bilgileri" sayfa 674).

Örneğin aşağıdaki kod kullanıcının documents dizinindeki test.txt dosyasını senkronize olarak siler:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

Aşağıdaki kod kullanıcının documents dizininin test.txt dosyasını senkronize olmayan bir şekilde siler:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

Ayrıca, bir dosyayı veya dizini Sistem çöp kutusuna taşımak için kullanabileceğiniz `moveToTrash()` ve `moveToTrashAsync()` yöntemleri de bulunmaktadır. Ayrıntılar için bkz. "Bir dosyayı çöp kutusuna taşıma" sayfa 694.

Bir dosyayı çöp kutusuna taşıma

Adobe AIR 1.0 ve üstü

File sınıfı bir `moveToTrash()` ve bir `moveToTrashAsync()` yöntemi içerir. Bu yöntemler bir dosyayı veya dizini Sistem çöp kutusuna gönderir, ilki senkronize, ikincisi senkronize olmayan bir şekilde çalışır (bkz. "AIR dosyası temel bilgileri" sayfa 674).

Örneğin aşağıdaki kod kullanıcının documents dizinindeki test.txt dosyasını senkronize olarak Sistem çöp kutusuna gönderir:

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

Not: *Kurtarılabilir çöp klasörü konseptini desteklemeyen işletim sistemlerinde dosyalar hemen kaldırılır.*

Geçici bir dosya oluşturma

Adobe AIR 1.0 ve üstü

File sınıfı, aşağıdaki örnekte olduğu gibi Sistem için geçici bir dizin klasöründe bir dosya oluşturan `createTempFile()` yöntemini içerir.

```
var temp:File = File.createTempFile();
```

`createTempFile()` yöntemi otomatik olarak benzersiz bir geçici dosya oluşturur. (Yeni bir benzersiz konum belirlemekle uğraşmamış olursunuz.)

Geçici bir dosyayı, bir uygulama oturumu için kullanılan bilgileri geçici olarak saklamak için kullanabilirsiniz. Ayrıca Sistem geçici dizininde benzersiz geçici bir dizin oluşturmak için `createTempDirectory()` yönteminin bulunduğunu unutmayın.

Tüm aygıtlarda otomatik olarak *silinmediğinden*, uygulamayı kapatmadan önce geçici dosyayı silmek isteyebilirsiniz.

Depolama birimleriyle çalışma

Adobe AIR 2 ve üstü

AIR 2 uygulamasında yığın depolama birimlerinin takılıp takılmadığını algılayabilirsiniz. StorageVolumeInfo sınıfı, tek bir `storageVolumeInfo` nesnesi örneği tanımlar. Depolama birimi takıldığında `StorageVolumeInfo.storageVolumeInfo` nesnesi bir `storageVolumeMount` olayı gönderir. Bir birim çıkarıldığında, `storageVolumeUnmount` olayı gönderir. Bu olaylar StorageVolumeChangeEvent sınıfında tanımlanmıştır.

Not: *Modern Linux dağıtımlarında, StoreVolumeInfo nesnesi `storageVolumeMount`ve `storageVolumeUnmount` olaylarını yalnızca belirli konumlarda oluşturulmuş fiziksel aygıtlar ve ağ sürücüleri için gönderir.*

StorageVolumaChangeEvent sınıfının `storageVolume` özelliği bir Storage Volume nesnesidir. StorageVolume sınıfı depolama biriminin temel özelliklerini tanımlar:

- `sürücü`—Windows'ta birim sürücüsü harfi (diğer işletim sistemlerinde `null` değeridir)
- `fileSystemType`— Depolama birimindeki dosya sisteminin türü ("FAT", "NTFS", "HFS" veya "UFS" gibi)
- `isRemoveable`— Depolama biriminin çıkartılabilir olup (`true`) olmadığı (`false`)
- `isWritable`—Bir veri kütlesinin yazılabilir (`true`) olup olmadığı (`false`)
- `name`— Birimin adı
- `rootDirectory`— Birimin kök dizinine karşılık gelen File nesnesi

StorageVolumeChangeEvent sınıfında bir `rootDirectory` özelliği de bulunur. `rootDirectory` özelliği, takılmış veya sökülmüş depolama kapasitesinin kök diziniyle ilgili Dosya nesnesidir.

StorageVolumeChangeEvent nesnesinin `storageVolume` özelliği, takılı olmayan birim için tanımsızdır (`null`). Ancak olayın `rootDirectory` özelliğine erişebilirsiniz.

Aşağıdaki kod, takıldığında, depolama biriminin dosya yolunu çıktı alır.

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_MOUNT, onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

Aşağıdaki kod, çıkartıldığında, depolama biriminin dosya yolunu çıktı alır.

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME
_UNMOUNT, onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

`StorageVolumeInfo.storageVolumeInfo` nesnesi `getStorageVolumes()` yöntemi içerir. Bu yöntem StorageVolume nesnelerinin monte edilmiş depolama birimlerine karşılık gelen bir vektörünü verir. Aşağıdaki kod takılmış tüm depolama birimlerinin isimlerinin ve kök dizinlerinin nasıl listeleneceğini gösterir:

```
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);
}
```

Not: *Modern Linux dağıtımlarında, `getStorageVolumes()` yöntemi nesneleri ilgili fiziksel aygıtlara ve belli konumlara takılmış ağ sürücülerine geri gönderir.*

`File.getRootDirectories()` yöntemi kök dizinleri listeler (bkz. "Dosya sistemi köküne işaret etme" sayfa 680. Ancak StorageVolume nesneleri (`StorageVolumeInfo.getStorageVolumes()` yöntemiyle numaralandırılmış) depolama birimleri hakkında daha fazla bilgi sağlar.

`spaceAvailable` özelliği, Storage Volume nesnesinin `rootDirectory` özelliğinin bir özelliğidir, bu özelliği depolama biriminde ne kadar kullanılabilir alan olduğunu öğrenmek için kullanabilirsiniz. (Bkz. "Bir birimdeki kullanılabilir alanı belirleme" sayfa 687.)

Daha fazla Yardım konusu

StorageVolume

StorageVolumeInfo

Dosyaları okuma ve dosyalara yazma

Adobe AIR 1.0 ve üstü

FileStream sınıfı AIR uygulamalarının dosya sistemini okumasına ve dosya sistemine yazmasına izin verir.

Dosyaları okuma ve dosyalara yazma için iş akışı

Adobe AIR 1.0 ve üstü

Dosyaları okuma ve dosyalara yazma için iş akışı şu şekildedir.

Yola işaret eden bir File nesnesini başlatın.
Dosya nesnesi, üzerinde çalışmak istediğiniz dosyanın yolunu gösterir.

```
 var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

Bu örnek File nesnesini başlatmak için File nesnesinin `File.documentsDirectory` özelliğini ve `resolvePath()` yöntemini kullanır. Ancak bir File nesnesini bir dosyaya işaret etmenin başka birçok yolu vardır. Daha fazla bilgi için bkz. "File nesnesini bir dosyaya işaret etme" sayfa 682.

Bir FileStream nesnesini başlatın.

FileStream nesnesinin open() veya openAsync() yöntemini çağırın.
Çağırdığınız yöntem, dosyayı senkronize işlemler için mi, senkronize olmayan işlemler için mi açmak istediğinize bağlı olarak değişir. File nesnesini open yönteminin `file` parametresi olarak kullanın. `fileMode` parametresi için, FileMode sınıfından dosyayı kullanma şeklinizi belirten bir sabit belirtin.

Örneğin aşağıdaki kod bir dosya oluşturmak ve varolan verilerin üzerine yazmak için kullanılan bir FileStream nesnesi başlatır:

```
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

Daha fazla bilgi için bkz. "Bir FileStream nesnesi başlatma ve dosyaları açma ve kapama" sayfa 698 ve "FileStream açma modları" sayfa 698.

Dosyayı senkronize olmayan bir şekilde açtıysanız (openAsync() yöntemini kullanarak), FileStream nesnesi için olay dinleyicileri oluşturun ve ekleyin.
Bu olay dinleme yöntemleri, çeşitli durumlardaki FileStream nesnesi tarafından gönderilen olaylara cevap verir. Bu durumlar arasında, I/O hatalarıyla karşılaştırıldığında veya yazılacak toplam veri miktarının yazıldığında verilerin dosyadan okunduğu durumlar bulunur.

Ayrıntılar için bkz. "Senkronize olmayan bir şekilde açılan FileStream nesnesi tarafından oluşturulan olaylar ve senkronize olmayan programlama" sayfa 702.

Verileri okuma ve yazma için kodu gerektiği şekilde dahil etme.
FileStream sınıfının okuma ve yazmayla ilgili birçok yöntemi vardır. (Her biri "read" veya "write" ifadesiyle başlar.) Veri okumak veya yazmak için kullanmayı seçtiğiniz yöntem, hedef dosyadaki verilerin biçimine bağlıdır.

Örneğin hedef dosyadaki veriler UTF kodlu metinse, `readUTFBytes()` ve `writeUTFBytes()` yöntemlerini kullanabilirsiniz. Verileri bayt dizileri olarak işlemek istiyorsanız, `readByte()`, `readBytes()`, `writeByte()` ve `writeBytes()` yöntemlerini kullanabilirsiniz. Ayrıntılar için bkz. "Veri biçimleri ve kullanılacak okuma ve yazma yöntemlerini seçme" sayfa 703.

Dosyayı senkronize olmayan bir şekilde açtıysanız, bir read yöntemini çağırmadan önce yeterli verinin kullanılabilir olduğundan emin olun. Ayrıntılar için bkz. "FileStream nesnesinin okuma arabelleği ve bytesAvailable özelliği" sayfa 701.

Bir dosyaya yazmadan önce kullanılabilir disk alanını kontrol etmek istiyorsanız, File nesnesinin spaceAvailable özelliğini kontrol edebilirsiniz. Daha fazla bilgi için bkz. "Bir birimdeki kullanılabilir alanı belirleme" sayfa 687.

Dosyayla çalışmanız bittiğinde FileStream nesnesinin close() yöntemini çağırın.
close() yönteminin çağırılması dosyanın diğer uygulamalarda kullanılabilmesini sağlar.

Ayrıntılar için bkz. "Bir FileStream nesnesi başlatma ve dosyaları açma ve kapama" sayfa 698.

Dosyaları okumak ve yazmak için FileStream sınıfı kullanan örnek bir uygulama görmek için, Adobe AIR Geliştirici Merkezi'ndeki şu makalelere bakın:

- Metin dosyası düzenleyicisi oluşturma
- Metin dosyası düzenleyicisi oluşturma
- XML tercihler dosyasından okuma ve yazma

FileStream nesneleriyle çalışma

Adobe AIR 1.0 ve üstü

FileStream sınıfı dosyaları açmak, okumak ve yazmak için yöntemler tanımlar.

FileStream açma modları

Adobe AIR 1.0 ve üstü

FileStream nesnesinin `open()` ve `openAsync()` yöntemlerinin her biri bir `fileMode` parametresi içerir, bu parametre bir dosya akışı için, aşağıdakiler de dahil olmak üzere bazı özellikler tanımlar:

- Dosyadan okuma özelliği
- Dosyaya yazma özelliği
- Verilerin her zaman dosyanın sonuna eklenip eklenmeyeceği (yazarken)
- Dosya yoksa ne yapılacağı (üst dizinleri de olmadığında)

Aşağıda çeşitli dosya modları verilmiştir (`open()` ve `openAsync()` yöntemlerinin `fileMode` parametresi olarak belirtebileceğiniz):

Dosya modu	Açıklama
FileMode.READ	Dosyanın yalnızca okuma için açık olduğunu belirtir.
FileMode.WRITE	Dosyanın yalnızca yazma için açık olduğunu belirtir. Dosya yoksa, FileStream nesnesi açıldığında oluşturulur. Dosya varsa, varolan tüm veriler silinir.
FileMode.APPEND	Dosyanın yalnızca ekleme için açık olduğunu belirtir. Dosya yoksa oluşturulur. Dosya varsa, varolan verilerin üzerine yazılmaz ve tüm yazma işlemleri dosyanın sonundan başlar.
FileMode.UPDATE	Dosyanın yalnızca okuma ve yazma için açık olduğunu belirtir. Dosya yoksa oluşturulur. Dosyaya rastgele okuma/yazma erişimi için bu modu belirtin. Dosyadaki bütün konumlardan okuyabilirsiniz. Dosyaya yazarken yalnızca yazılmış baytlar, mevcut baytların üzerine yazılır (diğer tüm baytlar değişmez).

Bir FileStream nesnesi başlatma ve dosyaları açma ve kapama

Adobe AIR 1.0 ve üstü

Bir FileStream nesnesini açtığınızda, nesneyi bir dosyaya veri okumak ve yazmak için kullanılabilir hale getirirsiniz. FileStream nesnesini, FileStream nesnesinin `open()` veya `openAsync()` yöntemine File nesnesini ileterek açarsınız:

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

`fileMode` parametresi (`open()` ve `openAsync()` yöntemlerinin ikinci parametresi), dosyanın açılacağı modu belirtir: okuma, yazma, ekleme veya güncelleme için. Ayrıntılar için önceki bölümlerden "FileStream açma modları" sayfa 698 bölümüne bakın.

`openAsync()` yöntemini dosyayı senkronize olmayan dosya işlemleri için açmak üzere kullanırsanız, senkronize olmayan olayları işlemek için olay dinleyicileri ayarlayın:

```
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 {
    // ...
}
```

Dosya, `open()` veya `openAsync()` yöntemlerinden hangisini kullandığınıza bağlı olarak senkronize veya senkronize olmayan işlemler için açılır. Ayrıntılar için bkz. "AIR dosyası temel bilgileri" sayfa 674.

`fileMode` parametresini FileStream nesnesinin open yönteminde `FileMode.READ` veya `FileMode.UPDATE` olarak ayarlarsanız, siz FileStream nesnesini açar açmaz veriler okuma arabelleğine okunur. Ayrıntılar için bkz. "FileStream nesnesinin okuma arabelleği ve bytesAvailable özelliği" sayfa 701.

İlişkilendirilen dosyayı kapatmak için FileStream nesnesinin `close()` yöntemini çağırabilirsiniz, böylece dosyayı diğer uygulamalar tarafından kullanılabilir hale getirirsiniz.

FileStream nesnesinin position özelliği

Adobe AIR 1.0 ve üstü

FileStream nesnesinin `position` özelliği, sonraki read veya write yönteminde verilerin nereye okunduğunu veya yazıldığını belirler.

Bir okuma veya yazma işleminden önce, `position` özelliğini dosyada geçerli bir konuma ayarlayın.

Örneğin aşağıdaki kod `"hello"` dizesini (UTF kodlamada) dosyada konum 8'de yazar:

```
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");
```

Bir FileStream nesnesini ilk açtığınızda, `position` özelliği 0'a ayarlıdır.

Bir okuma işleminden önce `position` değeri en az 0 ve dosyadaki bayt sayısından (dosyadaki varolan konumlar) az olmalıdır.

`position` özelliğinin değeri yalnızca aşağıdaki durumlarda değiştirilir:

- `position` özelliğini açık olarak ayarladığınızda.
- Bir read yöntemini çağırdığınızda.
- Bir write yöntemini çağırdığınızda.

FileStream nesnesinin read veya write yöntemini çağırdığınızda, `position` özelliği hemen okuduğunuz veya yazdığınız bayt sayısı kadar artırılır. Kullandığınız read yöntemine bağlı olarak, `position` özelliği okumak için belirttiğiniz bayt sayısı kadar veya kullanılabilir bayt sayısı kadar artırılır. Bir read veya write yöntemini sonradan çağırdığınızda, yeni konumdan başlayarak okur veya yazar.

```
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
```

Ancak bir istisna söz konusudur: Ekleme modunda açılmış bir FileStream için, write yöntemine yapılan bir çağrıdan sonra `position` özelliği değişmez. (Ekleme modunda veriler, `position` özelliğinin değerinden bağımsız olarak her zaman dosyanın sonuna yazılır.)

Senkronize olmayan işlemler için açılan bir dosyada, kodun bir sonraki satırı çalıştırılmadan yazma işlemi tamamlanmaz. Ancak birden çok senkronize olmayan yöntemi sıralı olarak çağırabilirsiniz ve çalışma zamanı onları sırayla çalıştırır:

```
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.");
}
```

Bu kod için izleme çıktısı şöyledir:

```
started.
finished.
```

`position` değerini bir read veya write yöntemini çağırdıktan hemen sonra (veya istediğiniz zaman) *belirtebilirsiniz*, böylece bir sonraki okuma veya yazma işlemi o konumdan başlar. Örneğin, aşağıdaki kod `position` özelliğini `writeBytes()` işlemine bir çağrı yapıldıktan hemen sonra ayarlar ve yazma işlemi tamamlandıktan sonra bile `position` o değere ayarlanır (300):

```
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
```

FileStream nesnesinin okuma arabelleği ve bytesAvailable özelliği

Adobe AIR 1.0 ve üstü

Okuma özelliklerine sahip bir FileStream nesnesi (`open()` veya `openAsync()` yönteminin `fileMode` parametresinin `READ` veya `UPDATE` olarak ayarlı olduğu) açıldığında çalışma zamanı, verileri dahili bir arabellekte saklar. FileStream nesnesi, siz dosyayı açar açmaz (FileStream nesnesinin `open()` veya `openAsync()` yöntemini çağırarak) verileri arabelleğe okumaya başlar.

Aşağıdaki kodda gösterildiği gibi (dosyanın en az 100 bayttan oluştuğunu varsayar), senkronize işlemler için açılan bir dosyada (`open()` yöntemi kullanılarak), `position` işaretçisini herhangi bir geçerli konuma ayarlayabilirsiniz (dosyanın sınırları içinde) ve herhangi bir miktardaki veriyi okumaya başlayabilirsiniz (dosyanın sınırları içinde):

```
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);
```

Read yöntemleri, bir dosyanın senkronize veya senkronize olmayan işlemler için açıldığına bakılmaksızın, okuma işlemini her zaman `bytesAvalable` özelliği tarafından temsil edilen "kullanılabilir" baytlardan gerçekleştirir. Senkronize olarak okurken, dosyadaki tüm baytlar her zaman kullanılabilirdir. Senkronize olmayan bir şekilde okuma yaparken, `progress` olayları tarafından bildirilen senkronize olmayan arabellek dolumları dizisinde, baytlar, `position` özelliği tarafından belirtilen konumdan başlayarak kullanılabilir olur.

Senkronize işlemler için açılan dosyalarda, `bytesAvailable` özelliği her zaman `position` özelliğinden dosyanın sonuna kadar olan bayt sayısını temsil edecek şekilde ayarlıdır. (Dosyadaki tüm baytlar her zaman okunabilir.)

Senkronize olmayan işlemler için açılan dosyalarda, okuma arabelleğinin bir read yöntemini çağırmadan önce yeterli veri tüketmiş olduğundan emin olmanız gerekir. Senkronize olmayan bir şekilde açılmış bir dosyada, okuma işlemi ilerledikçe, dosyadaki veriler okuma işlemi başlatıldığında belirtilen `position` öğesinden başlayarak arabelleğe eklenir ve okunan her baytla `bytesAvailable` özelliği artar. `bytesAvailable` özelliği, `position` özelliği tarafından belirtilen konumdaki baytla başlayarak arabelleğin sonuna kadar kullanılabilir olan baytların sayısını gösterir. FileStream nesnesi düzenli aralıklarla bir `progress` olayı gönderir.

Senkronize olmayan bir şekilde açılan dosyada, veriler okuma arabelleğinde kullanılabilir oldukça, FileStream nesnesi düzenli aralıklarla `progress` olayını gönderir. Örneğin aşağıdaki kod, arabelleğe okundukça verileri `bytes` adlı bir ByteArray nesnesine okur:

```
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);
}
```

Senkronize olmayan bir şekilde açılan dosyada, yalnızca okuma arabelleğindeki veriler okunabilir. Ayrıca siz verileri okudukça, veriler okuma arabelleğinden kaldırılır. Okuma işlemleri için, okuma işlemini çağırmadan önce verilerin okuma arabelleğinde bulunduğundan emin olmanız gerekir. Örneğin aşağıdaki kod, dosyada konum 4000'den başlayarak 8000 bayt veri okur:

```
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");
    }
}
```

FileStream nesnesi bir yazma işlemi boyunca verileri okuma arabelleğine okumaz. Bir yazma işlemi tamamlandığında (yazma arabelleğindeki tüm veriler dosyaya yazılır), FileStream nesnesi yeni bir okuma arabelleğine başlar (ilişkilendirilen FileStream nesnesinin okuma özellikleriyle açıldığı varsayılarak) ve `position` özelliği tarafından belirtilen konumdan başlayarak okuma arabelleğine veri okumaya başlar. `position` özelliği yazılan son baytın konumu veya kullanıcı yazma işleminden sonra `position` nesnesi için farklı bir değer belirtirse farklı bir konum olabilir.

Senkronize olmayan bir şekilde açılan FileStream nesnesi tarafından oluşturulan olaylar ve senkronize olmayan programlama

Adobe AIR 1.0 ve üstü

Bir dosya senkronize olmayan bir şekilde açıldığında (`openAsync()` yöntemi kullanılarak), dosyaları okuma ve yazma işlemi senkronize olmayan bir şekilde gerçekleştirilir. Veriler okuma arabelleğine okunduğu ve çıktı verileri yazıldığı için, diğer ActionScript kodu çalıştırılabilir.

Bu, FileStream nesnesi tarafından oluşturulan senkronize olmayan bir şekilde açılan olaylar için kaydolmanız gerektiği anlamına gelir.

Aşağıdaki kodda olduğu gibi, `progress` olayı için kaydolduğunuzda yeni veriler okuma işlemi için kullanılabilir olduğunda size bilgi verilebilir:

```
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");
}
```

Aşağıdaki kodda olduğu gibi `complete` olayı için kaydolarak tüm verileri okuyabilirsiniz:

```
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");
}
```

Senkronize olmayan okuma işleminin gerçekleşmesi için giriş verilerinin arabelleğe alınma şekline oldukça benzer bir şekilde, senkronize olmayan bir akışa yazdığınız veriler arabelleğe alınır ve dosyaya senkronize olmayan bir şekilde yazılır. Veriler bir dosyaya yazıldıkça, FileStream nesnesi düzenli aralıklarla bir `OutputProgressEvent` nesnesi gönderir. `OutputProgressEvent` nesnesi, yazılmak için kalan bayt sayısına ayarlı bir `bytesPending` özelliği içerir. Belki de bir ilerleme iletişim kutusu görüntülemek için bu arabellek gerçekten bir dosyaya yazıldığı için, `outputProgress` olayı için bilgi almak üzere kaydolabilirsiniz. Ancak genelde bunu yapmanıza gerek yoktur. Özellikle, yazılmamış baytları dikkate almadan `close()` yöntemini çağırabilirsiniz. FileStream nesnesi veri yazmaya devam eder ve son bayt dosyaya yazıldıktan ve alttaki dosya kapatıldıktan sonra `close` olayı iletilir.

Veri biçimleri ve kullanılacak okuma ve yazma yöntemlerini seçme

Adobe AIR 1.0 ve üstü

Her dosya diskteki bir bayt kümesidir. ActionScript'te bir dosyadan gelen veriler her zaman bir ByteArray olarak temsil edilebilir. Örneğin aşağıdaki kod verileri bir dosyadan `bytes` adlı bir ByteArray nesnesine okur:

```
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);
}
```

Benzer şekilde aşağıdaki kod verileri `bytes` adlı bir ByteArray nesnesinden bir dosyaya yazar:

```
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);
```

Ancak genellikle verileri bir ActionScript ByteArray nesnesinde saklamak istemezsiniz. Genellikle veri dosyası belirtilen dosya biçimindedir.

Örneğin dosyadaki veriler metin dosyası biçiminde olabilir ve bu tür verileri bir String nesnesinde temsil etmek isteyebilirsiniz.

Bu nedenle FileStream sınıfı, ByteArray nesnelerinin dışındaki türlere ve türlerden yapılacak veri okuma ve yazma işlemleri için read ve write yöntemleri içerir. Örneğin `readMultiByte()` yöntemi sayesinde, aşağıdaki kodda olduğu gibi verileri bir dosyadan okuyup bir dizede saklayabilirsiniz:

```
 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");
}
```

`readMultiByte()` yönteminin ikinci parametresi, ActionScript'in verileri yorumlamak için kullandığı metin biçimini belirtir (örnekte "iso-8859-1"). Adobe AIR yaygın kullanılan karakter seti kodlamalarını destekler (bkz. Desteklenen karakter setleri).

FileStream sınıfı ayrıca `readUTFBytes()` yöntemini de içerir, bu yöntem UTF-8 karakter kümesini kullanarak okuma arabelleğinden verileri bir dizeye okur. UTF-8 karakter kümesindeki karakterler çeşitli uzunluklarda olduğu için, `readUTFBytes()` öğesini `progress` olayına yanıt veren bir yöntemde kullanmayın, okuma arabelleğinin sonundaki veriler eksik bir karakteri temsil edebilir. (Bu ayrıca `readMultiByte()` yöntemini çeşitli uzunluklardaki karakter kodlamasıyla kullanırken de geçerlidir.) Bu nedenle FileStream nesnesi `complete` olayını gönderdiğinde veri kümesinin tamamını okuyun.

Ayrıca String nesneleriyle ve metin dosyalarıyla çalışmak için, `writeMultiByte()` ve `writeUTFBytes()` gibi benzer write yöntemleri de vardır.

`readUTF()` ve `writeUTF()` yöntemleri de (`readUTFBytes()` ve `writeUTFBytes()` yöntemleriyle karıştırılmamalıdır) metin verilerini bir dosyaya okur ve yazar, ancak onlar metin verilerinden önce, metin verilerinin uzunluğunu belirten verilerin geldiğini varsayar, bu da standart metin dosyaları için yaygın bir uygulama değildir.

Bazı UTF kodlu metin dosyaları "UTF-BOM" (bayt sırası işareti) karakteriyle başlar, bu karakter kodlama biçiminin yanı sıra endian olduğunu belirtir (UTF-16 veya UTF-32 gibi).

Bir dosyaya okuma ve yazma örneği için bkz. “Örnek: Bir XML dosyasını XML nesnesine okuma” sayfa 705.

`readObject()` ve `writeObject()`, karmaşık ActionScript nesneleri için veri saklamak ve almak için pratik yollardır. Veriler AMF'de kodlanmıştır (ActionScript Message Format). Adobe AIR, Flash Player, Flash Media Server, ve Flex Data Hizmetleri bu biçimde veriyle çalışmaya yarayan API'ler içerir.

Başka read ve write yöntemleri (`readDouble()` ve `writeDouble()` gibi) vardır. Ancak bunları kullanırsanız, dosya biçiminin bu yöntemler tarafından tanımlanan verilerin biçimleriyle eşleştiğinden emin olun.

Dosya biçimleri genellikle basit metin biçimlerinden daha karmaşıktır. Örneğin bir MP3 dosyası, yalnızca belirli MP3 dosyalarına özgü olan açma ve kod çözme algoritmalarıyla yorumlanabilen sıkıştırılmış veriler içerir. MP3 dosyaları da dosya hakkında meta etiket bilgilerine (bir şarkının başlığı ve sanatçı gibi) sahip ID3 etiketleri içerebilir. ID3 biçiminin birden çok sürümü vardır, ancak bunlardan en basiti (ID3 sürüm 1) “Örnek: Rastgele erişimle veri okuma ve yazma” sayfa 707 bölümünde anlatılmaktadır.

Diğer dosya biçimleri (görüntüler, veri tabanları, uygulama belgeleri, vb.) farklı yapılara sahiptir ve ActionScript'te onların verileriyle çalışmak için, verilerin nasıl yapılandırıldığını anlamanız gerekir.

load() ve save() yöntemlerini kullanma

Flash Player 10 ve üstü, Adobe AIR 1.5 ve üstü

Flash Player 10, FileReference sınıfına `load()` ve `save()` yöntemlerini eklemiştir. Bu yöntemler AIR 1.5'te de bulunur ve File sınıfı yöntemleri FileReference sınıfından miras alır. Bu yöntemler kullanıcıların Flash Player'da dosya verilerini yüklemeleri ve kaydetmeleri için güvenli bir yol sağlamak üzere tasarlanmıştır. Ancak, AIR uygulamaları bu yöntemleri dosyaları senkronize olmayan şekilde yükleme ve kaydetmenin kolay bir yolu olarak da kullanabilir.

Örneğin, aşağıdaki kod bir dizeyi metin dosyasına kaydeder.

```
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.");
}
```

`save()` yönteminin `data` parametresi bir String, XML veya ByteArray değeri alabilir. Argüman bir String veya XML değeri olduğunda, yöntem dosyayı UTF-8 kodlu bir metin dosyası olarak kaydeder.

Bu kod örneği yürütüldüğünde, uygulama kullanıcının kaydedilen dosya hedefini seçtiği bir iletişim kutusu görüntüler.

Aşağıdaki kod, UTF-8 kodlu metin dosyasından bir dize yükler:

```
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);
}
```

FileStream sınıfı, `load()` ve `save()` yöntemlerinin sağladığından daha fazla işlev sağlar:

- FileStream sınıfını kullanarak, verileri senkronize ve senkronize olmayan biçimde okuyabilir ve yazabilirsiniz.
- FileStream sınıfını kullanarak, bir dosyaya artımlı olarak yazabilirsiniz.
- FileStream sınıfının kullanılması rastgele erişim için dosya açabilmenizi sağlar (dosyanın herhangi bir bölümünden okuyarak veya herhangi bir bölümüne yazarak).
- FileStream sınıfı, `open()` veya `openAsync()` yönteminin `fileMode` parametresini ayarlayarak dosyaya olan erişiminizin türünü belirlemenize olanak verir.
- FileStream sınıfı, kullanıcıya Açma veya Kaydetme iletişim kutusunu sunmadan verileri kaydedebilmenizi sağlar.
- FileStream sınıfıyla verileri okuma sırasında bayt dizileri dışındaki türleri doğrudan kullanabilirsiniz.

Örnek: Bir XML dosyasını XML nesnesine okuma

Adobe AIR 1.0 ve üstü

Aşağıdaki örnekler, XML verileri içeren bir metin dosyasına okuma ve yazma işlemlerinin nasıl yapılacağını gösterir.

Dosyadan okumak için, File ve FileStream nesnelerini başlatın, FileStream nesnesinin `readUTFBytes()` yöntemini çağırın ve dizeyi bir XML nesnesine dönüştürün:

```
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();
```

Benzer şekilde dosyaya veri yazmak da, uygun File ve FileStream nesnelerini ayarlayıp FileStream nesnesinin write yöntemini çağırmak kadar kolaydır. XML verilerinin dize sürümünü write yöntemine aşağıdaki kodda olduğu gibi iletin:

```
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();
```

Bu örnekler `readUTFBytes()` ve `writeUTFBytes()` yöntemlerini kullanır, çünkü dosyaların UTF-8 biçiminde olduğunu varsayar. Değilse, farklı bir yöntem kullanmanız gerekebilir (bkz. "Veri biçimleri ve kullanılacak okuma ve yazma yöntemlerini seçme" sayfa 703).

Önceki örnekler senkronize işlem için açılmış FileStream nesnelerini kullanır. Senkronize olmayan işlemler (olaylara yanıt vermek için olay dinleyicisi işlevlerine dayanan) için de dosyalar açabilirsiniz. Örneğin aşağıdaki kod, bir XML dosyasının senkronize olmayan şekilde nasıl okunacağını gösterir:

```
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();
}
```

`processXMLData()` yöntemi, dosyanın tamamı okuma arabelleğine okunduğu zaman çağrılır (FileStream nesnesi `complete` olayını gönderdiğinde). Okunan verilerin dize sürümünü alması için `readUTFBytes()` yöntemini çağırır ve o dizeye dayalı olarak `prefsXML` adlı bir XML nesnesi oluşturur.

Bu özellikleri gösteren örnek bir uygulama için bkz. XML tercihler dosyasından okuma ve yazma.

Örnek: Rastgele erişimle veri okuma ve yazma

Adobe AIR 1.0 ve üstü

MP3 dosyaları, dosyanın başında veya sonunda yer alan, kaydı tanımlayan meta verileri içeren bölümler olan ID3 etiketlerine sahip olabilir. ID3 etiket biçiminin farklı sürümleri vardır. Bu örnek, en basit ID3 biçimini içeren (ID3 sürüm 1.0) MP3 dosyasından, dosyadaki rastgele konumlara okuma ve yazma işleminin gerçekleştirildiği anlamına gelen "dosya verilerine rastgele erişim" kullanılarak nasıl veri okunacağını ve yazılacağını gösterir.

ID3 sürüm 1 etiketini içeren bir MP3 dosyası, ID3 verilerini dosyanın sonunda, son 128 baytta içerir.

Bir dosyaya rastgele okuma/yazma erişimi için erişirken, `open()` veya `openAsync()` yönteminin `fileMode` parametresi olarak `FileMode.UPDATE` öğesini belirtmek gerekir:

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

Böylece dosyaya hem okur hem de yazarsınız.

Dosyayı açtıktan sonra, `position` işaretçisini, dosyanın sonundan 128 bayt önce konumuna getirebilirsiniz:

```
fileStr.position = file.size - 128;
```

Bu kod `position` özelliğini dosyada bu konuma ayarlar, çünkü ID3 v1.0 biçimi, ID3 etiket verilerinin dosyanın son 128 baytında saklandığını belirtir. Bu belirtim ayrıca şunu da gösterir:

- Etiketin ilk 3 baytı `"TAG"` dizesini içerir.
- Sonraki 30 karakter, bir dize olarak MP3 parçasının başlığını içerir.
- Sonraki 30 karakter, bir dize olarak sanatçının adını içerir.
- Sonraki 30 karakter, bir dize olarak albümün adını içerir.
- Sonraki 4 karakter, bir dize olarak yılı içerir.
- Sonraki 30 karakter, bir dize olarak yorumu içerir.
- Sonraki bayt, parçanın türünü belirten bir kod içerir.
- Tüm metin verileri ISO 8859-1 biçimindedir.

`id3TagRead()` yöntemi, okunduktan sonra verileri kontrol eder (`complete` olayından sonra):

```
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);
    }
}
```

Ayrıca dosyada rastgele erişimli yazma işlemi de gerçekleştirebilirsiniz. Örneğin `id3Title` değişkenini baş harfinin doğru şekilde büyütüldüğünden emin olmak için ayrıştırabilirsiniz (String sınıfının yöntemlerini kullanarak) ve ardından aşağıdaki gibi, dosyaya `newTitle` adlı değiştirilmiş bir dize yazabilirsiniz:

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

ID3 sürüm 1 standardına uymak için, `newTitle` dizesinin uzunluğu, sonunda karakter kodu 0 ile tamponlu 30 karakter olmalıdır (`String.fromCharCode(0)`).

Bölüm 39: Yerel verileri saklama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İstemci bilgisayarında az miktarda veri depolamak için SharedObject sınıfını kullanırsınız. Adobe AIR'de, bir AIR uygulamasında yerel bilgisayarda gizlilik açısından önem taşıyan az miktarda kullanıcı verisi depolamak için EncryptedLocalStore sınıfını da kullanabilirsiniz.

Ayrıca dosya sisteminde (Adobe AIR'de) dosya okuyabilir ve yazabilirsiniz ve yerel veritabanı dosyalarına erişebilirsiniz. Daha fazla bilgi için, bkz. "Dosya sistemiyle çalışma" sayfa 659 ve "AIR'de yerel SQL veritabanlarıyla çalışma" sayfa 722

Paylaşılan nesnelerle ilgili bazı güvenlik faktörleri vardır. Daha fazla bilgi için, bkz. "Güvenlik" sayfa 1053 bölümünde "Paylaşılan nesneler" sayfa 1086.

Paylaşılan nesneler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazen "Flash tanımlama bilgisi" olarak da ifade edilen paylaşılan bir nesne, ziyaret ettiğiniz siteler tarafından bilgisayarınızda oluşturulabilen bir veri dosyasıdır. Paylaşılan nesneler daha çok web tarama deneyiminizi geliştirmek için kullanılır—örneğin, sıkça ziyaret ettiğiniz bir web sitesinin görünümünü kişiselleştirmenize olanak sağlar.

Paylaşılan nesneler hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Paylaşılan nesneler tarayıcı tanımlama bilgileri gibi işlev görür. Kullanıcının yerel sabit sürücüsünde veri depolamak ve aynı oturumda veya daha sonraki bir oturumda bu veriyi çağırmak için SharedObject sınıfını kullanırsınız. Uygulamalar yalnızca kendi SharedObject verilerine erişebilir, ve bunu yalnızca aynı etki alanında çalışıyorsa yapabilir. Veri sunucuya gönderilmez ve diğer etki alanlarında çalışan uygulamaları tarafından erişilemez, ancak aynı etki alanında bulunan uygulamalar tarafından erişilebilir hale getirilebilir.

Tanımlama bilgileriyle karşılaştırılan paylaşılan objeler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tanımlama bilgileri ve paylaşılan nesneler oldukça benzerdir. Birçok web programcısı tanımlama bilgilerinin nasıl işlediğini bildiği için tanımlama bilgilerini ve yerel SharedObject öğelerini harmanlamak faydalı olabilir.

RFC 2109 standardına bağlı kalan tanımlama bilgileri genellikle aşağıdaki özelliklere sahiptir:

- Genellikle bir oturumun sonunda varsayılan olarak süreleri dolabilir.
- Siteye özgü temelde, istemci tarafından devre dışı bırakılabilirler
- Site başına toplam 300 tanımlama bilgisi ve maksimum 20 tanımlama bilgisi sınırı vardır.
- Genellikle her biri 4 KB'lık bir boyutla sınırlıdırlar.

- Tanımlama bilgileri bazen bir güvenlik tehdidi olarak algılanabilir ve bu nedenle, bazen istemci üzerinde devre dışı bırakılır.
- İstemci tarayıcısı tarafından belirlenmiş bir konumda saklanırlar.
- HTTP üzerinden istemciden sunucuya iletilirler.

  Paylaşılan nesneler, bunun aksine aşağıdaki özelliklere sahiptir:

- Süreleri varsayılan olarak dolmaz.
- Varsayılan olarak, her biri 100 KB'lık bir boyutla sınırlanır.
- Basit veri tiplerini saklayabilirler (Dize, Dizi ve Tarih gibi).
- Uygulama tarafından belirlenen bir konumda saklanırlar (kullanıcının ana dizini içinde).
- Hiçbir zaman istemci ve sunucu arasında iletilmezler.

SharedObject sınıfı hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SharedObject sınıfını kullanarak, kullandığınız bir SharedObject nesnesinin geçerli boyutunu belirlemenin yanı sıra, paylaşılan nesneler oluşturabilir ve silebilirsiniz.

Bir paylaşılan nesne oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir SharedObject nesnesi oluşturmak için şu sözdizimine sahip olan `SharedObject.getLocal()` yöntemini kullanın:

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

Aşağıdaki örnek mySO adlı bir paylaşılan nesne oluşturur:

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

Bu işlem, istemcinin makinesinde preferences.sol adlı bir dosya oluşturur.

Yerel terimi paylaşılan nesnenin konumuyla ilgilidir. Bu durumda Adobe® Flash® Player, SharedObject dosyasını istemcinin ana dizininde yerel olarak saklar.

Bir paylaşılan nesne oluşturduğunuzda, Flash Player, sanal alanının içindeki etki alanı ve uygulama için yeni bir dizin oluşturur. Ayrıca SharedObject verisini saklayan bir *.sol dosyası oluşturur. Bu dosyanın varsayılan konumu, kullanıcının ana dizininin bir altdizinidir. Aşağıdaki tabloda bu dizinin varsayılan konumları gösterilmektedir:

İşletim Sistemi	Konum
Windows 95/98/ME/2000/XP	`c:/Documents and Settings/`*`username`*`/Application Data/Macromedia/Flash Player/#SharedObjects`
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`

#SharedObjects dizininin altında rastgele adlandırılmış bir dizin mevcuttur. Onun altında uygulamayı önce ana bilgisayar adıyla, ardından yolla ve son olarak da *.sol dosyasıyla eşleştiren bir dizin bulunmaktadır.

Örneğin, yerel ana bilgisayarda, /sos adlı bir altdizinde bulunan MyApp.swf adlı bir uygulama talep ederseniz, Flash Player *.sol dosyasını Windows XP'de aşağıdaki konumda saklar:

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

Not: *`SharedObject.getLocal()` yönteminde bir ad belirtmezseniz, Flash Player dosyayı undefined.sol olarak adlandırır.*

Varsayılan olarak Flash, etki alanı başına 100 KB'a kadar büyüklüğe sahip yerel olarak kalıcı SharedObject nesnelerini kaydedebilir. Bu değer kullanıcı tarafından yapılandırılabilir. Uygulama, bir paylaşılan nesneye veri kaydetmeye çalıştığında (bu işlem veri boyutunu 100 KB'tan fazla yapar), Flash Player, Yerel Saklama iletişim kutusunu görüntüler ve bu da kullanıcının erişim talep eden etki alanı için daha fazla yerel saklama alanına izin verebilmesini veya bunu engelleyebilmesini sağlar.

Bir yol belirleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SharedObject dosyası için konum belirtmek üzere isteğe bağlı *pathname* parametresini kullanabilirsiniz. Bu dosya o etki alanının SharedObject dizinine ait bir altdizin olmalıdır. Örneğin, yerel ana bilgisayar üzerinde bir uygulama talep edip aşağıdakileri belirlerseniz:

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

Flash Player SharedObject dosyasını /#localhost dizinine (veya uygulama çevrimdışı ise, /localhost) yazar. Bu işlem, istemci üzerindeki birden fazla uygulamanın aynı paylaşılan nesneye erişebilmesini istiyorsanız kullanışlıdır. Bu durumda, istemci ikisi de etki alanının kökü olan paylaşılan nesneye giden bir yol belirleyen iki Flex uygulaması çalıştırabilir; böylece istemci, aynı paylaşılan nesneye iki uygulamadan da erişebilir. Birden fazla uygulama arasında kalıcılık olmadan veri paylaşmak için, LocalConnection nesnesini kullanabilirsiniz.

Olmayan bir dizin belirlerseniz, Flash Player bir SharedObject dosyası oluşturmaz.

Bir paylaşılan nesneye veri ekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SharedObject öğesinin*.sol dosyasına SharedObject nesnesinin `data` özelliğini kullanarak veri eklersiniz. Paylaşılan nesneye yeni veri eklemek için aşağıdaki sözdizimini kullanın:

```
sharedObject_name.data.variable = value;
```

Aşağıdaki örnek bir SharedObject öğesine `userName`,`itemNumbers` ve `adminPrivileges` özelliklerini ve onların değerlerini ekler.

```
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;
```

`Data` özelliğine değer atadıktan sonra, Flash Player'a o değerleri SharedObject öğesinin dosyasına yazacak şekilde talimat vermelisiniz. Flash Player'ı değerleri SharedObject öğesinin dosyasına yazmaya zorlamak için `SharedObject` öğesini kullanın.`flush()` yöntemi, aşağıdaki gibi:

```
mySO.flush();
```

`SharedObject.flush()` yöntemini çağırmazsanız, uygulamadan çıkıldığı zaman Flash Player, değerleri dosyaya yazar. Ancak bu işlem, verilerin varsayılan ayarları aşması durumunda kullanıcıya, Flash Player'ın verileri saklamak zorunda kalacağı kullanılabilir alanı artırma imkanı sağlamaz. Bu yüzden, `SharedObject.flush()` öğesini çağırmak iyi bir fikirdir.

Bir kullanıcının sabit sürücüsüne paylaşılan nesneler yazmak için `flush()` yöntemini kullanırken, aşağıdaki örnekte gösterildiği gibi, kullanıcının Flash Player Settings Manager (www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager07.html) kullanarak yerel depolama alanını açıkça devre dışı bırakıp bırakmadığını kontrol ettiğinizden emin olmanız gerekir:

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

Nesneleri paylaşılan nesnelerde saklama

Diziler veya Dizeler gibi basit nesneleri SharedObject öğesinin `data` özelliğinde saklayabilirsiniz.

Aşağıdaki örnek paylaşılan nesneyle etkileşimi kontrol eden yöntemleri tanımlayan bir ActionScript sınıfıdır. Bu yöntemler kullanıcının nesneleri paylaşılan nesneye ekleyebilmesini veya paylaşılan nesneden çıkarabilmesini sağlar. Bu sınıf basit nesneler içeren bir ArrayCollection öğesi saklar.

```
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();
        }
    }

}
```

Aşağıdaki Flex uygulaması gereksinim duyduğu paylaşılan nesne türlerinin her biri için ActionScript sınıfının bir örneğini oluşturur. Sonra da kullanıcı bloglar veya site URL'leri eklediğinde veya kaldırdığında o sınıfta yöntemler çağırır.

```
<?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>
```

Yazılan nesneleri paylaşılan nesnelerde saklama

Yazılan ActionScript örneklerini paylaşılan nesnelerde saklayabilirsiniz. Bu işlemi, sınıfı kaydetmek için `flash.net.registerClassAlias()` yöntemini çağırarak yaparsınız. Sınıfınızın bir örneğini oluşturup paylaşılan nesnenizin veri üyesinde saklar ve daha sonra nesneyi okursanız, bir yazılı örnek elde edeceksiniz. Varsayılan olarak, SharedObject `objectEncoding` özelliği AMF3 kodlamasını destekler ve saklanan örneğinizi SharedObject nesnesinden dışarı çıkarır; saklanan örnek `registerClassAlias` yöntemini çağırdığınız sırada belirlediğiniz aynı türü korur.

(Yalnızca iOS) Yerel olarak paylaşılan nesnelerin bulutta yedeklenmesini önleme

Adobe AIR 3.7 ve üstü, yalnızca iOS

Yerel olarak paylaşılan nesnelerin iOS bulut yedekleme servisinde yedeklenip yedeklenmeyeceğini denetlemek için `SharedObject.preventBackup` özelliğini ayarlayabilirsiniz. Bu, yeniden oluşturulabilen veya tekrar indirilebilen ancak çevrimiçi kullanım sırasında uygulamanızın düzgün çalışması için gerekli olan içerik için Apple tarafından gerektirilir.

Birden çok paylaşılan nesne oluşturma

Aynı Flex uygulamasından birden çok paylaşılan nesne oluşturabilirsiniz. Bu işlemi gerçekleştirmek için, paylaşılan nesnelerin her birine aşağıdaki örnekte görüldüğü gibi farklı bir örnek ismi atayabilirsiniz:

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

Bu işlem, Flex uygulamasının yerel dizininde bir preferences.sol dosyası ve bir history.sol dosyası oluşturur.

Güvenli bir SharedObject oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`getLocal()` veya `getRemote()` öğesini kullanarak yerel ya da uzak bir SharedObject oluşturduğunuzda, bu paylaşılan nesneye erişimin, HTTPS bağlantısı üzerinden teslim edilen SWF dosyalarıyla kısıtlı olup olmadığını belirleyen `secure` adında isteğe bağlı bir parametre vardır. Bu parametre `true` değerine ayarlanırsa ve SWF dosyanız HTTPS üzerinden teslim edilirse, Flash Player uygulaması yeni bir güvenli paylaşılan nesne oluşturur veya varolan güvenli paylaşılan nesneye başvuruyu alır. Yalnızca güvenli parametresi `true` değerine ayarlanmış şekilde `SharedObject.getLocal()` öğesini çağıran ve HTTPS üzerinden teslim edilen SWF dosyaları tarafından bu güvenli paylaşılan nesne okunabilir veya bu güvenli paylaşılan nesneye yazılabilir. Bu parametre `false` değerine ayarlanırsa ve SWF dosyanız HTTPS üzerinden teslim edilirse, Flash Player uygulaması yeni bir paylaşılan nesne oluşturur veya varolan paylaşılan nesneye başvuruyu alır.

HTTPS olmayan bağlantılar üzerinden teslim edilen SWF dosyaları tarafından bu paylaşılan nesne okunabilir veya bu paylaşılan nesneye yazılabilir. SWF dosyanız HTTPS olmayan bir bağlantı üzerinden teslim edilirse ve siz bu parametreyi `true` değerine ayarlamayı denerseniz, yeni paylaşılan nesne oluşturma işlemi (veya önceden oluşturulmuş güvenli paylaşılan nesne erişimi) başarısız olur, bir hata atılır ve paylaşılan nesne `null` değerine ayarlanır. HTTPS olmayan bir bağlantıdan aşağıdaki kod parçasını çalıştırmayı denerseniz, `SharedObject.getLocal()` yöntemi bir hata atar:

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

Bu parametrenin değerine bakılmaksızın, oluşturulmuş paylaşılan nesnelerin sayısı, bir etki alanı için izin verilen toplam disk alanı miktarı kadar olabilir.

Paylaşılan bir nesnenin içeriklerini görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Değerler, `data` özelliği içinde paylaşılan nesnelerde saklanır. Aşağıdaki örnekte gösterildiği gibi, bir `for..in` döngüsünü kullanarak paylaşılan nesne örneği içinde her değer üzerinde döngü oluşturabilirsiniz:

```
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]);
}
```

Paylaşılan nesneleri ortadan kaldırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İstemcide bulunan bir SharedObject öğesini ortadan kaldırmak için, `SharedObject.clear()` yöntemini kullanın. Bu işlem, uygulamanın paylaşılan nesnelerine ait varsayılan yoldaki dizinleri yok etmez.

Aşağıdaki örnek istemciden SharedObject dosyasını siler:

```
public function destroySharedObject():void {
    mySO.clear();
}
```

SharedObject örneği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, bir Date nesnesi gibi basit nesneleri, o nesneleri el ile serileştirmek veya seri numarasını kaldırmak zorunda kalmadan bir SharedObject nesnesinde depolayabileceğinizi gösterir.

Aşağıdaki örnek sizi ilk defa gelen bir ziyaretçi gibi karşılayarak başlar. Oturumu Kapat'ı tıklattığınızda, uygulama geçerli tarihi paylaşılan bir nesnede saklar. Bu uygulamayı bir sonraki sefer başlattığınızda veya sayfayı yenilediğinizde, uygulama sizi oturumu kapattığınız zamanı gösteren bir hatırlatma ile tekrar karşılar.

Uygulamayı çalışır halinde görmek için başlatın, Oturumu Kapat'ı tıklatın ve ardından sayfayı yenileyin. Uygulama bir önceki ziyaretinizde Oturumu Kapat düğmesini tıklattığınız tarih ve zamanı gösterir. Saklanan bilgileri istediğiniz zaman LSO'yu Sil düğmesine basarak silebilirsiniz.

```
<?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>
```

Şifrelenmiş yerel depolama

EncryptedLocalStore sınıfı (ELS), bir uygulamaya ilişkin kişisel veriler için küçük bir önbellek olarak kullanabileceğiniz, şifreli, yerel bir saklama mekanizması sağlar. ELS verileri uygulamalar arasında paylaşılamaz. ELS'nin amacı, oturum açmak için gerekli kimlik bilgileri ve diğer kişisel bilgiler gibi kolayca yeniden oluşturulabilen öğeleri saklamak için bir uygulama sağlamaktır. Aşağıdaki "Şifreli yerel saklama alanı sınırlamaları" ve "En iyi uygulamalar" içinde vurgulandığı üzere, ELS verisi kalıcı olarak kabul edilmemelidir.

Not: *AIR, şifreli yerel deponun yanı sıra SQL veritabanlarında saklanan içerik için şifreleme de sağlar. Ayrıntılar için bkz. "Şifrelemeyi SQL veritabanlarıyla kullanma" sayfa 766.*

Web hizmetleri için oturum açma bilgileri gibi güvence altına alınması gereken bilgileri önbelleğe almak için şifreli yerel depo kullanmak isteyebilirsiniz. ELS, diğer kullanıcılardan gizli tutulması gereken bilgileri depolamak için uygundur. Ancak, verileri aynı kullanıcı hesabında çalışan diğer işlemlerden korumaz. Bu yüzden, DRM veya şifreleme anahtarları gibi gizli uygulama verilerini korumak için uygun değildir.

Masaüstü platformlarında AIR, şifreli yerel depoyu her bir uygulama ve kullanıcıyla ilişkilendirmek için Windows'ta DPAPI'yı, Mac OS'de ve iOS'de KeyChain'i, Linux'ta ise KeyRing veya KWallet'ı kullanır. Şifreli yerel depo AES-CBC 128-bit şifreleme kullanır.

EncryptedLocalStorage sınıfı tarafından depolanan veriler Android'de şifrelenmez. Bunun yerine, veriler işletim sistemi tarafından sağlanan kullanıcı düzeyi güvenliğince korunur. Android işletim sistemi her uygulamaya ayrı bir kullanıcı kimliği atar. Uygulamalar yalnızca kendi dosyalarına ve ortak konumda oluşturulmuş dosyalara (kaldırılabilir depolama kartı gibi) erişebilir. "Root edilmiş" Android cihazlarında kök kullanıcı yetkileri ile çalışan uygulamaların, diğer uygulama dosyalarına giriş YAPABİLDİĞİNİ unutmayın. Bu nedenle şifreli yerel depo, root edilmiş bir cihazda root edilmemiş bir cihazda olduğu kadar yüksek bir veri koruma düzeyi sağlamaz.

Şifreli yerel depodaki bilgiler yalnızca uygulama güvenliği sanal alanındaki AIR uygulama içeriği tarafından kullanılabilir.

Bir AIR uygulamasını güncellerseniz, şu durumlarda güncellenmiş sürüm şifreli yerel depodaki herhangi bir veriye erişimi engeller:

- Öğeler, `true` olarak ayarlanmış `stronglyBound` öğesiyle eklenmemişse
- Varolan ve güncellenen sürümler AIR 1.5.3’ten önce yayınlanır ve güncelleme bir geçiş imzasıyla imzalanır.

Şifreli yerel depo sınırları

Şifreli yerel depodaki veriler, kullanıcının işletim sisteminin hesap kimlik bilgileri tarafından korunur. Diğer varlıklar o kullanıcı olarak oturum açmadıkları sürece depodaki veriye erişemez. Ancak veriler, kimliği doğrulanmış bir kullanıcı tarafından çalıştırılan diğer uygulamaların erişimine karşı güvenli değildir.

Bu saldırıların çalışması için kullanıcının kimlik doğrulamasından geçmiş olması gerektiğinden, kullanıcının özel verisi korunmaya devam eder (kullanıcı hesabının kendisinden ödün verilmiş olmadığı sürece). Ancak, uygulamanızın kullanıcılardan gizli tutmak isteyebileceği veriler, örneğin lisans için kullanılan anahtarlar veya dijital haklar yönetimi, güvenli değildir. Bu yüzden, ELS bu tip bilgileri depolamak için uygun bir konum değildir. Yalnızca şifre gibi özel kullanıcı verilerini depolamak için uygun bir yerdir.

ELS'deki veriler birkaç nedenden dolayı kaybolabilir. Örneğin, kullanıcı uygulamayı kaldırabilir ve şifreli dosyayı silebilir. Ayrıca, yayıncı kimliği bir güncelleme sonucu değiştirilebilir. Bu yüzden ELS, kalıcı bir veri deposu olarak değil özel bir önbellek gibi düşünülmelidir.

`stronglyBound` parametresi onaylanmamıştır ve `true` olarak ayarlanmamalıdır. Parametreyi `true` olarak ayarlamak veriler için ek bir koruma sağlamaz. Aynı zamanda, yayıncı kimliği aynı kalsa bile, uygulama her güncellendiğinde verilere olan erişim kaybedilir.

Saklanılan verilerin boyutu 10MB'ı geçerse şifreli yerel depo yavaş çalışabilir.

Bir AIR uygulamasını kaldırdığınızda, kaldırma uygulaması şifreli yerel depoda saklanan verileri silmez.

En iyi uygulamalar

ELS'yi kullanmanın en iyi şekillerine şunlar dahildir:

- ELS'yi şifreler gibi (stronglyBound'u false değerine ayarlayarak) depolama duyarlı kullanıcı verilerini depolamak için kullanın.
- ELS'yi DRM anahtarları veya lisans verme işaretleri gibi uygulama sırlarını saklamak için kullanmayın..
- ELS verileri kaybolduysa, uygulamanızın ELS'de depolanmış verileri yeniden oluşturması için bir yol sağlayın. Örneğin, gerektiğinde kullanıcıdan hesap kimlik bilgilerini yeniden girmesini isteyerek bu işlemi gerçekleştirebilirsiniz.
- `stronglyBound` parametresini kullanmayın.

- `stronglyBound` parametresini `true` olarak ayarlarsanız, depolanmış öğelere güncelleme sırasında geçiş yaptırmayın. Bunun yerine verileri güncellemeden sonra yeniden oluşturun.
- Yalnızca nispeten daha az miktarda veri depolayın. Daha fazla miktardaki veriler için şifreli bir AIR SQL veritabanı kullanın.

Daha fazla Yardım konusu

flash.data.EncryptedLocalStore

Şifrelenmiş yerel depoya veri ekleme

Yerel depoda veri saklamak için EncryptedLocalStore sınıfının `setItem()` statik yöntemini kullanın. Veriler, verileri bayt dizileri olarak saklayan ve dizeleri anahtar olarak kullanan karma bir tabloda saklanır.

Örneğin aşağıdaki kod bir dizeyi şifreli yerel depoda saklar:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

`setItem()` yönteminin üçüncü parametresi olan `stronglyBound` isteğe bağlıdır. Bu parametre `true` olarak ayarlandığında, şifreli yerel depo, depolanan öğeyi saklayan AIR uygulamasının dijital imza ve kısımlarına bağlar:

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

`stronglyBound` parametresi `true` olarak ayarlıyken depolanan bir öğe için, `getItem()` öğesine yapılan sonraki çağrılar yalnızca çağıran AIR uygulamasını depolayan uygulamayla aynıysa başarılı olur (uygulama dizininde bulunan dosyalardaki hiçbir veri değiştirilmemişse). Çağıran AIR uygulaması saklayan uygulamadan farklıysa, güçlü şekilde bağlı bir öğe için `getItem()` öğesini çağırdığınızda uygulama bir Hata istisnası atar. Uygulamanızı güncellerseniz, şifreli yerel depoya önceden yazılmış olan güçlü şekilde bağlı verileri okuyamaz. Mobil cihazlarda `stronglyBound` öğesinin `true` olarak ayarlanması yoksayılır; parametre her zaman `false` olarak değerlendirilir.

`stronglyBound` parametresi `false` değerine ayarlıysa (varsayılan olarak), uygulamanın veriyi okuması için yalnızca yayıncı kimliğinin aynı kalması gerekir. Uygulamanın bazı kısımları değişebilir (ve aynı yayıncı tarafından imzalanmaları gerekir) ancak veriyi depolayan uygulamada olduğu gibi aynı kısımlar olmaları gerekmez. Orijinaliyle aynı yayıncı kimliğine sahip olan güncellenmiş uygulamalar verilere erişmeye devam edebilir.

Not: *Pratikte, `stronglyBound` öğesinin `true` olarak ayarlanması fazladan veri koruması sağlamaz. "Kötü niyetli" bir kullanıcı ELS'de depolanan öğelere erişebilmek için uygulamayı değiştirebilir. Ayrıca, `stronglyBound` öğesinin `true` veya `false` olarak ayarlanmasına bakılmaksızın veriler harici ve kullanıcı kaynaklı olmayan tehditlerden aynı şekilde korunur. Bu nedenlerle, `stronglyBound` öğesinin `true` olarak ayarlanması önerilmez.*

Şifrelenmiş yerel depodaki veriye erişme

Adobe AIR 1.0 ve üstü

`EncryptedLocalStore.getItem()` yöntemini kullanarak, aşağıdaki örnekte olduğu gibi şifreli yerel depodan bir değer alabilirsiniz:

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

Şifrelenmiş yerel depodan veri kaldırma

Adobe AIR 1.0 ve üstü

`EncryptedLocalStore.removeItem()` yöntemini kullanarak, aşağıdaki örnekte olduğu gibi şifreli yerel depodan bir değer silebilirsiniz:

```
EncryptedLocalStore.removeItem("firstName");
```

`EncryptedLocalStore.reset()` yöntemini çağırarak, aşağıdaki örnekte olduğu gibi şifreli yerel depodan tüm verileri temizleyebilirsiniz:

```
EncryptedLocalStore.reset();
```

Bölüm 40: AIR'de yerel SQL veritabanlarıyla çalışma

Adobe AIR 1.0 ve üstü

Adobe® AIR®, yerel SQL veritabanları oluşturma ve bu veritabanlarıyla çalışma özelliğini içerir. Çalışma zamanı, açık kaynak SQLite veritabanı sistemini kullanarak birçok standart SQL özelliğini destekleyen bir SQL veritabanı motoru içerir. Yerel bir SQL veritabanı, yerel, kalıcı verileri saklamak için kullanılabilir. Örneğin uygulama verileri, uygulama kullanıcı ayarları, belgeler veya uygulamanızın yerel olarak kaydetmesini istediğiniz diğer türde veriler için kullanılabilir.

Yerel SQL veritabanları hakkında

Adobe AIR 1.0 ve üstü

SQL veritabanlarının kullanmakla ilgili hızlı açıklama ve kod örnekleri için aşağıda bulunan Adobe Geliştirici Bağlantısı'ndaki hızlı başlatma makalelerine bakın.

- Yerel bir SQL veritabanıyla senkronize olmayan şekilde çalışma (Flex)
- Yerel bir SQL veritabanıyla senkronize çalışma) (Flex)
- Şifreli bir veritabanı kullanma) (Flex)
- Yerel bir SQL veritabanıyla senkronize olmayan şekilde çalışma) (Flash)
- Yerel bir SQL veritabanıyla senkronize çalışma) (Flash)
- Şifreli bir veritabanı kullanma) (Flash)

Adobe AIR, AIR uygulamasının çalıştığı bilgisayardaki veritabanı dosyalarında yerel olarak saklanan verileri içeren çalışma zamanında çalışan SQL tabanlı ilişkisel bir veritabanı motoru içerir (örneğin, bilgisayarın sabit sürücüsünde). Veritabanı yerel olarak çalıştığı ve veri dosyaları yerel olarak saklandığı için, bir veritabanı AIR uygulaması tarafından ağ bağlantısının kullanılabilir olup olmadığına bakılmaksızın kullanılabilir. Bu nedenle, çalışma zamanının yerel SQL veritabanı motoru, özellikle SQL ve ilişkisel veritabanlarlarına ilişkin deneyiminiz varsa, kalıcı, yerel uygulama verilerinin saklanması için uygun bir mekanizma sağlar.

Yerel SQL veritabanları için kullanımlar

Adobe AIR 1.0 ve üstü

AIR yerel SQL veritabanı işlevi, uygulama verilerini kullanıcının yerel bilgisayarında saklamak isteyebileceğiniz tüm amaçlar için kullanılabilir. Adobe AIR, verilerin yerel olarak saklanması için, her biri farklı avantajlara sahip çeşitli mekanizmalar içerir. AIR uygulamanızdaki yerel SQL veritabanı için bazı olası kullanımlar şunlardır:

- Veritabanı, veri odaklı bir uygulama (örneğin bir adres defteri) için ana uygulama verilerini saklamak amacıyla kullanılabilir.

- Kullanıcıların kaydetmek ve belki de paylaşmak için belgeler oluşturduğu belge odaklı uygulamalar için, belgeler kullanıcının belirlediği konumda bir veritabanı dosyası olarak kaydedilebilir. (Ancak, veritabanı şifreli olmadıkça herhangi bir AIR uygulamasının veritabanı dosyasını açabileceğini dikkate alın. Şifreleme, duyarlı olabilecek belgeler için önerilir.)
- Veritabanı, ağ kullanan bir uygulama için uygulama verilerinin yerel önbelleğini saklamak veya ağ bağlantısı kullanılabilir olmadığında verileri geçici olarak saklamak üzere kullanılabilir. Yerel veritabanını ağ veri deposuyla senkronize etmek için bir mekanizma oluşturabilirsiniz.
- Veritabanı, herhangi bir uygulama için tek bir kullanıcının uygulama ayarlarını, örneğin pencere boyutu ve konumu gibi kullanıcı seçenekleri ve uygulama bilgisini saklamak üzere kullanılabilir.

Daha fazla Yardım konusu

Christophe Coenraets: Employee Directory on AIR for Android

Raymond Camden: jQuery and AIR - Moving from web page to application

AIR veritabanları ve veritabanı dosyaları hakkında

Adobe AIR 1.0 ve üstü

Tek bir Adobe AIR yerel SQL veritabanı, bilgisayarın dosya sisteminde tek bir dosya olarak saklanır. Çalışma zamanı, veritabanı dosyalarının oluşturulup yapılandırılmasını, verilerin bir veritabanı dosyasından alınması ve değiştirilmesini yöneten bir SQL veritabanı motoru içerir. Çalışma zamanı, veritabanı verilerinin dosya sisteminin neresinde ve nasıl saklandığını belirtmez; bunun yerine, her veritabanının tamamı tek bir dosyada saklanır. Veritabanı dosyasının saklandığı dosya ssitemindeki konumu siz belirlersinz. Tek bir AIR uygulaması bir veya daha çok ayrı veritabanına erişebilir. (Bunlar, ayrı veritabanı dosyalarıdır.) Çalışma zamanı her veritabanını dosya sisteminde tek bir dosya olarak sakladığından, veritabanınızı uygulamanızın tasarımının ve işletim sistemi dosya erişim sınırlamalarının gerektirdiği şekilde konumlandırabilirsiniz. Her kullanıcı, özel verileri için ayrı bir veritabanı dosyasına sahiptir veya paylaşılan veriler için tek bir bilgisayardaki tüm uygulama kullanıcıları veritabanı dosyasına erişebilir. Veriler tek bir bilgisayar için yerel olduğundan, farklı bilgisayarlardaki kullanıcılar tarafından otomatik olarak paylaşılmaz. Yerel SQL veritabanı motoru, uzak veya sunucu tabanlı bir veritabanına karşı SQL ifadelerinin yürütülmesi için herhangi bir özellik sağlamaz.

İlişkisel veritabanları hakkında

Adobe AIR 1.0 ve üstü

İlişkisel veritabanı, verilerin bir bilgisayarda saklanmasına (ve geri alınmasına) yönelik bir mekanizmadır. Veriler, tablolar halinde düzenlenir: satırlar kayıtları veya öğeleri temsil eder ve sütunlar (bazen “alanlar” da denir) her kaydı ayrı ayrı değerlere böler. Örneğin, adres defteri uygulaması bir “arkadaşlar” tablosu içerebilir. Tablodaki her satır, veritabanında saklanan tek bir arkadaşı temsil edecektir. Tablonun sütunları, ad, soyad, doğum tarihi vb. gibi verileri temsil eder. Veritabanı, tablodaki her arkadaş satırına ilişkin her sütun için ayrı bir değer saklar.

İlişkisel veritabanları, bir öğenin başka türden öğelerle ilişkilendirilmiş veya başka türden öğelerle olduğu karmaşık verilerin saklanması için tasarlanmıştır. İlişkisel bir veritabanında, bire çok ilişkiye sahip —tek bir kaydın farklı türdeki birden çok kayıtla ilişkilendirilebildiği —veriler farklı tablolar halinde bölümlenmelidir. Örneğin, adres defteri uygulamanızın her arkadaş için birden çok telefon numarası saklamasını istediğinizi varsayın; bu, bire çok ilişkidir. “Arkadaşlar” tablosu, her arkadaş için tüm kişisel bilgileri içerecektir. Ayrı bir “telefon numaraları” tablosu, tüm arkadaşlara ilişkin tüm telefon numaralarını içerir.

Arkadaşlara ve telefon numaralarına ilişkin verileri saklamanın yanı sıra, her tablo, iki tablo arasındaki ilişkiyi izlemek için—ayrı ayrı arkadaş kayıtlarını telefon numaralarıyla eşlemek için bir veri parçasına ihtiyaç duyacaktır. Bu veri, ana anahtar—tablodaki her satırı bu tablodaki diğer satırlardan ayırt eden benzersiz bir kimlik olarak bilinir. Ana anahtar, tablodaki her kaydı doğal olarak ayırt eden veri öğelerinden biri olduğu anlamına gelen "doğal anahtar" olabilir. Hiçbir arkadaşınızın aynı doğum tarihini paylaşmadığını biliyorsanız, "arkadaşlar" tablosunda doğum tarihi sütununu ana anahtar (doğal anahtar) olarak kullanabilirsiniz. Doğal anahtar yoksa, "arkadaş kimliği" gibi, uygulamanın satırları ayırt etmek için kullanacağı yapay bir değer olan ayrı bir ana anahtar sütunu oluşturabilirsiniz.

Ana anahtarı kullanarak birden çok tablo arasında ilişkiler kurabilirsiniz. Örneğin, "arkadaşlar" tablosunun her satır (her arkadaş) için benzersiz bir numara içeren bir "arkadaş kimliği" sütununa sahip olduğunu düşünün. İlgili "telefon numaraları" tablosu, iki sütunla yapılandırılabilir: telefon numarasının ait olduğu arkadaşın "arkadaş kimliği" bilgisini içeren sütun ve geçerli telefon numarasını içeren sütun. Bu şekilde, tek bir arkadaş kaç telefon numarasına sahip olursa olsun, bunların tümü "telefon numaraları tablosunda saklanabilir ve "arkadaş kimliği" ana anahtarı kullanılarak ilgili arkadaşa bağlanabilir. Bir tablodaki ana anahtar, ilgili bir tabloda kayıtlar arasındaki bağlantıyı belirtmek üzere kullanıldığında, ilgili tablodaki değer yabancı anahtar olarak bilinir. Birçok veritabanıın aksine, AIR yerel veritabanı motoru, eklenen veya güncellenen yabancı anahtara ana anahtar tablosunda karşılık gelen bir satırın bululup bulunmadığını otomatik olarak kontrol eden sınırlamalar olan yabancı anahtar sınırlamaları oluşturmanıza olanak vermez. Yine de, yabancı anahtar ilişkileri ilişkisel veritabanı yapısının önemli bir parçasıdır ve veritabanınızdaki tablolar arasında ilişki oluştururken yabancı anahtarlar kullanılmalıdır.

SQL hakkında

Adobe AIR 1.0 ve üstü

Yapılandırılmış Sorgu Dili (Structured Query Language - SQL) verileri değiştirmek ve getirmek için ilişkisel veritabanlarıyla kullanılır. SQL, yordam temelli bir dilden çok, tanımlayıcı bir dildir. Bir SQL ifadesi, verilerin nasıl getirileceğine ilişkin bilgisayar talimatları vermek yerine, istediğiniz veri kümesini açıklar. Veritabanı motoru, verilerin nasıl getirileceğini belirler.

SQL dili, American National Standards Institute (ANSI) tarafından standartlaştırılmıştır. Adobe AIR yerel SQL veritabanı SQL-92 standardının büyük bölümünü destekler.

Adobe AIR'de desteklenen SQL dilinin belirleyici açıklamaları için bkz. "yerel veritabanlarında SQL desteği" sayfa 1116.

SQL veritabanı sınıfları hakkında

Adobe AIR 1.0 ve üstü

ActionScript 3.0'da yerel SQL veritabanlarıyla çalışmak için bu sınıfların flash.data paketindeki örneklerini kullanırsınız:

Sınıf	Açıklama
flash.data.SQLConnection	Veritabanları (veritabanı dosyaları) oluşturma ve açmaya yönelik yolların yanı sıra, veritabanı düzeyinde işlemler gerçekleştirmek ve veritabanı işlemlerini denetlemek için yöntemler sağlar.
flash.data.SQLStatement	İfade metninin tanımlanması ve parametre değerlerinin ayarlanması dahil, veritabanında yürütülen tek bir SQL ifadesini (tek bir sorgu veya komut) temsil eder.
flash.data.SQLResult	`SELECT` ifadesinden sonuç satırları, `UPDATE` veya `DELETE` ifadesinden etkilenen bir dizi satır vb. ifadelerin yürütülmesinden doğan sonuçların ve bu yürütmelere ilişkin bilgilerin getirilmesi için bir yol sağlar.

Veritabanı yapısını açıklayan şema bilgilerini elde etmek için flash.data paketindeki şu sınıfları kullanırsınız:

Sınıf	Açıklama
flash.data.SQLSchemaResult	`SQLConnection.loadSchema()` yönteminin çağrılmasıyla oluşan veritabanı şeması sonuçları için kap görevi yapar.
flash.data.SQLTableSchema	Veritabanındaki tek bir tabloyu açıklayan bilgiyi sağlar.
flash.data.SQLViewSchema	Veritabanındaki tek bir görünümü açıklayan bilgiyi sağlar.
flash.data.SQLIndexSchema	Veritabanındaki tek bir tablo veya görünüm sütununu açıklayan bilgiyi sağlar.
flash.data.SQLTriggerSchema	Veritabanındaki tek bir tetikleyiciyi açıklayan bilgiyi sağlar.

flash.data paketindeki diğer sınıflar, SQLConnection sınıfı ve SQLColumnSchema sınıfıyla kullanılan sabitleri sağlar:

Sınıf	Açıklama
flash.data.SQLMode	`SQLConnection.open()` ve `SQLConnection.openAsync()` yöntemlerinin `openMode` parametresi için olası değerleri temsil eden bir dizi sabiti tanımlar.
flash.data.SQLColumnNameStyle	`SQLConnection.columnNameStyle` özelliği için olası değerleri temsil eden bir dizi sabiti tanımlar.
flash.data.SQLTransactionLockType	`SQLConnection.begin()` yönteminin seçenek parametresi için olası değerleri temsil eden bir dizi sabiti tanımlar.
flash.data.SQLCollationType	`SQLColumnSchema()` yapıcısının `SQLColumnSchema.defaultCollationType` özelliği ve `defaultCollationType` parametresi için olası değerleri temsil eden bir dizi sabiti tanımlar.

Ayrıca, flash.events paketinde bulunan aşağıdaki sınıflar kullandığınız olayları (ve desteklenen sabitleri) temsil eder:

Sınıf	Açıklama
flash.events.SQLEvent	İşlemlerinden herhangi biri başarıyla yürütüldüğünde SQLConnection veya SQLStatement örneklerinin gönderdiği olayları tanımlar. Her işlem, SQLEvent sınıfında tanımlanmış, ilişkili bir olay türüne sahiptir.
flash.events.SQLErrorEvent	İşlemlerinden herhangi biri hatayla sonuçlandığında, SQLConnection veya SQLStatement örneklerinin gönderdiği olayı tanımlar.
flash.events.SQLUpdateEvent	Bağlı veritabanlarından birinde yer alan tablo verileri, yürütülen bir `INSERT`, `UPDATE` veya `DELETE` SQL ifadesinin sonucu olarak değiştiğinde SQLConnection örneğinin gönderdiği olayı tanımlar.

Son olarak, flash.errors paketinde yer alan aşağıdaki sınıflar, veritabanı işlem hataları hakkında bilgi sağlar:

Sınıf	Açıklama
flash.errors.SQLError	Gerçekleştirilmeye çalışılan işlem ve hatanın nedeni dahil, veritabanı işlemi hatası hakkında bilgi sağlar.
flash.errors.SQLErrorOperation	Hatayla sonuçlanan veritabanı işlemini gösteren SQLError sınıfının `operation` özelliği için olası değerleri temsil eden bir dizi sabiti tanımlar.

Senkronize ve senkronize olmayan yürütme modları hakkında

Adobe AIR 1.0 ve üstü

Yerel SQL veritabanıyla çalışmak üzere bir kod yazdığınızda, bu veritabanı işlem yürütmesini iki yürütme modundan birinde belirlersiniz: senkronize olmayan veya senkronize yürütme modu. Kod örnekleri genellikle ihtiyaçlarınıza en uygun olan örneği kullanabilmeniz için, her bir işlemin her iki yolla da nasıl gerçekleştirileceğini gösterir.

Senkronize olmayan yürütme modunda, çalışma zamanına bir talimat verirsiniz ve istediğiniz işlem tamamlandığında veya başarısız olduğunda çalışma zamanı bir olay gönderir. Önce veritabanı motoruna bir işlem gerçekleştirmesini söylersiniz. Uygulama çalışmaya devam ederken, veritabanı motoru işini arka planda yapar. Son olarak, veritabanı motoru işlem tamamlandığında (veya başarısız olduğunda) bir olay gönderir. Olay tarafından tetiklenen kodunuz, sonraki işlemleri gerçekleştirir. Bu yaklaşımın önemli bir faydası vardır: ana uygulama kodu çalışmaya devam ederken, çalışma zamanı veritabanı işlemlerini arka planda gerçekleştirir. Veritabanı işlemi büyük ölçüde zaman alırsa, uygulama çalışmaya devam eder. En önemlisi, kullanıcı ekranı dondurmadan uygulamayla etkileşim kurmaya devam edebilir. Yine de, senkronize olmayan işlem kodunun yazılması diğer kodlara göre daha karmaşık olabilir. Bu karmaşıklık, genellikle çoklu bağımlı işlemlerin çeşitli olay dinleyici yöntemlerine bölümlenmesi gerektiği durumlarda mevcuttur.

Kavramsal olarak, işlemleri birkaç olay dinleyici yöntemine bölünmüş bir işlem kümesi yerine, tek bir adım sırası—bir senkronize işlemler kümesi—olarak kodlamak daha kolaydır. Senkronize olmayan veritabanı işlemlerine ek olarak, Adobe AIR veritabanı işlemlerini senkronize olarak yürütmenize de olanak verir. Senkronize yürütme modunda, işlemler arka planda çalışmaz. Bunun yerine, diğer tüm uygulama kodları gibi aynı yürütme sırasında çalışır. Veritabanı motoruna bir işlem gerçekleştirmesini söylersiniz. Daha sonra, veritabanı motoru işini yaparken, kod bu noktada duraklar. İşlem tamamlandığında, yürütme sonraki kod satırıyla devam eder.

İşlemlerin senkronize olmayan veya senkronize biçimde yürütüleceği, SQLConnection düzeyinde ayarlanır. Tek bir veritabanı bağlantısı kullanarak bazı işlemler ve ifadeleri senkronize, diğerlerini senkronize olmayan biçimde yürütemezsiniz. Veritabanını açmak için SQLConnection yöntemini çağırarak, SQLConnection öğesinin senkronize veya senkronize olmayan yürütme modunda çalışacağını belirlersiniz. `SQLConnection.open()` yöntemini çağırırsanız, bağlantı senkronize yürütme modunda, `SQLConnection.openAsync()` yöntemini çağırırsanız senkronize olmayan yürütme modunda çalışır. SQLConnection örneği, `open()` veya `openAsync()` kullanılarak bir veritabanına bağlandığında, veritabanı bağlantısını kapatıp açmadığınız sürece, senkronize veya senkronize olmayan yürütme moduna sabitlenir.

Her yürütme modunun kendi faydaları vardır. Modların çoğu yönü birbirine benzese de, her bir modda çalışırken aklınızda bulundurmak isteyeceğiniz bazı farklar vardır. Bu başlıklar hakkında daha fazla bilgi ve her bir modda çalışmaya ilişkin öneriler için bkz. “Senkronize ve senkronize olmayan veritabanı işlemlerini kullanma” sayfa 761.

Veritabanı oluşturma ve değiştirme

Adobe AIR 1.0 ve üstü

Uygulamanızın verileri ekleyebilmesi ve getirebilmesi için, uygulamanızın erişebileceği, içinde tanımlanmış tablolar bulunan bir veritabanı gereklidir. Burada, bir veritabanı oluşturma ve bir veritabanı içindeki veri yapısını oluşturmaya ilişkin görevler tanımlanmıştır. Bu görevler veri ekleme ve veri getirmeden daha az kullanılsa da, çoğu uygulama için gereklidir.

Daha fazla Yardım konusu

Mind the Flex: Updating an existing AIR database

Veritabanı oluşturma

Adobe AIR 1.0 ve üstü

Bir veritabanı dosyası oluşturmak için, önce bir SQLConnection örneği oluştururusunuz. Bunu senkronize yürütme modunda açmak için `open()` yöntemini veya senkronize olmayan yürütme modunda açmak için `openAsync()` yöntemini çağırırsınız. `open()` ve `openAsync()` yöntemleri, bir veritabanı bağlantısını açmak için kullanılır. `reference` parametresine (ilk parametre) ilişkin var olmayan bir dosyaya başvuran bir File örneği ilettiğinizde, `open()` veya `openAsync()` yöntemi bu dosya konumunda bir veritasbanı dosyası oluşturur ve yeni oluşturulan veritabanına bağlantı kurar.

Veritabanı oluşturmak için `open()` yöntemini veya `openAsync()` yöntemini çağırdığınızda, veritabanı dosyasının adı herhangi bir dosya adı uzantısına sahip olan herhangi bir geçerli dosya adı olabilir. `reference` parametresi için `null` değeriyle `open()` veya `openAsync()` yöntemini çağırdığınızda, diskteki bir veritabanı dosyası yerine yeni bir bellek içi veritabanı oluşturulur.

Aşağıdaki kod listesi, senkronize yürütme modu yoluyla bir veritabanı dosyası (yeni bir veritabanı) oluşturma işlemini gösterir. Bu durumda, veritabanı dosyası "Uygulama depolama dizinine işaret etme" sayfa 679 "DBSample.db" dosya adıyla kaydedilir:

```
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>
```

Not: *File sınıfı belirli bir yerel dosya yoluna işaret etmenize izin verse de, bunu yapmak farklı platformlar üzerinde çalışmayacak uygulamalara sebep olabilir. Örneğin, C:\Documents and Settings\cem\test.db yalnızca Windows'da çalışır. Bu sebeplerden dolayı, File sınıfı'nın statik özelliklerini kullanmak en iyisidir, örneğin* `File.applicationStorageDirectory` *ve* `resolvePath()` *yöntemi (önceki örnekte gösterildiği gibi). Daha fazla bilgi için, bkz. "File nesnelerinin Yolları" sayfa 675.*

SQLConnection örneğiyle bir veritabanı bağlantısını açtığınızda, işlemleri senkronize olarak yürütmek için `open()` yöntemini çağırın. Aşağıdaki örnek, işlemlerini senkronize olarak yürüten bir SQLConnection örneğinin nasıl oluşturulacağını ve açılacağını gösterir:

```
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>
```

Veritabanı tabloları oluşturma

Adobe AIR 1.0 ve üstü

Veritabanında bir tablo oluşturma, bir SQL ifadesini yürütmek için kullandığınız `SELECT`, `INSERT` vb. gibi bir işlemin aynısını kullanarak bu veritabanında bir SQL ifadesi yürütmeyi içerir. Tablo oluşturmak için, sütun tanımlarını ve yeni tabloya ilişkin sınırlamaları içeren `CREATE TABLE` ifadesini kullanırsınız. SQL ifadelerini yürütme hakkında daha fazla bilgi için bkz. "SQL ifadeleriyle çalışma" sayfa 737.

Aşağıdaki örnek, var olan bir veritabanı dosyasında senkronize olmayan yürütme modunu kullanarak "çalışanlar" adlı bir tablo oluşturmayı gösterir. Bu kodun somut olarak gösterilmiş ve bir veritabanına bağlanmış `conn` adı altında bir SQLConnection örneğini varsaydığını unutmayın.

```
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>
```

Aşağıdaki örnek, varolan bir veritabanı dosyasında senkronize yürütme modunu kullanarak "çalışanlar" adlı bir tablonun nasıl oluşturulacağını gösterir. Bu kodun somut olarak gösterilmiş ve bir veritabanına bağlanmış `conn` adı altında bir SQLConnection örneğini varsaydığını unutmayın.

```
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>
```

SQL veritabanı verilerini değiştirme

Adobe AIR 1.0 ve üstü

Yerel SQL veritabanlarıyla çalışırken gerçekleştirmeniz gereken bazı genel görevler vardır. Bu görevler veritabanına bağlanmayı, tablolara veri eklemeyi ve veritabanındaki tablolardan veri almayı içerir. Ayrıca, veri türleriyle çalışma ve hataları işleme gibi, bu görevleri gerçekleştirirken aklınızda bulundurmak isteyeceğiniz birkaç konu mevcuttur.

Bunun yanı sıra, daha az ilgileneceğiniz, ancak bu yaygın görevleri gerçekleştirmeden önce ilgilenmeniz gereken birkaç veritabanı görevi vardır. Örneğin, veritabanına bağlanmadan ve bir tablodan veri almadan önce, veritabanı oluşturmanız ve bu veritabanında veri yapısı oluşturmanız gerekecektir. Bu daha nadir başlangıç kurulum görevleri "Veritabanı oluşturma ve değiştirme" sayfa 726 bölümünde tartışılır.

Veritabanı işlemlerini senkronize olmayan biçimde gerçekleştirmeyi, yani veritabanı motorunun arka planda çalışmasını ve işlem başarıyla tamamlandığında veya başarısız olduğunda bir olay göndererek sizi bilgilendirmesini seçebilirsiniz. Bu işlemleri senkronize olarak da gerçekleştirebilirsiniz. Bu durumda veritabanı işlemleri ardarda gerçekleştirilir ve tüm uygulama (ekrana güncellemeler dahil) diğer kodun yürütülmesinden önce işlemlerin tamamlanmasını bekler. Senkronize olmayan ve senkronize yürütme modunda çalışma hakkında daha fazla bilgi için bkz. "Senkronize ve senkronize olmayan veritabanı işlemlerini kullanma" sayfa 761.

Veritabanına bağlanma

Adobe AIR 1.0 ve üstü

Herhangi bir veritabanı işlemini gerçekleştirmeden önce, bir veritabanı dosyası bağlantısı açın. Bir SQLConnection örneği, bir veya daha çok veritabanına bağlantıyı temsil etmek üzere kullanılır. SQLConnection örneği kullanılarak bağlanılan ilk veritabanı, "ana" veritabanı olarak bilinir. Bu veritabanına `open()` yöntemi (senkronize yürütme modu için) veya `openAsync()` yöntemi (senkronize olmayan yürütme modu için) kullanılarak bağlanılır.

Senkronize olmayan `openAsync()` işlemini kullanarak bir veritabanına bağlandığınızda, `openAsync()` işleminin ne zaman tamamlandığını bilmek için SQLConnection örneğinin `open` olayına kaydolun. İşlemin başarısız olup olmadığını belirlemek için SQLConnection örneğinin `error` olayına kaydolun.

Aşağıdaki örnek, senkronize olmayan yürütme için var olan bir veritabanı dosyasının nasıl açılacağını gösterir. Veritabanı dosyası "DBSample.db" adını taşır ve kullanıcının "Uygulama depolama dizinine işaret etme" sayfa 679 içinde bulunur.

```
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>
```

Aşağıdaki örnek, senkronize yürütme için var olan bir veritabanı dosyasının nasıl açılacağını gösterir. Veritabanı dosyası “DBSample.db” adını taşır ve kullanıcının “Uygulama depolama dizinine işaret etme” sayfa 679'inde bulunur.

```
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>
```

Senkronize olmayan örnekteki `openAsync()` yöntemi çağrısı ve senkronize örnekteki `open()` yöntemi çağrısında ikinci argümanın `SQLMode.UPDATE` sabiti olduğunu unutmayın. İkinci parametre (`openMode`) için `SQLMode.UPDATE` öğesinin belirlenmesi, belirtilen dosya mevcut değilse çalışma zamanının bir hata göndermesine neden olur. `openMode` parametresi için `SQLMode.CREATE` iletirseniz (veya `openMode` parametresini bırakırsanız), belirtilen dosya mevcut değilse çalışma zamanı bir veritabanı dosyası oluşturmaya çalışır. Ancak, dosya mevcutsa, `SQLMode.Update` kullanmanızla aynı anlama gelecek şekilde açılır. Var olan bir veritabanını salt okunur modda açmak üzere, `openMode` parametresi için `SQLMode.READ` de belirleyebilirsiniz. Bu durumda veriler veritabanından alınabilir, ancak veriler eklenemez, silinemez veya değiştirilemez.

SQL ifadeleriyle çalışma

Adobe AIR 1.0 ve üstü

Tek bir SQL ifadesi (sorgu veya komut) çalışma zamanında bir SQLStatement nesnesi olarak temsil edilir. Bir SQL ifadesi oluşturmak ve yürütmek için şu adımları takip edin:

SQLStatement örneği oluşturur.
SQLStatement nesnesi, uygulamanızdaki SQL ifadesini temsil eder.

```
var selectData:SQLStatement = new SQLStatement();
```

Veritabanının hangi sorguya karşı çalıştığını belirtin.
Bunu yapmak için, SQLStatement nesnesinin `sqlConnection` özelliğini istenen veritabanıyla bağlantısı olan SQLConnection örneğine ayarlayın.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

Geçerli SQL ifadesini belirtin.
İfade metnini bir Dize olarak oluşturun ve SQLStatement örneğinin `text` özelliğine atayın.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

Yürütme işleminin sonucunu işlemek için işlevleri tanımlayın (yalnızca senkronize olmayan yürütme modu).
İşlevleri SQLStatement örneğinin `result` ve `error` olayları olarak kaydetmek için `addEventListener()` yöntemini kullanın.

```
// 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
}
```

Alternatif olarak, Responder nesnesi kullanarak dinleyici yöntemleri belirleyebilirsiniz. Bu durumda Responder örneğini oluşturur ve dinleyici yöntemlerini bu örneğe bağlarsınız.

```
// 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
}
```

İfade metni parametre tanımları içeriyorsa, bu parametreler için değer atayın.
Parametre değerleri atamak için, SQLStatement örneğinin `parameters` ilişkilendirici dizi özelliğini kullanın.

```
selectData.parameters[":param1"] = 25;
```

SQL ifadesini yürütün.
SQLStatement örneğinin `execute()` yöntemini çağırın.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

Ayrıca, senkronize olmayan yürütme modunda olay dinleyicileri yerine Responder kullanıyorsanız, Responder örneğini `execute()` yöntemine iletin.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

Bu adımları gösteren belirli örnekler için şu başlıklara bakın:

"Bir veritabanından veri alma" sayfa 741

"Veri ekleme" sayfa 751

"Verileri değiştirme veya silme" sayfa 757

İfadelerde parametreler kullanma

Adobe AIR 1.0 ve üstü

SQL ifadesi parametresi kullanma yeniden kullanılabilir bir SQL ifadesi oluşturmanızı sağlar. İfade parametreleri kullandığınızda, ifadenin içindeki değerler değişebilir (bir `INSERT` ifadesine eklenen değerler gibi), ancak temel ifade metni aynı kalır. Sonuç olarak, parametre kullanmak uygulamanın kodlanmasını kolaylaştırmasının yanı sıra, performans açısından da fayda sağlar.

İfade parametrelerini anlama

Adobe AIR 1.0 ve üstü

Bir uygulama genellikle tek bir SQL ifadesini bir uygulamada küçük değişimlerle birçok kez kullanır. Örneğin, kullanıcının veritabanına yeni stok kalemleri ekleyebildiği bir stok izleme uygulaması düşünün. Veritabanına yeni bir stok kalemi ekleyen uygulama kodu, veriyi veritabanına gerçekten ekleyen bir SQL `INSERT` ifadesini yürütür. Ancak, ifade her yürütüldüğünde, küçük bir değişime uğrar. Özellikle, tabloya eklenen gerçek değerler farklıdır, çünkü bu değerler eklenen stok kalemine özgüdür.

İfadede farklı değerlerle birçok kez kullanılan bir SQL ifadenizin olduğu durumlarda en iyi yaklaşım, SQL metninde değişmez değerler yerine parametreler içeren bir SQL ifadesi kullanmaktır. Bir parametre, ifade her yürütüldüğünde geçerli bir değerle değiştirilen ifade metnindeki yer tutucudur. SQL ifadesinde parametreler kullanmak için, SQLStatement örneğini her zamanki gibi oluşturun. `text` özelliğine atanan geçerli SQL ifadesi için değişmez değerler yerine parametre yer tutucuları kullanın. SQLStatement örneğinin `parameters` özelliğindeki öğenin değerini ayarlayarak, her parametreye ilişkin değeri tanımlayabilirsiniz. `Parametreler`özelliği ilişkilendirici bir dizidir. Bu nedenle aşağıdaki sözdizimini kullanarak belirli bir değer ayarlarsınız:

```
statement.parameters[parameter_identifier] = value;
```

Parameter_identifier, adlandırılmış bir parametre kullanıyorsanız bir dize, adlandırılmamış bir parametre kullanıyorsanız bir tam sayı dizinidir.

Adlandırılmış parametreleri kullanma

Adobe AIR 1.0 ve üstü

Parametre, adlandırılmış bir parametre olabilir. Adlandırılmış bir parametre, veritabanının parametre değerini ifade metnindeki yer tutucu konumuna eşlemek için kullandığı belirli bir ada sahiptir. Bir parametre adı, aşağıdaki örneklerdeki gibi arkasından bir ad gelen ":" veya "@" karakterinden oluşur:

```
:itemName
@firstName
```

Aşağıdaki kod listelemesi, adlandırılmış parametrelerin kullanımını gösterir:

```
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();
```

Adlandırılmamış parametreleri kullanma

Adobe AIR 1.0 ve üstü

Adlandırılmış parametre kullanmaya alternatif olarak, adlandırılmamış parametreler de kullanabilirsiniz. Adlandırılmamış bir parametre kullanmak için, bir SQL ifadesindeki parametreyi "?" kullanarak belirtirsiniz karakter. İfadedeki parametrelerin sırasına göre, ilk parametrenin dizin 0'la başlayacağı şekilde her parametreye sayısal bir dizin atanmıştır. Aşağıdaki örnek, önceki örneğin adlandırılmamış parametre kullanılan versiyonunu gösterir:

```
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();
```

Parametre kullanmanın faydaları

Adobe AIR 1.0 ve üstü

Bir SQL ifadesinde parametreler kullanmak, çeşitli faydalar sağlar:

Daha iyi performans Parametreler kullanan bir SQLStatement örneği, her yürütüldüğünde dinamik olarak SQL metni oluşturan bir örneğe göre daha verimli çalışır. Performans iyileştirmesi, ifadenin tek bir kez hazırlanmış ve SQL ifadesinin yeniden derlenmesine ihtiyaç duymadan farklı parametre değerleri kullanarak birçok kez yürütülebiliyor olmasından kaynaklanmaktadır.

Açık veri türleme Parametreler, SQL ifadesi oluşturulduğunda bilinmeyen değerlerin türlenmiş değişikliğine izin vermek için kullanılır. Parametre kullanımı, veritabanına iletilen bir değer için depo sınıfını garanti etmenin tek yoludur. Parametreler kullanılmadığında, çalışma zamanı tüm değerleri metin temsillerinden ilişkili sütunun tür yakınlığına dayanan bir depo sınıfına dönüştürmeye çalışır.

Depolama sınıfları ve sütun benzerliğiyle ilgili daha fazla bilgi için bkz. "Veri türü desteği" sayfa 1137.

Daha fazla güvenlik Parametrelerin kullanılması, SQL yerleştirme saldırısı olarak bilinen kötü amaçlı bir tekniğin engellenmesine yardımcı olur. Kullanıcı, SQL yerleştirme saldırısında kullanıcı tarafından erişilebilir bir konuma SQL kodu girer (örneğin, veri giriş alanı). Uygulama kodu SQL ifadesini doğrudan kullanıcı girdisini SQL metnine bitiştirerek oluşturuyorsa, kullanıcı taradından girilen SQL kodu veritabanına karşı yürütülür. Aşağıdaki listeleme, kullanıcı girdisinin SQL metnine bitiştirilmesinin bir örneğini gösterir. **Bu tekniği kullanmayın**:

```
// 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;
```

Kullanıcı tarafından girilen değerleri ifade metnine bitiştirmek yerine ifade parametrelerinin kullanılması, SQL yerleştirme saldırısını engeller. Parametre değerleri değişmez ifade metninin bir parçası olmaktansa, açıkça değiştirilmiş değerler olarak değerlendirildiğinden, SQL yerleştirme gerçekleşemez. Aşağıdaki, bir önceki listelemenin önerilen bir alternatifidir:

```
// 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;
```

Bir veritabanından veri alma

Adobe AIR 1.0 ve üstü

Bir veritabanından veri alma, iki adım içerir. Önce veritabanından almak istediğiniz veri kümesini açıklayan bir SQL `SELECT` ifadesi yürütürsünüz. Daha sonra, alınan verilere erişerek bunları görüntüler ve uygulamanız için gerektirdiği biçimde değiştirirsiniz.

Bir SELECT ifadesi yürütme

Adobe AIR 1.0 ve üstü

Varolan verileri veritabanından almak için bir SQLStatement örneği kullanırsınız. Uygun SQL `SELECT` ifadesini örneğin `text` özelliğine atayın, daha sonra `execute()` yöntemini çağırın.

`SELECT` ifadesinin sözdizimiyle ilgili ayrıntılar için bkz. "yerel veritabanlarında SQL desteği" sayfa 1116.

Aşağıdaki örnek, senkronize olmayan yürütme modunu kullanan "ürünler" adlı tablodan veri almak için `SELECT` ifadesinin yürütülmesini gösterir:

```
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>
```

Aşağıdaki örnek, senkronize yürütme modunu kullanan "ürünler" adlı tablodan veri almak için `SELECT` ifadesinin yürütülmesini gösterir:

```
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>
```

Senkronize olmayan yürütme modunda ifadenin yürütülmesi tamamlandığında, SQLStatement örneği ifadenin başarıyla çalıştığını gösteren bir `result` olayı (`SQLEvent.RESULT`) gönderir. Alternatif olarak, Responder nesnesi bir argüman olarak `execute()` çağrısına iletildiyse, Responder nesnesinin sonuç işleyici özelliği çağrılır. Senkronize yürütme modunda, `execute()` işlemi tamamlanana kadar yürütme duraklar, daha sonra, sonraki kod satırında devam eder.

SELECT ifadesi sonuç verilerine erişme

Adobe AIR 1.0 ve üstü

`SELECT` ifadesinin yürütülmesi tamamlandığında, sonraki adım alınan verilere erişmektir. Sonuç verilerini, SQLStatement nesnesinin `getResult()` yöntemini çağırarak`SELECT` ifadesini yürütme yoluyla geri getirebilirsiniz.

```
var result:SQLResult = selectStatement.getResult();
```

`getResult()` yöntemi SQLResult nesnesini geri getirir. SQLResult nesnesinin `veri` özelliği `SELECT` ifadesinin sonucunu içeren bir Dizidir.

```
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];
}
```

`SELECT` sonucundaki her veri sütunu, `veri` Dizisinde yer alan bir Nesne örneğidir. Bu nesne, adları sonuç kümesindeki sütun adlarıyla eşleşen özellikler içerir. Özellikler, sonuç kümesinin sütunlarından değerler içerir. Örneğin, bir `SELECT` ifadesinin “öğeKimliği”, “öğeAdı” ve “fiyat” adlı üç sütun içeren bir sonuç kümesi belirttiğini düşünün. Sonuç kümesindeki her satır için `itemId`, `itemName`, and `price` adlı özellikleri içeren bir Object örneği oluşturulur. Bu özellikler, ilgili sütunlarından değerler içerir.

Aşağıdaki kod listelemesi, metni bir `SELECT` ifadesi olan SQLStatement örneğini tanımlar. İfade, `employees` adlı tablonun tüm satırlarının `firstName` ve`lastName` sütun değerlerini içeren satırlarını getirir. Bu örnek senkronize olmayan yürütme modunu kullanır. Yürütme tamamlandığında `selectResult()` yöntemi çağrılır, sonuçlanan veri satırlarına `SQLStatement.getResult()` kullanılarak erişilir ve bu satırlar `trace()` yöntemi kullanılarak görüntülenir. Bu listelemenin önceden başlatılmış ve bir veritabanına bağlanmış `conn` adlı bir SQLConnection örneğinin var olduğunu varsaydığını unutmayın. Ayrıca, "çalışanlar" tablosunun önceden oluşturulduğunu ve veriyle doldurulduğunu varsayar.

```
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>
```

Aşağıdaki kod listelemesi, önceki örnekteki tekniklerin aynısını senkronize yürütme modunda gösterir. Örnek, metni bir `SELECT` ifadesi olan bir SQLStatement örneğini tanımlar. İfade, `employees` adlı tablonun tüm satırlarının `firstName` ve`lastName` sütun değerlerini içeren satırlarını getirir. Sonuçlanan veri satırlarına `SQLStatement.getResult()` kullanılarak erişilir ve bu satırlar `trace()` yöntemi kullanılarak görüntülenir. Bu listelemenin önceden başlatılmış ve bir veritabanına bağlanmış `conn` adlı bir SQLConnection örneğinin var olduğunu varsaydığını unutmayın. Ayrıca, "çalışanlar" tablosunun önceden oluşturulduğunu ve veriyle doldurulduğunu varsayar.

```
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>
```

SELECT sonuç verilerinin veri türünü tanımlama

Adobe AIR 1.0 ve üstü

Varsayılan olarak, `SELECT` ifadesinin döndürdüğü her satır, sonuç kümesinin sütun adlarıyla adlandırılan özellikler ve her biri ilgili özelliğinin değerine sahip sütun değerini içeren bir Object örneği olarak oluşturulur. Ancak, bir SQL `SELECT` ifadesini yürütmeden önce SQLStatement örneğinin `itemClass` özelliğini bir sınıfa ayarlayabilirsiniz. `itemClass` özelliğinin ayarlanmasıyla, `SELECT` ifadesinin döndürdüğü her satır, belirlenen sınıfın bir örneği olarak oluşturulur. Çalışma zamanı, `SELECT` sonuç kümesindeki sütun adlarını `itemClass` sınıfındaki özellik adlarına eşleyerek özellik değerlerine sonuç sütunu değerleri atar.

`itemClass` özellik değeri olarak atanan her sınıf, herhangi bir parametre gerektirmeyen bir yapıcı içermelidir. Sınıf ayrıca, `SELECT` ifadesinin döndürdüğü her sütun için tek bir özellik içermelidir. `SELECT` listesindeki bir sütun `itemClass` sınıfında kendisiyle eşleşen bir özellik adına sahip değilse, bu bir hata olarak değerlendirilir.

SELECT sonuçlarını bölümler halinde alma

Adobe AIR 1.0 ve üstü

Varsayılan olarak, bir `SELECT` ifade yürütmesi sonuç kümesinin tüm satırlarını bir seferde alır. İfade tamamlandığında, alınan verileri genellikle nesneler oluşturma veya verileri ekranda görüntüleme gibi bir yolla işlersiniz. İfade çok sayıda satır döndürürse, tüm verilerin bir defada işlenmesi, kullanıcı arabiriminin kendisini yeniden düzenlememesine neden olacak biçimde bilgisayar için zorlayıcı olabilir.

Çalışma zamanına bir defada belirli sayıda sonuç satırı döndürmesini söyleyerek uygulamanızın algılanan performansını iyileştirebilirsiniz. Bu, ilk sonuç verilerinin daha çabuk dönmesini sağlar. Ayrıca sonuç satırlarını kümeler halinde bölümlemenizi ve böylece kullanıcı arabiriminin işlenen her satır kümesinden sonra güncellenmesini sağlar. Bu tekniği kullanmanın yalnızca senkronize olmayan modda kullanışlı olduğunu unutmayın.

`SELECT` sonuçlarını bölümler halinde almak için `SQLStatement.execute()` yönteminin ilk parametresi için bir değer belirleyin (`prefetch` parametresi). `prefetch` parametresi, ifade ilk yürütüldüğünde getirilecek satır sayısını gösterir. Bir SQLStatement örneğinin `execute()` yöntemini çağırdığınızda bir `prefetch` parametresi belirleyin, yalnızca belirlediğiniz kadar satır getirilir:

```
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
```

İfade, ilk sonuç satırı kümesinin kullanılabilir olduğunu gösteren `result` olayını gönderir. Sonuçlanan SQLResult örneğinin `data` özelliği, veri satırlarını ve getirilecek ek sonuç satırları olup olmadığını gösteren `complete` özelliğini içerir. Ek sonuç satırlarını almak için SQLStatement örneğinin `next()` yöntemini çağırın. `execute()` yöntemi gibi, `next()` yönteminin ilk parametresi, sonuç olayının sonraki gönderilişinde kaç satır alınacağını göstermek için kullanılır.

```
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);
        }
    }
}
```

`next()` yöntemi sonraki sonuç satırlarını her döndürdüğünde, SQLStatement bir `result` olayı gönderir. Sonuç olarak, tüm satırlar alınana kadar sonuçları (`next()` çağrılarından) işlemeye devam etmek için aynı dinleyici işlevi kullanılabilir.

Daha fazla bilgi için, `SQLStatement.execute()` (`prefetch` parametresinin açıklaması) ve `SQLStatement.next()` yöntemlerinin açıklamalarına bakın.

Veri ekleme

Adobe AIR 1.0 ve üstü

Bir veritabanına veri eklemede, bir SQL `INSERT` ifadesinin yürütülmesi yer alır. İfadenin yürütülmesi tamamlandığında, veritabanı tarafından oluşturulduysa, yeni eklenen satıra ilişkin ana anahtara erişebilirsiniz.

Bir INSERT ifadesini yürütme

Adobe AIR 1.0 ve üstü

Veritabanındaki bir tabloya veri eklemek için, metni SQL `INSERT` ifadesi olan bir SQLStatement örneği oluşturur ve bu örneği yürütürsünüz.

Aşağıdaki örnek, önceden var olan çalışanlar tablosuna bir veri satırı eklemek için SQLStatement örneğini kullanır. Bu örnek, senkronize olmayan yürütme modunu kullanarak veri eklemeyi gösterir. Bu listelemenin somut olarak gösterilmiş ve bir veritabanına bağlanmış `conn` adı altında bir SQLConnection örneğini varsaydığını unutmayın. Ayrıca “çalışanlar” tablosunun önceden oluşturulduğunu varsayar.

```
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>
```

Aşağıdaki örnek, senkronize yürütme modunu kullanarak önceden var olan çalışanlar tablosuna bir veri satırı ekler. Bu listelemenin somut olarak gösterilmiş ve bir veritabanına bağlanmış `conn` adı altında bir SQLConnection örneğini varsaydığını unutmayın. Ayrıca "çalışanlar" tablosunun önceden oluşturulduğunu varsayar.

```
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>
```

Eklenen bir satırın veritabanı tarafından oluşturulan ana anahtarını alma

Adobe AIR 1.0 ve üstü

Bir tabloya veri satırı eklendikten sonra kodunuz genellikle yeni eklenen satıra ilişkin veritabanı tarafından oluşturulan ana anahtarı veya satır kimlik değerini bilmek ister. Örneğin, bir tabloya satır eklediğinizde, ilgili bir tabloya satır eklemek isteyebilirsiniz. Bu durumda, ana anahtar değerini ilişkili tabloya yabancı anahtar olarak girmek istersiniz. Yeni eklenen satırın ana anahtarı, ifade yürütme tarafından oluşturulan SQLResult nesnesi kullanılarak alınabilir. Bu, `SELECT` ifadesinin yürütülmesinden sonra sonuç verilerine erişmek için kullanılan nesnenin aynısıdır. Tüm SQL ifadelerinde olduğu gibi, `INSERT` ifadesinin yürütülmesi tamamlandığında, çalışma zamanı bir SQLResult örneği oluşturur. Bir olay dinleyicisi veya senkronize yürütme modunu kullanıyorsanız SQLStatement nesnesinin `getResult()` yöntemini çağırarak SQLResult örneğine erişebilirsiniz. Alternatif olarak, senkronize olmayan yürütme modunu kullanıyorsanız ve `execute()` çağrısına bir Responder örneği iletirseniz, SQLResult örneği bir argüman olarak sonuç işleyici işlevine iletilir. Her durumda, yürütülen SQL ifadesi bir `INSERT` ifadesiyse, SQLResult örneği, en son eklenen satırın satır kimliğini içeren bir `lastInsertRowID` özelliğine sahiptir.

Aşağıdaki örnek, senkronize olmayan yürütme modunda eklenen bir satırın ana anahtarına erişmeyi gösterir:

```
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
}
```

Aşağıdaki örnek, senkronize yürütme modunda eklenen bir satırın ana anahtarına erişmeyi gösterir:

```
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
}
```

Aşağıdaki kurallara göre, satır kimliğinin tablo tanımında ana anahtar sütunu olarak belirlenen sütunun değeri olabileceğini veya olmayabileceğini unutmayın:

- Tablo, yakınlığı (sütun veri türü) `INTEGER` olan bir ana anahtar sütunuyla tanımlandıysa, `lastInsertRowID` özelliği, bu satıra eklenen değeri (veya `AUTOINCREMENT` sütunuysa, çalışma zamanı tarafından oluşturulan değeri) içerir.
- Tablo birden çok ana anahtar sütunuyla (bileşik anahtar) veya yakınlığı `INTEGER` olan tek bir ana anahtar sütunuyla tanımlandıysa, veritabanı perde arkasında satıra ilişkin tam sayı olan bir satır kimlik değeri oluşturur. Oluşturulan bu değer, `lastInsertRowID` özelliğinin değeridir.
- Değer, her zaman en son eklenen satırın satır kimliğidir. Bir `INSERT` ifadesi satır ekleyen bir tetikleyicinin çalışmasına neden olursa, `lastInsertRowID` özelliği, `INSERT` ifadesinin oluşturduğu satırın değil, tetikleyicinin eklediği son satırın satır kimliğini içerir.

Sonuç olarak, değeri `INSERT` komutundan sonra `SQLResult.lastInsertRowID` özelliği aracılığıyla kullanılabilen, açıkça tanımlanmış bir ana anahtar sütununa sahip olmak istiyorsanız, sütun bir `INTEGER PRIMARY KEY` sütunu olarak tanımlanmalıdır. Tablonuz açık bir `INTEGER PRIMARY KEY` sütunu içermese bile, ilgili tablolarla ilişkilerin tanımlanması bakımından, tablonuz için ana anahtar olarak veritabanı tarafından oluşturulmuş bir satır kimliği kullanmanın eşit derecede kabul edilebilirdir. Satır kimliği sütun değeri tüm SQL ifadelerinde `ROWID`, `_ROWID_` veya `OID` adındaki özel sütun adlarından biri aracılığıyla kullanılabilir. İlgili bir tabloda yabancı bir anahtar oluşturabilir ve

satır kimlik değerini, açıkça bildirilen `INTEGER PRIMARY KEY` sütunu için yapacağınız gibi yabancı anahtar sütun değeri olarak kullanabilirsiniz. Bu anlamda, doğan anahtar yerine rasgele bir ana anahtar kullanıyorsanız, çalışma zamanının sizin için ana anahtar değeri oluşturmasını önemsemediğiniz sürece, iki tablo arasındaki yabancı anahtar ilişkisini tanımlamak için tablonun ana anahtarı olarak `INTEGER PRIMARY KEY` sütunu veya sistem tarafından oluşturulmuş bir satır kimliği kullanmanız arasında çok az fark vardır.

Ana anahtarlar ve oluşturulan satır tanımlayıcıları için bkz. "yerel veritabanlarında SQL desteği" sayfa 1116.

Verileri değiştirme veya silme

Adobe AIR 1.0 ve üstü

Diğer veri değiştirme işlemlerinin yürütülmesi işlemi, "SQL ifadeleriyle çalışma" sayfa 737'da ele alınan SQL `SELECT` veya `INSERT` ifadelerinin yürütülmesi işlemiyle aynıdır. SQLStatement özelliğinin `text` özelliğine farklı bir SQL ifadesi koyun.

- Bir tabloda var olan verileri değiştirmek için `UPDATE` ifadesini kullanın.
- Tablodan bir veya daha çok satırı silmek için `DELETE` ifadesini kullanın.

Bu ifadelerin açıklamaları için bkz. "yerel veritabanlarında SQL desteği" sayfa 1116.

Birden çok veritabanıyla çalışma

Adobe AIR 1.0 ve üstü

Önceden açık bir veritabanı içeren SQLConnection örneğinde ek bir veritabanı bağlantısı açmak için `SQLConnection.attach()` yöntemini kullanın. `Attach()` yöntem çağrısındaki ad parametresini kullanarak, eklenen veritabanına bir ad verirsiniz. Bu veritabanını değiştirmek için ifadeler yazarken, SQL ifadenizde tablonun adlandırılmış veritabanında bulunabileceği çalışma zamanını gösteren tablo adlarını nitelemek için önekte bulunan bu adı (`database-name.table-name` biçimini kullanan) kullanabilirsiniz.

Aynı SQLConnection örneğine bağlı olan birden çok veritabanından tablolar içeren tek bir SQL ifadesi yürütebilirsiniz. Bir işlem SQLConnection örneğinde oluşturulduysa, bu işlem SQLConnection örneği kullanılarak yürütülen tüm SQL ifadelerine uygulanır. Bu, ifadenin hangi eklenmiş veritabanında çalıştığına bakılmaksızın geçerli olur.

Alternatif olarak, bir uygulamada her biri bir veya daha çok veritabanına bağlı olan birden çok SQLConnection örneği oluşturabilirsiniz. Ancak, aynı veritabanı için birden çok bağlantı kullanırsanız, veritabanı işlemlerinin SQLConnection örnekleri tarafından paylaşılmadığını aklınızda bulundurmanız gerekir. Sonuç olarak, birden çok SQLConnection örneği kullanarak aynı veritabanı dosyasına bağlanırsanız, tüm bağlantılardaki veri değişikliklerinin beklenen biçimde uygulanacağından emin olamazsınız. Örneğin, iki `UPDATE` veya `DELETE` farklı SQLConnection örnekleri aracılığıyla aynı veritabanına karşı çalışır ve işlem gerçekleştikten sonra uygulama hatası oluşursa, veritabanı verileri geri alınamayacak bir biçimde ara durumda bırakılabilir ve bu durum veritabanının bütünlüğünü (ve sonuç olarak uygulamayı) etkileyebilir.

Veritabanı hatalarını işleme

Adobe AIR 1.0 ve üstü

Veritabanı hatalarının işlenmesi genellikle diğer çalışma zamanı hatalarının işlenmesine benzer. Oluşabilecek hatalar için hazırlanmış bir kod yazmalı ve bunu yapmayı çalışma zamanına bırakmak yerine hatalara yanıt vermelisiniz. Olası veritabanı hataları genel anlamda üç kategoriye ayrılabilir: bağlantı hataları, SQL sözdizim hataları ve sınırlama hataları.

Bağlantı hataları

Adobe AIR 1.0 ve üstü

Veritabanı hatalarının çoğu bağlantı hatalarıdır ve bunlar herhangi bir işlem sırasında oluşabilir. Bunlar bağlantı hatalarını engellemeye yönelik stratejiler olsa da, veritabanı uygulamanızın çok önemli bir parçasıysa, bazen bir bağlantı hatasını düzgün biçimde gidemenin kolay bir yolu vardır.

Çoğu bağlantı hatası, çalışma zamanının işletim sistemiyle, dosya sistemiyle ve veritabanı dosyasıyla iletişim kurma biçimiyle ilgilidir. Örneğin, kullanıcının dosya sistemindeki belirli bir konumda veritabanı dosyası oluşturmak için izni yoksa, bir bağlantı hatası oluşur. Aşağıdaki stratejiler, bağlantı hatalarını engellemeye yardımcı olur:

Kullanıcıya özgü veritabanı dosyaları kullanın Tek bir bilgisayardaki uygulamayı kullanan tüm kullanıcılar için tek bir veritabanı dosyası kullanmak yerine, her kullanıcıya kendi veritabanı dosyasını verin. Dosya, kullanıcının hesabıyla ilişkilendirilmiş bir dizinde konumlandırılmalıdır. Bu örneğin, uygulama depo dizini, kullanıcının belgeler klasörü, kullanıcının masaüstü, vb. olabilir.

Farklı kullanıcı türlerini dikkate alın Uyulamanızı farklı işletim sistemlerinde farklı kullanıcı hesabı türleriyle test edin. Kullanıcının bilgisayarda yönetici iznine sahip olduğunu varsaymayın. Ayrıca, uygulamayı yükleyen kişinin, uygulamayı çalıştıran kullanıcı olduğunu varsaymayın.

Farklı dosya konumlarını dikkate alın Kullanıcının bir veritabanı dosyasını nereye kaydedeceğini veya açacağı dosyanın yerini belirlemesine izin verirseniz, kullanıcının kullanabileceği olası farklı dosya konumlarını dikkate alın. Ayrıca, kullanıcının veritabanı dosyalarını saklayacağı (veya açacağı) konumlara ilişkin sınırlar tanımlamayı düşünün. Örneğin, kullanıcıların yalnızca kullanıcı hesabının depo konumunda yer alan dosyaları açmasına izin verebilirsiniz.

Bağlantı hatası oluşursa, büyük olasılıkla ilk veritabanı oluşturma veya açma girişimi sırasında gerçekleşir. Bu, kullanıcının uygulamadaki herhangi bir veritabanı ilişkili işlemi yapamadığı anlamına gelir. Salt okunur veya izin hataları gibi belirli hata türleri için, olası kurtarma tekniklerinden biri veritabanı dosyasını farklı bir konuma kopyalamaktır. Uygulama veritabanı dosyasını kullanıcının dosya oluşturma ve yazma izninin olmadığı farklı bir konuma kopyalayabilir ve bu konumu kullanabilir.

Sözdizim hataları

Adobe AIR 1.0 ve üstü

Bir SQL ifadesi yanlış biçimde oluşturulduğunda ve uygulama ifadeyi yürütmeye çalıştığında bir sözdizim hatası oluşur. Yerel veritabanı SQL ifadeleri dizeler halinde oluşturulduğundan, derleme zamanı SQL sözdiziminin kontrol edilmesi mümkün değildir. Sözdizimlerini kontrol etmek için, tüm SQL ifadeleri yürütülmelidir. SQL sözdizim hatalarını önlemek için aşağıdaki stratejileri kullanın:

Tüm SQL ifadelerini test edin Uygulamanızı geliştirirken mümkünse SQL ifadelerinizi uygulama kodunda ifade metni olarak kodlamadan önce ayrıca test edin. Ayrıca, tüm olası seçenekleri ve koddaki değişimi deneyen bir dizi test oluşturmak için birim testi gibi bir kod testi yaklaşımını kullanın.

İfade parametrelerini kullanın ve SQL'i bitiştirmekten (dinamik olarak oluşturmaktan) kaçının Parametre kullanmak ve dinamik olarak oluşturulan SQL ifadelerinden kaçınmak, bir ifade her yürütüldüğünde aynı SQL ifade metninin kullanılması anlamına gelir. Sonuç olarak, ifadelerinizi test etmeniz ve olası değişimleri sınırlamanız çok daha kolaydır. Bir SQL ifadesini dinamik olarak oluşturmanız gerekiyorsa, ifadenin dinamik bölümlerini minimum düzeyde tutun. Ayrıca, sözdizim hatalarına neden olmayacağından emin olmak için, her kullanıcı girdisini dikkatlice doğrulayın.

Uygulamanın, bir sözdizim hatasını kurtarmak için bir SQL ifadesini incelemek ve sözdizimini düzeltmek üzere karmaşık mantığa ihtiyacı olacaktır. Kodunuz, sözdizim hatalarını önlemek için yukarıda belirtilen yönergeleri takip ederek herhangi bir olası SQL sözdizim hatası çalışma zamanı kaynağını tanımlayabilir (bir ifadede kullanılan

kullanıcı girişi gibi). Sözdizim hatalarının kurtarılması için kullanıcıya yol gösterin. İfadenin düzgün biçimde yürütülmesi için neyin düzeltilmesi gerektiğini gösterin.

Sınırlama hataları

Adobe AIR 1.0 ve üstü

Bir `INSERT` veya `UPDATE` ifadesi bir sütuna veri eklemeye çalıştığında sınırlama hataları oluşur. Yeni veriler tablo veya sütun için tanımlanan sınırlamalardan birini ihlal ediyorsa hata oluşur. Olası sınırlamalar kümesi şunları içerir:

Benzersiz sınırlama Bir tablodaki satırlar arasında, bir sütunda çift değer olamayacağını gösterir. Alternatif olarak, birden çok sütun benzersiz bir sınırlamada birleştirildiğinde, bu sütunlardaki değerlerin bileşimi çift olamaz. Diğer bir deyişle, belirtilen benzersiz sütun veya sütunlar bakımından, her satır farklı olmalıdır.

Ana anahtar sınırlaması Bir sınırlamanın izin verdiği ve vermediği veriler bakımından, ana anahtar sınırlaması benzersiz sınırlamayla aynıdır.

Null sınırlaması Tek bir sütunun `NULL` değerini saklayamayacağını ve sonuç olarak her satırda bu sütunun bir değere sahip olması gerektiğini belirtir.

Denetim sınırlaması Bir veya birden çok tablo için rasgele bir sınırlama belirtmenize olanak verir. Yaygın bir denetim sınırlaması, bir sütunun değerinin belirli sınırlar içinde olması gerektiğini tanımlayan bir kuraldır. (Örneğin, sayısal bir sütunun değeri 0'dan büyük olmalıdır.) Diğer bir yaygın denetim sınırlaması türü, sütun değerleri arasındaki ilişkileri belirtir. (Örneğin, bir sütunun değeri aynı satırdaki diğer sütunların değerinden farklı olmalıdır.)

Veri türü (sütun yakınlığı) sınırlaması Çalışma zamanı, sütun değerleri için bir veri türü belirler ve bir sütunda yanlış türde bir değeri saklama girişiminde bulunulduğunda hata oluşur. Ancak, çoğu durumda değerler sütunun belirtilen veri türüyle eşleşecek biçimde dönüştürülür. Daha fazla bilgi için bkz. "Veritabanı veri türleriyle çalışma" sayfa 760.

Çalışma zamanı, yabancı anahtar değerleri üzerinde sınırlama uygulamaz. Başka bir deyişle, yabancı anahtar değerlerinin var olan bir ana anahtar değeriyle eşleşmesi gerekmez.

Önceden tanımlı sınırlama türlerinin yanı sıra, çalışma zamanı SQL motoru, tetikleyicilerin kullanımını destekler. Tetikletici, olay işleyiciye benzer. Belirli bir eylem gerçekleştiğinde yürütülen, önceden tanımlı bir talimatlar kümesidir. Örneğin, bir tetikleyici belirli bir tabloya veri eklendiğinde veya bu tablodan veri silindiğinde çalışacak biçimde tanımlanabilir. Olası bir tetikleyici kullanımı, tetikleyicinin veri değişikliklerini incelemesi ve belirlenen koşullar karşılanmıyorsa bir hataya neden olmasıdır. Sonuç olarak tetikleyici, sınırlamayla aynı amaca hizmet edebilir ve sınırlama hatalarının önlenmesi ve kurtarılmasına ilişkin stratejiler tetikletici tarafından oluşturulan hatalar için de geçerlidir. Ancak, tetikleyici tarafından oluşturulan hatalara ilişkin hata kimliği, sınırlama hatalarına ilişkin hata kimliğinden farklıdır.

Belirli bir tabloya uygulanan sınırlama kimesi, uygulamanızı tasarladığınız sırada belirlenir. Sınırlamaların bilinçli şekilde tasarlanması, uygulamanızı sınırlama hatalarının önlenmesi ve kurtarılmasına yönelik tasarlamanızı kolaylaştırır. Ancak, sınırlama hatalarının sistematik olarak tahmin edilmesi ve önlenmesi zordur. Sınırlama hataları uygulama verileri eklenene kadar görünmediğinden, tahmin etmek zordur. Sınırlama hataları, oluşturulduktan sonra veritabanına eklenen verilerle oluşur. Bu hatalar genellikle yeni veriler ve veritabanında önceden bulunan veriler arasındaki ilişkinin bir sonucudur. Aşağıdaki stratejiler, birçok sınırlama hatasının önlenmesine yardımcı olur:

Veritabanı yapısını ve sınırlamaları dikkatlice planlayın Sınırlamaların amacı uygulama kuralları koymak ve veritabanı verilerinin bütünlüğünü korumaya yardımcı olmaktır. Uygulamanızı planlarken, veritabanınızı uygulamanızı desteklemesi açısından nasıl yapılandırmanız gerektiğini düşünün. Bu işlemin bir parçası olarak, verilerinize ilişkin belirli değerlerin gerekli olup olmadığı, bir değerin varsayılanı olup olmadığı, çift değerlerin izinli olup olmadığı, vb. kuralları tanımlayın. Bu kurallar, veritabanı sınırlamalarını tanımlamanızda size rehberlik eder.

Sütun adlarını açıkça belirtin `INSERT` ifadesi, değerlerin ekleneceği sütunları açık biçimde belirtmeden yazılabilir, ancak böyle yapmak, gerekli olmayan bir risktir. Değerlerin ekleneceği sütunları açık biçimde adlandırarak otomatik

olarak oluşturulan değerlere, varsayılan değerler içeren sütunlara ve `NULL` değerlerine izin veren sütunlara olanak tanıyabilirsiniz. Ayrıca, bunu yaparak tüm `NOT NULL` sütunlarına açık biçimde bir değerin eklenmesini garanti edebilirsiniz.

Varsayılan değerler kullanın Bir sütun için `NOT NULL` sınırlamasını belirlediğinizde, mümkünse sütun tanımında varsayılan bir değer belirleyin. Uygulama kodu da varsayılan değerler sağlayabilir. Örneğin, kodunuz bir String değişkeninin `null` olup olmadığını kontrol edebilir ve bir ifade parametre değeri ayarlamak için kullanmadan önce bu değişkene bir değer atayabilir.

Kullanıcı tarafından girilen verileri doğrulayın Özellikle `NOT NULL` ve `CHECK` sınırlamalarının olduğu durumlarda kullanıcı tarafından girilen verilerin sınırlamalar tarafından belirtilen sınırlara uyduğundan emin olmak için bu verileri önceden kontrol edin. Doğal olarak, bir `UNIQUE` sabitinin kontrol edilmesi daha zordur, çünkü bu kontrol verilerin benzersiz olup olmadığını belirlemek için bir `SELECT` sorgusunun yürütülmesini gerektirir.

Tetikleyiciler kullanın Eklenen verileri doğrulayan (ve değiştirebilen) veya geçersiz verileri düzeltmek için diğer eylemleri gerçekleştiren bir tetikleyici yazabilirsiniz. Bu doğrulama ve düzeltme, bir sınırlama hatasının oluşmasını önleyebilir.

Sınırlama hatalarının engellenmesi, birçok bakımdan diğer türdeki hataların engellenmesinden daha zordur. Yine de, uygulamayı istikrarsız ve kullanılamaz bir duruma getirmeden bu hataları kurtarmaya yönelik birkaç strateji vardır:

Çakışma algoritmaları kullanın Bir sütunda sınırlama tanımladığınızda ve bir `INSERT` veya `UPDATE` ifadesi oluşturduğunuzda, bir çakışma algoritması belirleme seçeneğine sahip olursunuz. Çakışma algoritması, bir sınırlama ihlali gerçekleştiğinde veritabanının yapacağı eylemi tanımlar. Veritabanı motorunun gerçekleştirebileceği birkaç olası eylem vardır. Veritabanı motoru tek bir ifadeyi veya tüm işlemi sonlandırabilir. Hatayı yoksayabilir. Eski verileri kaldırıp bunları kodun saklamaya çalıştığı verilerle değiştirebilir.

Daha fazla bilgi için, “yerel veritabanlarında SQL desteği” sayfa 1116 bölümünde bulunan “ON CONFLICT (çakışma algoritmaları)” bölümüne bakın.

Düzeltici geribildirim sağlayın Belirli bir SQL komutunu etkileyebilecek sınırlamalar kümesi önceden tanımlanabilir. Sonuç olarak, bir ifadenin neden olabileceği sınırlama hatalarını tahmin edebilirsiniz. Bu bilgiyle, bir sınırlama hatasına yanıt vermek için uygulama mantığı oluşturabilirsiniz. Örneğin, bir uygulamanın yeni ürünlerin girişi için bir veri giriş formu içerdiğini varsayın. Veritabanındaki ürün adı sütunu `UNIQUE` sınırlamasıyla tanımlanıyorsa, veritabanına yeni bir ürün satırı ekleme eylemi bir sınırlama hatasına neden olabilir. Sonuç olarak, uygulama bir sınırlama hatasını tahmin edecek biçimde tasarlanmıştır. Hata gerçekleştiğinde, uygulama kullanıcıyı belirtilen ürün adının zaten kullanılmakta olduğunu göstererek ve farklı bir ürün adı seçmesini isteyerek kullanıcıyı uyarır. Olası başka bir yanıt, kullanıcının aynı ada sahip diğer ürün hakkındaki bilgileri görüntülemesine izin vermektir.

Veritabanı veri türleriyle çalışma

Adobe AIR 1.0 ve üstü

Bir veritabanında tablo oluşturulduğunda, tablo oluşturmaya ilişkin SQL ifadesi tablodaki her sütun için yakınlığı veya veri türünü tanımlar. Yakınlık bildirimleri göz ardı edilebilse de, sütun yakınlığını `CREATE TABLE` SQL ifadenizde açık biçimde belirtmek iyi bir fikirdir.

Genel bir kural olarak, `INSERT` ifadesini kullanarak veritabanında sakladığınız her nesne, `SELECT` ifadesini yürüttüğünüzde aynı veri türünün bir örneği olarak döndürülür. Ancak, alınan değerin veri türü, verinin saklandığı veritabanı sütununun yakınlığına bağlı olarak değişik olabilir. Değer bir sütunda saklandığında, veri türü sütunun yakınlığıyla eşleşmiyorsa, veritabanı değeri sütunun yakınlığıyla eşleşecek biçimde dönüştürmeye çalışır. Örneğin, bir veritabanı sütunu için `NUMERIC` yakınlığı bildirildiyse, veritabanı verileri saklamadan önce girilen veriyi sayısal saklama sınıfına (`INTEGER` veya `REAL`) dönüştürmeye çalışır. Veri dönüştürülemezse, veritabanı bir hata verir. Bu kurala göre, "12345" Dizesi bir `NUMERIC` sütununa eklendiğinde, veritabanı bu dizeyi saklamadan önce otomatik olarak 12345 tam sayı değerine dönüştürür. Veri bir `SELECT` ifadesi kullanılarak alındığında, değer bir Dize örneği olarak değil, bir sayısal veri türü (örneğin Numara) örneği olarak döndürülür.

İstenmeyen veri türü dönüştürmesini önlemenin en iyi yolu iki kuralı takip etmektir. Öncelikle, her sütunu saklanması hedeflenen veri türüyle eşleşecek yakınlıkla tanımlayın. Daha sonra, yalnızca veri türü tanımlanan yakınlıkla eşleşen değerleri ekleyin. Bu kuralların takip edilmesi, iki fayda sağlar. Veri eklediğinizde, veriler beklenmedik şekilde (büyük olasılıkla sonuç olarak hedeflenen anlamını kaybedecek biçimde) dönüştürülmez. Ayrıca, verileri almak istediğinizde, veriler orijinal veri türünde döndürülür.

Mevcut sütun benzerliği türleri ve SQL ifadelerinde veri türlerini kullanmayla ilgili daha fazla bilgi için bkz. "Veri türü desteği" sayfa 1137.

Senkronize ve senkronize olmayan veritabanı işlemlerini kullanma

Adobe AIR 1.0 ve üstü

Önceki bölümler verilerin alınması, eklenmesi, güncellenmesi ve silinmesi gibi yaygın veritabanı işlemlerinin yanı sıra, bir veritabanı içinde veritabanı dosyası, tablolar ve diğer nesnelerin oluşturulmasını açıkladı. Örnekler bu işlemlerin senkronize olmayan ve senkronize biçimlerde nasıl gerçekleştirileceğini gösterdi.

Anımsatıcı olarak, senkronize olmayan yürütme modunda veritabanı motoruna bir işlem gerçekleştirmesini bildirirsiniz. Daha sonra, uygulama çalışmaya devam ederken veritabanı motoru arka planda çalışır. İşlem tamamlandığında, veritabanı motoru sizi bu konuda uyarmak için bir olay gönderir. Senkronize olmayan yürütmenin temel faydası, ana uygulama kodu yürütülmeye devam ederken çalışma zamanının veritabanı işlemlerini arka planda gerçekleştirmesidir. Bu özellikle, bir işlem önemli ölçüde zaman alıyorsa değerli bir faydadır.

Diğer taraftan, senkronize yürütme modu işlemleri arka planda çalışmaz. Veritabanı motoruna bir işlem gerçekleştirmesini söylersiniz. Veritabanı motoru işini yaparken, kod bu noktada duraklar. İşlem tamamlandığında, yürütme sonraki kod satırıyla devam eder.

Tek bir veritabanı bağlantısı, bazı işlemler ve ifadeleri senkronize, diğerlerini senkronize olmayan biçimde yürütemez. Veritabanı bağlantısını açtığınızda bir SQLConnection bağlantısının senkronize veya senkronize olmayan modda çalıştığını belirtmelisiniz. `SQLConnection.open()` yöntemini çağırırsanız, bağlantı senkronize yürütme modunda, `SQLConnection.openAsync()` yöntemini çağırırsanız senkronize olmayan yürütme modunda çalışır. SQLConnection örneği `open()` veya `openAsync()` kullanarak bir veritabanına bağlandığında, senkronize veya senkronize olmayan yürütmeye sabitlenir.

Senkronize veritabanı işlemlerini kullanma

Adobe AIR 1.0 ve üstü

Senkronize yürütmeyi kullanırken işlemleri yürütmek ve yanıtlamak için kullandığınız geçerli kodla, senkronize olmayan yürütme modunda kullandığınız kod arasında çok az fark vardır. İki yaklaşım arasındaki temel farklar iki alanda incelenebilir. Birincisi, başka bir işleme bağlı olan bir işlem yürütmektir (`SELECT` sonuç satırları veya bir `INSERT` ifadesi tarafından eklenen satırın ana anahtarı gibi). İkinci fark alanı, hataların işlenmesidir.

Senkronize işlemler için kod yazma

Adobe AIR 1.0 ve üstü

Senkronize ve senkronize olmayan işlemler arasındaki temel fark, senkronize modda kodun tek bir adım serisi olarak yazılmasıdır. Senkronize olmayan kodda ise, olay dinleyicileri kaydeder ve işlemleri genellikle dinleyici yöntemleri arasında bölümlersiniz. Bir veritabanına senkronize yürütme modunda bağlanıldığında, tek bir kod bloğu içinde art arda bir veritabanı işlemleri serisi yürütebilirsiniz. Aşağıdaki örnek bu tekniği göstermektedir:

```
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();
```

Gördüğünüz gibi, senkronize veya senkronize olmayan yürütme kullandığınızda veritabanı işlemlerini gerçekleştirmek için aynı yöntemleri çağırırsınız. İki yaklaşım arasındaki temel fark, başka bir işleme bağlı olan bir işlem yürütmek ve hataların işlenmesidir.

Başka bir işleme bağlı olan bir işlem yürütme

Adobe AIR 1.0 ve üstü

Senkronize yürütme modunu kullandığınızda, işlemin tamamlandığını belirlemek için bir olayı dinleyen kod yazmanız gerekmez. Bunun yerine, bir kod satırındaki işlem başarıyla tamamlandığında yürütmenün bir sonraki kod satırıyla devam ettiğini varsayabilirsiniz. Sonuç olarak, başka bir işlemin başarısına bağlı olan bir işlemi gerçekleştirmek için, yalnızca bağlı olduğu işlemin hemen arkasından gelen bağımlı kodu yazmalısınız. Örneğin, bir uygulamayı bir işlem başlatmak üzere kodlamak için `INSERT` ifadesini yürütün, eklenen satırın ana anahtarını alın, bu ana anahtarı farklı bir tablodaki başla bir satıra ekleyin ve son olarak işlemi tamamlayın, kod bir ifadeler serisi olarak yazılabilir. Aşağıdaki örnek bu işlemleri gösterir:

```
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();
```

Hataları senkronize yürütme ile işleme

Adobe AIR 1.0 ve üstü

Senkronize yürütme modunda, bir işlemin başarısız olduğunu belirlemeniz için herhangi bir hata olayını dinlemeniz gerekmez. Bunun yerine, bir `try..catch..finally` kod blokları serisinde hataları tetikleyebilecek olan herhangi bir kodu sararsınız. `Try` bloğunda hata veren kodu sararsınız. Ayrı `catch` bloklarındaki her hata türüne yanıt olarak gerçekleştirilecek eylemleri yazın. Başarı veya hatadan bağımsız olarak her zaman yürütülmesini istediğiniz herhangi bir kodu (örneğin, artık gerek duyulmayan bir veritabanı bağlantısını kapatma) bir `finally` bloğuna yerleştirin. Aşağıdaki örnek hata işleme için `try..catch..finally` bloklarının kullanımını gösterir. Önceki örneği hata işleme kodu ekleyerek devam ettirir:

```
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();
}
```

Senkronize olmayan yürütme modelini anlama

Adobe AIR 1.0 ve üstü

Senkronize olmayan yürütme modunun kullanımına ilişkin yaygın bir kanı, aynı veritabanı bağlantısına karşı başka bir SQLStatement örneğini başlatmayacak olmanıza ilişkin varsayımdır. Gerçekte, bu varsayım doğru değildir. SQLStatement örneği yürütülürken ifadenin `text` özelliğini değiştiremezsiniz. Ancak, yürütmek istediğiniz her farklı SQLStatement örneği için ayrı bir SQLStatement örneği kullanırsanız, başka bir SQLStatement örneği hala yürütülüyorken hataya neden olmadan `execute()` yöntemini çağırabilirsiniz.

Dahili olarak, veritabanı işlemlerimi senkronize olmayan yürütme modunu kullanarak yürüttüğünüzde, her veritabanı bağlantısı (her SQLConnection örneği) gerçekleştirme talimatını verdiğiniz kendi işlem kuyruğu veya listesine sahiptir. Çalışma zamanı her işlemi kuyruğa eklendikleri sırayla yürütür. Bir SQLStatement örneği oluşturduğunuzda ve bu örneğin `execute()` yöntemini çağırdığınızda, ifade yürütme işlemi bağlantı için kuyruğa eklenir. Bu SQLConnection örneğinde o anda yürütülen bir işlem yoksa, ifade arka planda yürütülmeye başlar. Kodla aynı blok içinde başka bir SQLStatement örneği oluşturduğunuzu ve bu yöntemin `execute()` yöntemini de çağırdığınızı varsayın. Bu ikinci ifade yürütme işlemi, kuyrukta ilk ifadenin arkasına eklenir. İlk ifadenin yürütülmesi tamamlanır tamamlanmaz, çalışma zamanı kuyruktaki bir sonraki işleme geçer. Ana uygulama kodunda ilk işleme ilişkin `result` olayının gönderildiği sırada bile, kuyrukta yer alan sonraki işlemlerin yürütülmesi arka planda devam eder. Aşağıdaki kod bu tekniği gösterir:

```
// 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.
```

Ardarda kuyruğa alınan ifadeleri otomatik olarak yürüten veritabanının önemli bir yan etkisi vardır. Bir ifade başka bir işlemin sonucuna bağlıysa, ilk işlem tamamlanana kadar bu ifadeyi kuyruğa ekleyemezsiniz. (Başka bir deyişle, bu ifadenin `execute()` yöntemini çağıramazsınız.) Çünkü ikinci ifadenin `execute()` yöntemini bir kez çağırdığınızda, ifadenin `text` veya `parameters` özelliklerini değiştiremezsiniz. Bu durumda sonraki işlemi başlatmak için, ilk işlemin tamamlandığını gösteren olayı beklemeniz gerekir. Örneğin, işlem bağlamında bir ifade yürütmek istiyorsanız, ifadenin yürütülmesi işlemi açma işlemine bağlıdır. İşlemi açmak için `SQLConnection.begin()` yöntemini çağırdıktan sonra SQLConnection örneğinin `begin` olayını göndermesini beklemeniz gerekir. Daha sonra SQLStatement örneğinin `execute()` yöntemini çağırabilirsiniz. Bu örnekte uygulamayı işlemlerin düzgün biçimde yürütülmesini sağlayacak biçimde düzenlemenin en basit yolu, `begin` olayı için dinleyici olarak kaydedilen bir yöntem oluşturmaktır. `SQLStatement.execute()` yöntemini çağıracak kod, bu dinleyici yönteminin içine yerleştirilir.

Şifrelemeyi SQL veritabanlarıyla kullanma

Adobe AIR 1.5 ve üstü

Tüm Adobe AIR uygulamaları aynı yerel veritabanı motorunu paylaşır. Sonuç olarak, tüm AIR uygulamaları şifrelenmemiş bir veritabanı dosyasına bağlanabilir, bu dosyadan okuyabilir ve bu dosyaya yazabilir. Adobe AIR 1.5'ten başlayarak, AIR şifreli veritabanı dosyaları oluşturma ve bu dosyalara bağlanma özelliğini içerir. Veritabanına bağlanmak için şifreli bir veritabanı kullandığınızda, uygulama doğru şifreleme anahtarını sağlamalıdır. Yanlış şifreleme anahtarı sağlandığında (veya anahtar sağlanmadığında), uygulama veritabanına bağlanamaz. Sonuç olarak, uygulama verileri veritabanından okuyamaz, veritabanına veri yazamaz veya verileri değiştiremez.

Şifreli bir veritabanı kullanmak için, veritabanını şifreli bir veritabanı olarak oluşturmalısınız. Var olan şifreli bir veritabanıyla, veritabanına bağlantı açabilirsiniz. Ayrıca, şifreli bir veritabanının şifreleme anahtarını değiştirebilirsiniz. Şifreli veritabanları oluşturma ve bu veritabanlarına bağlanmanın dışında, şifreli bir veritabanıyla çalışma teknikleri, şifrelenmemiş bir veritabanıyla çalışma tekniklerinin aynısıdır. Özellikle, veritabanının şifreli olup olmamasına bakılmaksızın, SQL ifadeleri yürütme teknikleri aynıdır.

Yerel SQL veritabanları için kullanımlar

Adobe AIR 1.5 ve üstü

Şifreleme, veritabanında saklanan bilgiye erişimi sınırlamak istediğinizde kullanışlı bir yoldur. Adobe AIR'in veritabanı şifreleme işlevleri farklı amaçlar için kullanılabilir. Şifreli bir veritabanı kullanmak isteyebileceğiniz durumlara ilişkin örnekler aşağıdadır:

- Bir sunucudan indirilen özel uygulama verilerinin salt okunur önbelleği
- Bir sunucuyla senkronize edilen özel veriler için yerel bir uygulama deposu (sunucuya veri gönderilir ve sunucudan veri yüklenir)
- Uygulama tarafından oluşturulan ve düzenlenen belgeler için dosya formatı olarak kullanılan şifreli dosyalar. Dosyalar bir kullanıcıya özgü olabilir veya uygulamanın tüm kullanıcıları tarafından paylaşılmak üzere tasarlanmış olabilir.
- "Yerel SQL veritabanları için kullanımlar" sayfa 722 bölümünde açıklananlar gibi bilgisayara veya veritabanı dosyalarına erişimi olan insanlardan saklı tutulması gereken diğer yerel veri deposu kullanımları.

Şifreli bir veritabanı kullanmak istemenizin nedenini anlamak, uygulamanızı nasıl yapılandıracağınıza karar vermenize yardımcı olur. Özellikle, uygulamanızın veritabanı için şifreleme anahtarını nasıl oluşturacağını, alacağını veya depolayacağını etkileyebilir. Dikkate alınması gereken bu konular hakkında daha fazla bilgi için bkz. "Şifrelemeyi veritabanıyla kullanmak için dikkate alınması gerekenler" sayfa 770.

Şifreli veritabanı dışında, kritik verilerin gizli kalmasını sağlamada kullanılan alternatif bir mekanizma, şifrelenmiş yerel depo.'dur. Şifrelenmiş yerel depoyla, String anahtarı kullanarak ByteArray değeri saklarsınız. Yalnızca değeri saklayan AIR uygulaması buna erişebilir ve yalnızca değerin saklandığı bilgisayarda erişebilir. Şifreli yerel depoyla, kendi şifreleme anahtarınızı oluşturmanız gerekmez. Bu nedenle, şifreli yerel depo bir ByteArray içinde kolayca kodlanabilen tek bir değer veya değerler kümesini kolayca saklamak için en uygun yoldur. Şifreli veritabanı, yapılandırılmış veri depolama ve sorgulamanın amaçlandığı daha büyük veri kümeleri için daha uygundur. Şifreli yerel depo kullanımı hakkında daha fazla bilgi için, bkz. "Şifrelenmiş yerel depolama" sayfa 718.

Şifreli veritabanı oluşturma

Adobe AIR 1.5 ve üstü

Şifreli bir veritabanı kullanmak için, veritabanı dosyası oluşturulduğu sırada şifrelenmelidir. Veritabanı bir kez şifrelenmemiş olarak oluşturulduğunda, sonradan şifrelenemez. Benzer şekilde, şifreli bir veritabanının daha sonra şifresi kaldırılamaz. Ayrıca, şifreli bir veritabanının şifreleme anahtarını değiştirebilirsiniz. Ayrıntılar için bkz. "Veritabanının şifreleme anahtarını değiştirme" sayfa 769. Şifrelenmemiş bir veritabanınız varsa ve veritabanı şifrelemeyi kullanmak istiyorsanız, yeni ve şifreli bir veritabanı oluşturabilir ve var olan tablo yapısını ve verileri yeni veritabanına kopyalayabilirsiniz.

Şifreli bir veritabanı oluşturmak, "Veritabanı oluşturma" sayfa 727 bölümünde anlatıldığı gibi, şifrelenmemiş bir veritabanı oluşturmakla hemen hemen aynıdır. Önce veritabanına bağlantıyı temsil eden birSQLConnection örneği oluşturuyorsunuz. Henüz var olmayan bir dosya için veritabanı konumunu belirterek, SQLConnection nesnesinin `open()` yöntemini veya `openAsync()` yöntemini çağırma yoluyla veritabanını oluşturabilirsiniz. Şifreli veritabanı oluşturmanın tek farkı, `encryptionKey` parametresi için (`open()` yönteminin beşinci parametresi ve `openAsync()` yönteminin altıncı parametresi) bir değer sağlamanız gerekmesidir.

Geçerli bir `encryptionKey` parametre değeri, tam olarak 16 bayt içeren bir ByteArray nesnesidir.

Aşağıdaki örneklerde şifrelenmiş veritabanı oluşturma gösterilmiştir. Basitleştirmek için, bu örnekte şifreleme anahtarı uygulama kodunda sabit olarak kodlanmıştır. Ancak, bu teknik güvenli olmadığı için kesinlikle önerilmez.

```
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);
```

Şifreleme anahtarı oluşturmak için önerilen bir yolu gösteren örnek için bkz. "Örnek: Şifreleme anahtarı oluşturma ve kullanma" sayfa 771.

Şifreli bir veritabanına bağlanma

Adobe AIR 1.5 ve üstü

Şifreli veritabanı oluşturma gibi, şifreli bir veritabanına bağlantı açma yordamı da şifrelenmemiş bir veritabanına bağlanmaya benzer. Bu yordam, "Veritabanına bağlanma" sayfa 734 bölümünde daha ayrıntılı biçimde açıklanmıştır. Senkronize yürütme modunda bir bağlantı açmak için`open()` yöntemini veyasenkronize olmayan yürütme modunda bağlantı açmak için`openAsync()` yöntemini kullanın. Aradaki tek fark, şifreli bir veritabanını açmak için `encryptionKey` parametresine ilişkin (`open()` yönteminin beşinci parametresi ve `openAsync()` yönteminin altıncı parametresi) doğru değeri belirtmeniz gerekiyor olmasıdır.

Sağlanan şifreleme anahtarı doğru değilse, bir hata oluşur. `open()` yöntemi için, SQLError istisnası verilir. SQLConnection nesnesi, `openAsync()` yöntemi için, `error` özelliği bir SQLError nesnesi içeren bir SQLErrorEvent gönderir. Her durumda, istisna tarafından oluşturulan SQLError nesnesi `errorID` özellik değeri 3138'e sahiptir. Bu hata kimliği, "Açılan dosya bir veritabanı dosyası değil" hata mesajına karşılık gelir.

Aşağıdaki örnek, senkronize olmayan yürütme modunda şifreli veritabanı açmayı gösterir. Basitleştirmek için, bu örnekte şifreleme anahtarı uygulama kodunda sabit olarak kodlanmıştır. Ancak, bu teknik güvenli olmadığı için kesinlikle önerilmez.

```
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);
    }
}
```

Aşağıdaki örnek, senkronize yürütme modunda şifreli veritabanı açmayı gösterir. Basitleştirmek için, bu örnekte şifreleme anahtarı uygulama kodunda sabit olarak kodlanmıştır. Ancak, bu teknik güvenli olmadığı için kesinlikle önerilmez.

```
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);
    }
}
```

Şifreleme anahtarı oluşturmak için önerilen bir yolu gösteren örnek için bkz. “Örnek: Şifreleme anahtarı oluşturma ve kullanma” sayfa 771.

Veritabanının şifreleme anahtarını değiştirme

Adobe AIR 1.5 ve üstü

Veritabanı şifrelendiğinde, veritabanına ilişkin şifreleme anahtarını sonradan değiştirebilirsiniz. Veritabanının şifreleme anahtarını değiştirmek için önce bir SQLConnection örneği oluşturarak ve bunun `open()` veya `openAsync()` yöntemini çağırarak veritabanına bağlantı açın. Veritabanına bağlanıldığında, argüman olarak yeni şifreleme anahtarını ileterek `reencrypt()` yöntemini çağırın.

Çoğu veritabanı işlemi gibi, `reencrypt()` yönteminin davranışı, veritabanı bağlantısının senkronize veya senkronize olmayan yürütme modunu kullanıyor olmasına göre değişir. Veritabanına bağlanmak için `open()` yöntemini kullanırsanız `reencrypt()` işlemi senkronize olarak çalışır. İşlem tamamlandığında, yürütme sonraki kod satırıyla devam eder:

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

Diğer yandan, veritabanı bağlantısı `openAsync()` yöntemi kullanılarak açıldıysa, `reencrypt()` işlemi senkronize olmayan modda çalışır. `reencrypt()` yönteminin çağrılması, yeniden şifreleme işlemini başlatır. İşlem tamamlandığında, SQLConnection nesnesi bir `reencrypt` olayı gönderir. Yeniden şifrelemenin tamamlandığını belirlemek için bir olay dinleyicisi kullanabilirsiniz:

```
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
}
```

`reencrypt()` işlemi, kendi işlemi içinde çalışır. İşlem kesintiye uğradığında veya başarısız olduğunda (örneğin, uygulama işlem tamamlanmadan kapatıldığında) işlem geri alınır. Bu durumda, ilk şifreleme anahtarı veritabanının şifreleme anahtarı olarak kalır.

`reencrypt()` yöntemi, veritabanından şifrelemeyi kaldırmak için kullanılamaz. `reencrypt()` yöntemine `null` değerinin veya 16 bayt olmayan bir ByteArray değerinin iletilmesi hatayla sonuçlanır.

Şifrelemeyi veritabanıyla kullanmak için dikkate alınması gerekenler

Adobe AIR 1.5 ve üstü

"Yerel SQL veritabanları için kullanımlar" sayfa 766 bölümü, şifreli veritabanı kullanmak isteyebileceğiniz çeşitli durumları gösterir. Farklı uygulamaların kullanım senaryolarının (bunlar ve diğer senaryolar dahil) farklı gizlilik gereksinimleri olduğu açıktır. Uygulamanızda şifreleme kullanımını yapılandırma biçiminiz, bir veritabanındaki verilerin ne kadar özel olduğunu denetlemenizin önemli bir parçasıdır. Örneğin, şifreli veritabanını kişisel verilerinizi aynı bilgisayarı kullanan diğer kullanıcılardan bile saklı tutmak için kullanıyorsanız, her kullanıcının veritabanı kendi şifreleme anahtarını gerektirir. En yüksek düzeyde güvenlik için, uygulamanız kullanıcı tarafından girilen paroladan anahtar oluşturabilir. Şifreleme anahtarını parolaya dayandırmak, başka bir kişi kullanıcının bilgisayardaki hesabına erişmeyi başarsa bile, verilere erişememesini sağlar. Güvenlik spektrumunun diğer ucunda, bir veritabanı dosyasının uygulamanızın tüm kullanıcıları için okunabilir olmasını, ancak diğer uygulamaların bu dosyaya erişememesini istiyorsunuz. Bu durumda, uygulamanın yüklenmiş her kopyasının, paylaşımlı bir şifreleme anahtarına erişimi olması gerekir.

Uygulamanızı ve özellikle şifreleme anahtarının oluşturulduğu tekniği uygulama verileriniz için kullanmak istediğiniz gizlilik düzeyine göre tasarlayabilirsiniz. Aşağıdaki liste, farklı veri gizliliği düzeyleri için tasarım önerileri sunar:

- Veritabanını herhangi bir bilgisayardaki uygulamaya erişimi olan tüm kullanıcılar için erişilebilir kılmak için uygulamanın tüm örnekleri tarafından kullanılabilen tek bir anahtar kullanın. Örneğin, uygulama ilk kez çalıştırıldığında, SSL gibi güvenli bir protokol kullanarak bir sunucudan paylaşımlı şifreleme anahtarını indirebilir. Anahtarı gelecekte kullanmak üzere şifreli yerel depoya kaydedebilir. Alternatif olarak, bilgisayardaki her kullanıcıya ilişkin verileri şifreleyin ve verileri taşınabilir kılmak için sunucu gibi uzak bir veri deposuyla senkronize edin.
- Veritabanını herhangi bir bilgisayardaki tek bir kullanıcı tarafından erişilebilir kılmak için, şifreleme anahtarını bir kullanıcı sırrından (parola gibi) oluşturun. Anahtarı oluşturmak için belirli bir bilgisayara bağlı olan herhangi bir değeri (şifreli yerel depoda saklanan bir değer gibi) kullanmayın. Alternatif olarak, bilgisayardaki her kullanıcıya ilişkin verileri şifreleyin ve verileri taşınabilir kılmak için sunucu gibi uzak bir veri deposuyla senkronize edin.
- Veritabanını tek bir bilgisayardaki tek bir kişi tarafından erişilebilir kılmak için, anahtarı bir paroladan ve oluşturulmuş bir salttan yaratın. Bu tekniğe ilişkin bir örnek için bkz. "Örnek: Şifreleme anahtarı oluşturma ve kullanma" sayfa 771.

Bir uygulamayı şifreli bir veritabanı kullanmak üzere tasarlarken göz önüne alınması gereken diğer güvenlik konuları şunlardır:

- Bir sistem her zaman en zayıf noktası kadar güvenlidir. Şifreleme anahtarı kullanmak için kullanıcı tarafından girilen bir parolayı kullanıyorsanız, parolalarda minimum uzunluk ve karmaşıklık sınırlamaları uygulamayı göz önünde bulundurun. Yalnızca temel karakterleri kullanan kısa bir parola, kolayca tahmin edilebilir.
- AIR uygulamasının kaynak kodu, kullanıcının bilgisayarında düz metinde (HTML içeriği için) veya derlemesi kolayca açılabilir bir ikili formatta (SWF içeriği için) saklanır. Kaynak kod erişilebilir olduğundan, hatırlanması gereken iki nokta şöyledir:
  - Şifreleme anahtarınızı kaynak kodunuzda asla sabit olarak kodlamayın
  - Şifreleme anahtarını oluşturmak için kullanılan tekniğin (rastgele karakter oluşturucu veya belirli bir karma algoritma) bir saldırgan tarafından kolayca çözülebileceğini her zaman göz önünde bulundurun
- AIR veritabanı şifreleme CBC-MAC (CCM) moduyla Counter içeren Gelişmiş Şifreleme Standardı'nı (AES) kullanır. Bu şifreleme, güvenli olmak için bir salt değeriyle birleştirilen kullanıcı tarafından girilmiş bir anahtarı gerektirir. Buna ilişkin bir örnek için bkz. "Örnek: Şifreleme anahtarı oluşturma ve kullanma" sayfa 771.
- Veritabanını şifrelemeyi seçtiğinizde, bu veritabanıyla bağlantılı olan veritabanı motoru tarafından kullanılan tüm disk dosyaları şifrelenir. Ancak, veritabanı motoru işlemlerdeki okuma ve yazma süresi performansını artırmak için bazı verileri geçici olarak bellek içi bir önbellekte tutar. Bellekte yerleşik veriler şifrelenmez. Saldırgan AIR uygulaması tarafından kullanılan belleğe erişmeyi örneğin hata ayıklayıcı kullanarak başarırsa, veritabanında açık ve şifrelenmemiş olan veriler kullanılabilir durumda olabilir.

Örnek: Şifreleme anahtarı oluşturma ve kullanma

Adobe AIR 1.5 ve üstü

Bu örnek uygulama, şifreleme anahtarı oluşturma için bir tekniği gösterir. Bu uygulama, kullanıcının verileri için en yüksek düzeyde gizlilik ve güvenlik sağlamak üzere tasarlanmıştır. Kişisel verileri korumanın önemli bir yönü, uygulamanın veritabanına her bağlanmasında kullanıcının parolayı girmesini gerektirmesidir. Sonuç olarak, bu örnekte gösterildiği gibi, bu gizlilik düzeyini gerektiren bir uygulama, veritabanı şifreleme anahtarını hiçbir zaman doğrudan saklamamalıdır.

Uygulama iki bölümden oluşur: Şifreleme anahtarını oluşturan ActionScript sınıfı (EncryptionKeyGenerator sınıfı) ve bu sınıfın nasıl kullanılacağını gösteren temel bir kullanıcı arabirimi. Tam kaynak kodu için bkz. "Şifreleme anahtarı oluşturmak için tam örnek kod" sayfa 773.

Güvenli bir şifreleme anahtarı elde etmek için EncryptionKeyGenerator sınıfını kullanma

Adobe AIR 1.5 ve üstü

EncryptionKeyGenerator sınıfının nasıl çalıştığının ayrıntılarını anlamak onu uygulamanızda kullanmak için gerekli değildir. Bir sınıfın bir veritabanı için nasıl bir şifreleme anahtarı oluşturduğunun ayrıntılarıyla ilgileniyorsanız, bkz. "EncryptionKeyGenerator sınıfını anlama" sayfa 778.

EncryptionKeyGenerator sınıfını uygulamanızda kullanmak için bu adımları izleyin:

1. EncryptionKeyGenerator sınıfını kaynak kodu veya bir derlenmiş SWC olarak indirin. EncryptionKeyGenerator sınıfı açık kaynaklı ActionScript 3.0 çekirdek kütüphanesi (as3corelib) projesine dahildir. Kaynak kod ve belgeleri içeren as3corelib paketini indirebilirsiniz. Ayrıca proje sayfasından SWC veya kaynak kod dosyalarını da indirebilirsiniz.

2 EncryptionKeyGenerator sınıfına ilişkin kaynak kodu (veya as3corelib SWC) uygulama kaynak kodunuzun onu bulabileceği bir konuma yerleştirin.

3 Uygulama kaynak kodunuzda, EncryptionKeyGenerator sınıfı için bir `import` ifadesi ekleyin.

```
import com.adobe.air.crypto.EncryptionKeyGenerator;
```

4 Kodun veritabanını oluşturduğu veya veritabanına bağlantı açtığı noktadan önce, `EncryptionKeyGenerator()` yapıcısını çağırarak bir EncryptionKeyGenerator örneği yaratacak kodu ekleyin.

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
```

5 Kullanıcıdan parola alın:

```
var password:String = passwordInput.text;

if (!keyGenerator.validateStrongPassword(password))
{
    // display an error message
    return;
}
```

EncryptionKeyGenerator örneği, bu parolayı şifreleme anahtarı için temel olarak kullanır. (Sonraki örnekte gösterilmiştir.) EncryptionKeyGenerator örneği, belirli güçlü parola doğrulama gereksinimlerine karşı parolayı test eder. Doğrulama başarısız olursa, bir hata oluşur. Örnek kodun gösterdiği gibi, parolayı daha sonra EncryptionKeyGenerator nesnesinin`validateStrongPassword()` yöntemini kullanarak kontrol edebilirsiniz. Bu yolla parolanın hatayı önlemek üzere güçlü bir parola olmak için minimum gereksinimleri karşılayıp karşılamadığını belirleyebilirsiniz.

6 Şifreleme anahtarını paroladan oluşturun:

```
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

`getEncryptionKey()` yöntemi, şifreleme anahtarını oluşturur ve döndürür (16 bayt ByteArray). Daha sonra şifreleme anahtarını yeni şifreli veritabanınızı oluşturmak için veya var olan veritabanını açmak için kullanabilirsiniz.

`getEncryptionKey()` yöntemi, adım 5'te elde edilen parola olan tek bir gerekli parametre içerir.

Not: *Veriler için güvenlik ve gizliliği en yüksek düzeyde tutmak için, uygulama veritabanına her bağlandığında kullanıcının şifre girmesini gerektirir. Kullanıcı parolasını veya veritabanı şifreleme anahtarını doğrudan saklamayın. Bunu yapmak güvenlik riskleri oluşturur. Bunun yerine, bu örnekte gösterildiği gibi, bir uygulama şifreli bir veritabanı oluştururken veya sonradan bu veritabanına bağlanırken paroladan şifreleme anahtarı türetmek için aynı tekniği kullanmalıdır.*

`getEncryptionKey()` yöntemi ikinci (isteğe bağlı) bir parametre olan `overrideSaltELSKey` parametresini de kabul eder. EncryptionKeyGenerator, şifreleme anahtarının parçası olarak kullanılan rastgele bir değer (*salt* olarak bilinir) oluşturur. Şifreleme anahtarının yeniden oluşturulabilmesi için, salt değeri AIR uygulamanızın Şifreli Yerel Depo'sunda (ELS) saklanır. Varsayılan olarak, EncryptionKeyGenerator sınıfı ELS anahtarı olarak belirli bir Dize kullanır. Düşük bir ihtimal olsa da, anahtar uygulama tarafından kullanılan başka bir anahtarla çakışabilir. Varsayılan anahtarı kullanmak yerine, kendi ELS anahtarınızı belirlemek isteyebilirsiniz. Bu durumda, burada gösterildiği gibi, ikinci `getEncryptionKey()` parametresi olarak iletilecek özel bir anahtar belirleyin:

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 Veritabanı oluşturma veya açma

Kodunuz `getEncryptionKey()` yöntemi tarafından döndürülen bir şifreleme anahtarıyla yeni bir şifreli veritabanı oluşturabilir veya var olan şifreli bir veritabanını açmayı deneyebilir. Her iki durumda da, "Şifreli veritabanı oluşturma" sayfa 767 ve "Şifreli bir veritabanına bağlanma" sayfa 767 bölümlerinde açıklandığı gibi SQLConnection sınıfının `open()` veya `openAsync()` yöntemini kullanırsınız.

Bu örnekte, uygulama veritabanını senkronize olmayan yürütme modunda açmak üzere tasarlanmıştır. Kod, uygun olay dinleyicilerini ayarlar ve şifreleme anahtarını son argüman olarak ileterek `openAsync()` yöntemini çağırır:

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

Kod, dinleyici yöntemlerinde olay dinleyicisi kayıtlarını kaldırır. Daha sonra veritabanının oluşturulduğunu, açıldığını veya bir hata oluştuğunu gösteren bir durum mesajı görüntüler. Bu olay işleyicilerinin en önemli bölümü `openError()` yöntemindedir. Bu yöntemde `if` ifadesi veritabanının var olup olmadığını (kodun var olan bir veritabanına bağlanmaya çalıştığı anlamına gelir) ve hata kimliği eşleşmelerinin sabit olup olmadığını kontrol eder `EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID`. Bu koşullardan her ikisi de doğruysa, bu büyük olasılıkla kullanıcının girdiği parolanın yanlış olduğu anlamına gelir. (Belirtilen dosyanın bir veritabanı dosyası olmadığı anlamına da gelebilir.) Hata kimliğini kontrol eden kod aşağıdadır:

```
if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

Örnek olay dinleyicileri için tam kod için bkz. "Şifreleme anahtarı oluşturmak için tam örnek kod" sayfa 773.

Şifreleme anahtarı oluşturmak için tam örnek kod

Adobe AIR 1.5 ve üstü

Aşağıdaki, "Şifreleme anahtarı oluşturma ve kullanma" örnek uygulaması için tam koddur. Kod, iki bölümden oluşur.

Örnek, paroladan şifreleme anahtarı oluşturmak için EncryptionKeyGenerator sınıfını kullanır. EncryptionKeyGenerator sınıfı açık kaynaklı ActionScript 3.0 çekirdek kütüphanesi (as3corelib) projesine dahildir. Kaynak kod ve belgeleri içeren as3corelib paketini indirebilirsiniz. Ayrıca proje sayfasından SWC veya kaynak kod dosyalarını da indirebilirsiniz.

Flex örneği

Uygulama MXML dosyası, şifreli bir veritabanına bağlantı oluşturan veya açan basit bir uygulamaya ilişkin kaynak kodunu içerir:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
creationComplete="init();">
    <mx:Script>
        <![CDATA[
            import com.adobe.air.crypto.EncryptionKeyGenerator;

            private const dbFileName:String = "encryptedDatabase.db";

            private var dbFile:File;
            private var createNewDB:Boolean = true;
            private var conn:SQLConnection;

            // ------- Event handling -------

            private function init():void
            {
                conn = new SQLConnection();
                dbFile = File.applicationStorageDirectory.resolvePath(dbFileName);
                if (dbFile.exists)
                {
                    createNewDB = false;
                    instructions.text = "Enter your database password to open the encrypted
database.";
                    openButton.label = "Open Database";
                }
            }

            private function openConnection():void
            {
                var password:String = passwordInput.text;

                var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();

                if (password == null || password.length <= 0)
                {
                    statusMsg.text = "Please specify a password.";
                    return;
                }

                if (!keyGenerator.validateStrongPassword(password))
                {
                    statusMsg.text = "The password must be 8-32 characters long. It must
contain at least one lowercase letter, at least one uppercase letter, and at least one number
or symbol.";
                    return;
                }

                passwordInput.text = "";
                passwordInput.enabled = false;
                openButton.enabled = false;

                var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);

                conn.addEventListener(SQLEvent.OPEN, openHandler);
                conn.addEventListener(SQLErrorEvent.ERROR, openError);
```

```
                conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
            }

            private function openHandler(event:SQLEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                statusMsg.setStyle("color", 0x009900);
                if (createNewDB)
                {
                    statusMsg.text = "The encrypted database was created successfully.";
                }
                else
                {
                    statusMsg.text = "The encrypted database was opened successfully.";
                }
            }

            private function openError(event:SQLErrorEvent):void
            {
                conn.removeEventListener(SQLEvent.OPEN, openHandler);
                conn.removeEventListener(SQLErrorEvent.ERROR, openError);

                if (!createNewDB && event.error.errorID ==
EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
                {
                    statusMsg.text = "Incorrect password!";
                }
                else
                {
                    statusMsg.text = "Error creating or opening database.";
                }
            }
        ]]>
    </mx:Script>
    <mx:Text id="instructions" text="Enter a password to create an encrypted database. The next
time you open the application, you will need to re-enter the password to open the database
again." width="75%" height="65"/>
    <mx:HBox>
        <mx:TextInput id="passwordInput" displayAsPassword="true"/>
        <mx:Button id="openButton" label="Create Database" click="openConnection();"/>
    </mx:HBox>
    <mx:Text id="statusMsg" color="#990000" width="75%"/>
</mx:WindowedApplication>
```

Flash Professional örneği

Uygulama FLA dosyası, şifreli bir veritabanına bağlantı oluşturan veya açan basit bir uygulamaya ilişkin kaynak kodunu içerir. FLA dosyası, sahnede konumlandırılmış dört bileşen içerir:

Örnek adı	Bileşen türü	Açıklama
`instructions`	Label	Kullanıcıya verilen talimatları içerir
`passwordInput`	TextInput	Kullanıcının parolayı girdiği giriş alanı
`openButton`	Düğme	Kullanıcının parolayı girdikten sonra tıklattığı düğme
`statusMsg`	Etiket	Durum (başarı veya hata) mesajlarını görüntüler

Uygulamaya ilişkin kod, ana zaman çizelgesinin kare 1'i üzerindeki anahtar karede tanımlanmıştır. Aşağıdaki, uygulamaya ilişkin koddur:

```
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.";
    }
}
```

EncryptionKeyGenerator sınıfını anlama

Adobe AIR 1.5 ve üstü

EncryptionKeyGenerator sınıfını uygulama veritabanınıza ilişkin güvenli bir şifreleme anahtarı oluşturmak üzere kullanmak için bu sınıfın iç çalışma yapısını anlamak gerekli değildir. Sınıfı kullanma işlemleri "Güvenli bir şifreleme anahtarı elde etmek için EncryptionKeyGenerator sınıfını kullanma" sayfa 771 bölümünde anlatılmıştır. Ancak, bu sınıfın kullandığı teknikleri anlamayı önemli görüyor olabilirsiniz. Örneğin, sınıfı uyarlamak veya farklı bir veri gizlilik düzeyi istenen durumlar için sınıfın tekniklerinden bazılarını birleştirmek isteyebilirsiniz.

EncryptionKeyGenerator sınıfı açık kaynaklı ActionScript 3.0 çekirdek kütüphanesi (as3corelib) projesine dahildir. Kaynak kodu ve belgeler dahil the as3corelib paketini indirebilirsiniz.Ayrıca proje sitesinde kaynak kodunu görüntüleyebilir ve açıklamalarla birlikte takip etmek üzere indirebilirsiniz.

Kod bir EncryptionKeyGenerator örneği oluşturduğunda ve bu örneğin `getEncryptionKey()` yöntemini çağırdığında, yalnızca hak sahibi kullanıcının verilere erişebilmesini sağlamak için bazı adımlar atılır. Veritabanı oluşturulmadan önce kullanıcı tarafından girilen paroladan bir şifreleme anahtarı oluşturma işlemi, veritabanını açmak için şifreleme anahtarını yeniden oluşturma işlemiyle aynıdır.

Güçlü bir parola elde edin ve doğrulayın

Adobe AIR 1.5 ve üstü

Kod `getEncryptionKey()` yöntemini çağırdığında, parametre olarak bir parola sunar. Parola, şifreleme anahtarı için temel olarak kullanılır. Bu tasarım, yalnızca kullanıcının bildiği bir bilgiyi kullanarak yalnızca parolayı bilen kullanıcının veritabanındaki verilere ulaşabilmesini sağlar. Bir saldırgan kullanıcının bilgisayardaki hesabına erişse bile, parolayı bilmeden veritabanına giremez. Maksimum güvenlik için, uygulama parolayı hiçbir zaman saklamaz.

Bir uygulamanın kodu bir EncryptionKeyGenerator örneği oluşturur ve argüman olarak kullanıcı tarafından girilen bir şifreyi geçerek onun `getEncryptionKey()` yöntemini çağırır (bu örnekte `password` değişkeni):

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

`getEncryptionKey()` yöntemi çağrıldığında EncryptionKeyGenerator sınıfının attığı ilk adım, parola güvenlik düzeyi gereksinimlerini karşıladığından emin olmak için kullanıcı tarafından girilen parolayı kontrol etmektir. EncryptionKeyGenerator sınıfı bir şifrenin 8 - 32 karakter arasında olmasını gerektirir. Şifre, büyük ve küçük harflerin karışımından ve en az bir sayı veya sembol karakterinden oluşmalıdır.

Bu modeli kontrol eden yaygın ifade, `STRONG_PASSWORD_PATTERN` adlı bir sabit olarak tanımlanır:

```
private static const STRONG_PASSWORD_PATTERN:RegExp =
/(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

Parolayı kontrol eden kod, EncryptionKeyGenerator sınıfının `validateStrongPassword()` yöntemindedir. Kod şöyledir:

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

Dahili olarak, `getEncryptionKey()` yöntemi EncryptionKeyGenerator sınıfının `validateStrongPassword()` yöntemini çağırır ve parola geçerli değilse bir istisna atar. `validateStrongPassword()` yöntemi, genel bir yöntemdir ve böylece uygulama kodu, hata oluşmasını önlemek için `getEncryptionKey()` yöntemini çağırmadan parolayı kontrol edebilir.

Parolayı 256 bit olarak genişletin

Adobe AIR 1.5 ve üstü

İşlemin sonraki aşamalarında, parola 256 bit uzunluğunda olmalıdır. Her kullanıcının tam olarak 256 bit (32 karakter) uzunluğunda olan bir parola girmesini gerektirmek yerine, kod parola karakterlerini yineleyerek daha uzun bir parola oluşturur.

`getEncryptionKey()` yöntemi, uzun parola oluşturma görevini gerçekleştirmek için `concatenatePassword()` yöntemini çağırır.

```
var concatenatedPassword:String = concatenatePassword(password);
```

Aşağıdaki, `concatenatePassword()` yöntemine ilişkin koddur:

```
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;
}
```

Parola 256 bitten kısa olduğunda, kod parolayı 256 bit yapmak için kendisiyle bitiştirir. Uzunluk tam olarak oluşmuyorsa, son yineleme parolanın tam olarak 256 bit olacağı biçimde kısaltılır.

256 bit salt değeri oluşturun veya elde edin

Adobe AIR 1.5 ve üstü

Sonraki adım, daha sonraki bir adımda parolayla birleştirilecek bir 256 bit salt değeri elde etmektir. *salt*, parola oluşturmak için kullanıcı tarafından girilen bir değere eklenen veya kullanıcı tarafından girilen bir değerle birleştirilen rastgele bir değerdir. Salt değerini bir parolayla kullanmak, kullanıcı parola olarak gerçek bir sözcük veya yaygın bir terim seçtiğinde bile sistemin kullandığı parola+salt bileşiminin rastgele bir değer olmasını sağlar. Bu değerin rastgele oluşu, saldırganın parolayı tahmin etmek için bir sözcük listesi kullandığı sözlük saldırısına karşı korunmaya yardımcı olur. Ayrıca, salt değerini oluşturma ve şifreli yerel depoda saklama, bu değeri veritabanı dosyasının bulunduğu bilgisayardaki kullanıcı hesabına bağlar.

Uygulama `getEncryptionKey()` yöntemini ilk kez çağırıyorsa, kod bir 256 bit salt değeri oluşturur. Aksi halde, kod salt değerini şifreli yerel depodan yükler.

Salt, `salt` adlı bir değişkende saklanır. Kod, salt değerini şifreli yerel depodan yüklemeyi deneyerek onu daha önceden oluşturulmuş olup olmadığını belirler:

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

Kod yeni bir salt değeri oluşturduğunda, `makeSalt()` yöntemi 256 bit rastgele bir değer oluşturur. Değer şifreli yerel depoda saklandığından, bir ByteArray nesnesi olarak oluşturulur. `makeSalt()` yöntemi, rastgele bir değer oluşturmak için `Math.random()` yöntemini kullanır. `Math.random()` yöntemi bir defada 256 bit oluşturamaz. Bunun yerine, kod `Math.random()` yöntemini sekiz defa çağırmak için bir döngü kullanır. Her defasında 0 ve 4294967295 arasında (maksimum uint değeri) yer alan rastgele bir uint değeri oluşturulur. Uint değeri tam olarak 32 bit kullandığından, kolaylık sağlamak için kullanılır. ByteArray öğesine sekiz uint değeri yazılarak bir 256 bit değer oluşturulur. Aşağıdaki, `makeSalt()` yöntemine ilişkin koddur:

```
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;
}
```

Kod salt değerini Şifreli Yerel Depo'ya (ELS) kaydettiğinde veya bu değeri ELS'den aldığında, salt değerinin altında saklanacağı String anahtarına ihtiyaç duyar. Anahtarı bilmeden, salt değeri elde edilemez. Bu durumda, şifreleme anahtarı veritabanını yeniden açmak için her defasında yeniden oluşturulamaz. EncryptionKeyGenerator, varsayılan olarak `SALT_ELS_KEY` sabitinde tanımlanan ön tanımlı bir ELS anahtarı kullanır. Varsayılan anahtarı kullanmak yerine, uygulama kodu `getEncryptionKey()` yöntemine yapılan çağrıda kullanmak üzere bir ELS anahtarı da belirleyebilir. Varsayılan veya uygulama tarafından belirlenen salt ELS anahtarı, `saltKey` adlı bir değişkende saklanır. Bu değişken, daha önce gösterildiği gibi, `EncryptedLocalStore.setItem()` ve `EncryptedLocalStore.getItem()` öğelerine yapan çağrılarda kullanılır.

XOR operatörünü kullanarak 256 bit parolayı ve salt değerini birleştirin.

Adobe AIR 1.5 ve üstü

Kod, artık bir 256 bit parola ve bir 256 bit salt değerine sahiptir. Daha sonra, salt değeri ve bitiştirilmiş parolayı tek bir değer halinde birleştirmek için bit tabanlı XOR işlemini kullanır. Bu teknik, tüm olası karakterler aralığında yer alan karakterlerden oluşan bir 256 bit parola oluşturur. Gerçek parola girişi genellikle temel olarak alfanümerik karakterlerden oluştuğu halde bu ilke geçerlidir. Rastgeleliğin artırılması, kullanıcının uzun, karmaşık bir parola girmesini gerektirmeden olası parolalar kümesini genişletir.

XOR işleminin sonucu `unhashedKey` değişkeninde saklanır. İki değer üzerinde bit tabanlı bir XOR gerçekleştirme işlemi `xorBytes()` yönteminde olur:

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

Bit tabanlı XOR operatörü (`^`) iki uint değerini alır ve bir uint değeri döndürür. (Bir uint değeri 32 bit içerir.) `xorBytes()` yöntemine iletilen giriş değerleri, String (parola) ve ByteArray (salt) değerleridir. Sonuç olarak, kod XOR operatörünü kullanarak her girişten bir defada birleştirilecek 32 bit ayıklamak için bir döngü kullanır.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

Döngü içinde ilk 32 bit (4 bayt) `passwordString` parametresinden ayıklanır. Bu bitler ayıklanır ve iki bölümlü bir işlemde bir uint (`o1`) değerine dönüştürülür. İlk olarak, `charCodeAt()` yöntemi her karakterin sayısal değerini getirir. Daha sonra, sol shift operatörü kullanılarak bu değer uint değerindeki uygun konuma kaydırılır (`<<`) ve kaydırılan değer `o1` değerine eklenir. Örneğin, bitleri 24 bit sola kaydırmak için bit tabanlı sol shift operatörü (`<<`) kullanılması ve bu değerin `o1` değerine atanması yoluyla, ilk karakter (`i`) ilk 8 bit olur. İkinci karakter `(i + 1)`, değerinin 16 bit sola kaydırılması ve sonucun `o1` değerine eklenmesiyle, ikinci 8 bit olur. Üçüncü ve dördüncü karakterler de aynı şekilde eklenir.

```
        // ...

        // 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);

        // ...
```

`o1` değişkeni artık `passwordString` parametresinden 32 bit içerir. Daha sonra, `readUnsignedInt()` yöntemi çağrılarak `salt` değerinden 32 bit ayıklanır. 32 bit, `o2` uint değişkeninde saklanır.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

Son olarak, iki 32 bit (uint) değer XOR operatörü kullanılarak birleştirilir ve sonuç `result` adlı ByteArray öğesine yazılır.

```
        // ...

        var xor:uint = o1 ^ o2;
        result.writeUnsignedInt(xor);
        // ...
```

Döngü tamamlandığında, XOR sonucunu içeren ByteArray döndürülür.

```
        // ...
    }

    return result;
}
```

Anahtar için karma oluşturun

Adobe AIR 1.5 ve üstü

Bitiştirilen parola ve salt değeri birleştirildiğinde, sonraki adım SHA-256 karma algoritma aracılığıyla karma oluşturmak ve bu değeri daha güvenli kılmaktır. Değer için karma oluşturmak, saldırganın tersine mühendislik uygulamasını zorlaştırır.

Bu noktada kod, salt değeriyle birleştirilen bitiştirilmiş parolayı içeren `unhashedKey` adlı bir ByteArray öğesi içerir. ActionScript 3.0 core library (as3corelib) projesi, com.adobe.crypto paketinde bir SHA256 sınıfı içerir. ByteArray üzerinde SHA-256 karma gerçekleştiren ve onaltılı bir sayı olarak 256 bit karma sonucu içeren bir String döndüren `SHA256.hashBytes()` yöntemi. EncryptionKeyGenerator sınıfı, anahtar için karma oluşturmak üzere SHA256 sınıfını kullanır.

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

Karmadan şifreleme anahtarını ayıklayın.

Adobe AIR 1.5 ve üstü

Şifreleme anahtarı, tam olarak 16 bayt (128 bit) uzunluğunda olan bir ByteArray olmalıdır. SHA-256 karma algoritma sonucu her zaman 256 karakter uzunluğundadır. Sonuç olarak, son adım geçerli şifreleme anahtarı olarak kullanılmak üzere karma oluşturulan sonuçtan 128 bit seçmektir.

Kod, EncryptionKeyGenerator sınıfında `generateEncryptionKey()` yöntemini çağırarak anahtarı 128 bite indirger. Daha sonra bu yöntemin sonucunu `getEncryptionKey()` yönteminin sonucu olarak döndürür:

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

İlk 128 biti şifreleme anahtarı olarak kullanmak gerekmez. İstediğiniz bir noktadan başlayan bir bit aralığı seçebilir, diğer bitleri seçebilir veya başka bir bit seçme yolu kullanabilirsiniz. Önemli olan, kodun 128 ayrı bit seçmesi ve her defasında aynı 128 bitin kullanılmasıdır.

Bu durumda, `generateEncryptionKey()` yöntemi şifreleme anahtarının 18. baytında başlayan bit aralığını kullanır. Önceden belirtildiği gibi, SHA256 sınıfı onaltılı bir sayı olan 256 bit karma içeren bir String değeri döndürür. 128 bitlik tek bir blok, bir defada ByteArray öğesine eklemek için çok fazla bayt içerir. Sonuç olarak, kod karakterleri onaltılı String öğesinden ayıklamak, bunları gerçek sayısal değerlere dönüştürmek ve ByteArray öğesine eklemek için `for` döngüsünü kullanır. SHA 256 sonuç String'i 64 karakter uzunluğundadır. 128 bit aralığı String'de 32 karaktere eşittir ve her karakter 4 biti temsil eder. ByteArray öğesine ekleyebileceğiniz en küçük veri artışı, `hash` String'de iki karaktere eşit olan bit bayttır (8 bit). Sonuç olarak, 2 karakter artışında döngü 0'dan 31'e kadar (32 karakter) sayar.

Döngü içinde, kod önce geçerli karakter çifti için başlangıç konumu belirler. İstenen aralık dizin konumu 17'deki (18. bayt) karakterde başladığından, `position` değişkenine geçerli yineleyici değeri (`i`) + 17 atanır. Kod, geçerli konumda iki karakter ayıklamak için String nesnesinin `substr()` yöntemini kullanır. Bu karakterler `hex` değişkeninde saklanır. Daha sonra, kod `hex` String'ini onlu tamsayı değerine dönüştürmek için `parseInt()` yöntemini kullanır. Bu değeri `byte` değişkeninde saklar. Son olarak, kod `writeByte()` yöntemini kullanarak `byte` içindeki değeri `result` ByteArray öğesine ekler. Döngü tamamlandığında, `result` ByteArray 16 bayt içerir ve veritabanı şifreleme anahtarı olarak kullanılmaya hazırdır.

```
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;
}
```

SQL veritabanlarıyla çalışmak için stratejiler

Adobe AIR 1.0 ve üstü

Bir uygulamanın yerel SQL veritabanlarına erişmesi ve bu veritabanlarıyla çalışmasının çeşitli yolları vardır. Uygulama tasarımı, uygulama kodunun düzenlenme biçimi, işlemlerin gerçekleştirilme sırası ve zamanlaması, vb. gibi konularda farklılaşır. Seçtiğiniz teknikler, uygulamanızı geliştirmenin kolaylığı üzerinde etkili olabilir. Bu teknikler, uygulamayı gelecekteki güncellemelerde değiştirmenin kolaylığını etkileyebilir. Uygulamanın kullanıcının bakış açısına göre ne kadar iyi çalıştığı da bu tekniklere göre değişebilir.

Önceden doldurulmuş bir veritabanını dağıtma

Adobe AIR 1.0 ve üstü

Uygulamanızda bir AIR yerel SQL veritabanı kullandığınızda, uygulama tablolar, sütunlar, vb. içeren belirli bir yapıya sahip bir veritabanı bekler. Bazı uygulamalar veritabanı dosyasının önceden veriyle doldurulmuş olmasını da bekler. Veritabanının uygun bir yapıya sahip olmasını sağlamanın bir yolu veritabanını uygulama kodunda oluşturmaktır. Uygulama yüklendiğinde, belirli bir konumdaki veritabanı dosyasının varlığını kontrol eder. Dosya mevcut değilse, uygulama veritabanı dosyasını ve veritabanı yapısını oluşturmak, tabloları başlangıç verileriyle doldurmak için bir dizi komut yürütür.

Veritabanını ve veritabanı tablolarını oluşturan kod genellikle karmaşıktır. Bu kod, uygulamanın yüklenen kullanım ömründe genellikle yalnızca bir defa kullanılır, ancak yine de uygulamanın boyutuna ve karmaşıklığına katkıda bulunur. Veritabanını, yapıyı ve verileri programlama yoluyla oluşturmaya alternatif olarak, uygulamanızla birlikte önceden doldurulmuş bir veritabanını dağıtabilirsiniz. Önceden doldurulmuş bir veritabanı dağıtmak için, veritabanı dosyasını uygulamanın AIR paketine dahil edin.

AIR paketinde bulunan tüm dosyalar gibi, paketlenmiş bir veritabanı dosyası uygulama dizinine yüklenir (`File.applicationDirectory` özelliğiyle temsil edilen dizin). Ancak, bu dizindeki dosyalar salt okunur dosyalardır. AIR paketindeki dosyayı bir "şablon" veritabanı olarak kullanın. Kullanıcı uygulamayı ilk kez çalıştırdığında, orijinal veritabanı dosyasını kullanıcının "Uygulama depolama dizinine işaret etme" sayfa 679'ne (veya başka bir konuma) kopyalayın ve uygulamada bu veritabanını kullanın.

Yerel SQL veritabanlarıyla çalışmak için en iyi uygulamalar

Adobe AIR 1.0 ve üstü

Aşağıdaki liste, yerel SQL veritabanlarıyla çalışırken performansı, güvenliği ve uygulama bakımınızın kolaylığını artırmak için kullanabileceğiniz önerilen tekniklerin kümesidir.

Veritabanı bağlantılarını önceden oluşturun

Adobe AIR 1.0 ve üstü

Uygulamanız ilk yüklendiğinde herhangi bir ifade yürütmese bile, bir SQLConnection nesnesini başlatın ve ifade çalıştırma sırasında gecikmeleri önlemek için önceden (örneğin ilk uygulama başlangıcından sonra) bu nesnenin `open()` veya `openAsync()` yöntemini çağırın. Bkz. "Veritabanına bağlanma" sayfa 734.

Veritabanı bağlantılarını yeniden kullanma

Adobe AIR 1.0 ve üstü

Uygulamanızın yürütülmesi sırasında belirli bir veritabanına erişiyorsanız, SQLConnection örneğine ilişkin bir başvuru tutun ve bağlantıyı kapatıp yeniden açmak yerine uygulama boyunca bu başvuruyu yeniden kullanın. Bkz. "Veritabanına bağlanma" sayfa 734.

Senkronize olmayan yürütme modunu tercih edin

Adobe AIR 1.0 ve üstü

Veri erişim kodunu yazma sırasında işlemleri senkronize olarak yürütmek uygun görünebilir, çünkü senkronize işlemleri kullanmak genellikle daha kısa ve daha az karmaşık kodlar gerektirir. Ancak, "Senkronize ve senkronize olmayan veritabanı işlemlerini kullanma" sayfa 761 bölümünde açıklandığı gibi, senkronize işlemlerin performans üzerinde kullanıcıların açıkça görebildiği, uygulamayı yavaşlatan bir etkisi vardır. Tek bir işlemin aldığı süre işleme ve özellikle işlemin içerdiği veri miktarına göre değişir. Örneğin, veritabanına yalnızca bir satır ekleyen bir SQL `INSERT` ifadesi, binlerce veri satırı alan bir `SELECT` ifadesinden çok daha az zaman alır. Ancak, birden çok işlemi gerçekleştirmek için senkronize yürütmeyi kullandığınızda, işlemler genellikle yanyana dizilir. Tek bir işlemin aldığı zaman çok kısa olsa bile, tüm senkronize işlem tamamlanana kadar uygulama donar. Sonuç olarak, yanyana dizilen birden çok işlemin gerektirdiği birikimli süre, uygulamanızı durdurmak için yeterli olabilir.

Özellikle çok sayıda satır içeren işlemler için standart bir yaklaşım olan senkronize olmayan işlemleri kullanın. "SELECT sonuçlarını bölümler halinde alma" sayfa 750 bölümünde anlatıldığı gibi, `SELECT` ifadesi sonuçları işlemesinin bölümlenmesi için bir teknik mevcuttur. Ancak, bu teknik yalnızca senkronize olmayan yürütme modunda kullanılabilir. Senkronize işlemleri yalnızca senkronize olmayan programlamayı kullanarak belirli işlevleri yürütemediğinizde, uygulamanızın kullanıcılarının performanstan ödün vermesi gerektiğini göz önüne alarak ve uygulamanızın performansının nasıl etkileneceğini bileceğiniz biçimde uygulamanızı test ettiyseniz kullanın. Senkronize olmayan yürütmenin kullanılması daha karmaşık kodlama içerebilir. Ancak, kodu yalnızca bir defa yazmanız gerektiğini, ancak uygulama kullanıcılarının bu kodu hızlı veya yavaş biçimde defalarca kullanmak zorunda olduklarını unutmayın.

Çoğu durumda, yürütülecek her SQL ifadesi için ayrı bir SQLStatement örneği kullanılarak, senkronize olmayan kodun yazılma biçimi bakımından senkronize koda benzemesini sağlayacak şekilde, birden çok SQL işlemi bir defada kuyruğa alınabilir. Daha fazla bilgi için bkz. "Senkronize olmayan yürütme modelini anlama" sayfa 765.

Ayrı SQL ifadeleri kullanın ve SQLStatement'ın text özelliğini değiştirmeyin.

Adobe AIR 1.0 ve üstü

Bir uygulamada bir defadan çok yürütülen bir SQL ifadesi için, her SQL ifadesine ilişkin ayrı bir SQLStatement örneği oluşturun. SQLStatement örneğini SQL komutu her yürütüldüğünde kullanın. Örneğin, birçok defa gerçekleştirilecek dört farklı SQL işlemi içeren bir uygulama oluşturduğunuzu varsayın. Bu durumda, dört ayrı SQLStatement örneği oluşturun ve her ifadeyi yürütmek için bu ifadenin `execute()` yöntemini çağırın. Tüm SQL ifadeleri için tek bir SQLStatement örneği kullanarak ifadenin her yürütülmesinden önce `text` özelliğini yeniden tanımlama alternatifinden kaçının.

İfade parametreleri kullanın

Adobe AIR 1.0 ve üstü

SQLStatement parametreleri kullanın—kullanıcı girişini hiçbir zaman ifade metnine bitiştirmeyin. SQL yerleştirme saldırıları olasılığını engellediğinden, parametre kullanmak uygulamanız için her zaman daha güvenlidir. Sorgularda nesneleri kullanmayı (yalnızca SQL değişmez değerleri yerine) olanaklı kılar. Ayrıca, her yürütüldüklerinde yeniden derlenmeleri gerekmeden yeniden kullanılabildiklerinden, ifadelerin daha verimli biçimde çalışmasını sağlar. Daha fazla bilgi için bkz. "İfadelerde parametreler kullanma" sayfa 738.

Bölüm 41: Bayt dizileriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ByteArray sınıfı, esasında bayt dizisi olan ikili veri akışını okumanızı veya ikili veri akışına yazmanızı sağlar. Bu sınıf verilere en basit şekilde erişmek için bir yol sağlar. Bilgisayar verileri baytlardan veya 8 bitlik gruplardan oluştuğu için, verilerin bayt olarak okunabilmesi, sınıf ve erişim yöntemi olmayan verilere erişebildiğiniz anlamına gelir. ByteArray sınıfı, bir bitmap'ten ağda dolaşan veri akışına kadar bayt düzeyinde herhangi bir veri akışını ayrıştırmanızı sağlar.

`writeObject()` yöntemi, bir nesneyi serileştirilmiş Action Message Format (AMF) olarak ByteArray öğesine yazmanızı sağlar; `readObject()` yöntemi ise ByteArray öğesinden orijinal veri türü değişkenine kadar, serileştirilmiş bir nesneyi okumanızı sağlar. Görüntüleme listesine yerleştirilebilen nesneler olan görüntüleme nesneleri dışındaki tüm nesneleri serileştirebilirsiniz. Ayrıca custom sınıfı çalışma zamanı için kullanılabilirse, serileştirilmiş nesneleri tekrar custom sınıfı örneklerine atayabilirsiniz. Bir nesneyi AMF'ye dönüştürdükten sonra, bir ağ bağlantısı üzerinden etkili bir şekilde aktarabilir veya bir dosyaya kaydedebilirsiniz.

Burada anlatılan örnek Adobe® AIR™ uygulaması bir .zip dosyasını, .zip dosyasının içerdiği dosya listesini çıkartarak ve onları masaüstüne yazarak bayt akışını işleme örneği olarak okur.

Daha fazla Yardım konusu

flash.utils.ByteArray

flash.utils.IExternalizable

Action Message Format belirtimi

ByteArray okuma ve yazma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ByteArray sınıfı, flash.utils paketinin bir parçasıdır. ActionScript 3.0'da ByteArray nesnesi oluşturmak için, aşağıdaki örnekte gösterildiği gibi ByteArray sınıfını içe aktarın ve yapıcıyı çağırın:

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

ByteArray yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Anlamlı bir veri akışı, istediğiniz bilgiyi bulmak için analiz edebileceğiniz bir biçim olarak organize edilir. Örneğin basit bir çalışan dizisindeki kayıt büyük olasılıkla kimlik numarası, ad, adres, telefon numarası vb. içerir. MP3 ses dosyası, indirilmekte olan dosyanın başlığını, yazarını, albümü, yayınlanma tarihini ve türünü belirten bir Kimlik3 etiketi içerir. Biçim sayesinde, veri akışındaki verileri hangi sırada bekleyeceğinizi bilirsiniz. Bayt akışını mantıklı bir şekilde okuyabilirsiniz.

ByteArray sınıfı bir veri akışından okumayı ve veri akışına yazmayı kolaylaştıran birçok yöntem içerir. Bunlardan bazıları `readBytes()` ve `writeBytes()`, `readInt()` ve `writeInt()`, `readFloat()` ve `writeFloat()`, `readObject()` ve `writeObject()` ve `readUTFBytes()` ve `writeUTFBytes()` yöntemleridir. Bu yöntemler sayesinde veri akışından verileri belirli veri türleri değişkenlerinde okuyabilir ve belirli veri türlerini doğrudan ikili veri akışına yazabilirsiniz.

Örneğin aşağıdaki kod basit bir dize dizisini ve kayar nokta sayısını okur ve her öğeyi ByteArray öğesine yazar. Dizinin organizasyonu sayesinde kod, verileri yazmak için uygun ByteArray yöntemlerini (`writeUTFBytes()` ve `writeFloat()`) çağırabilir. Yinelenen veri modeli sayesinde diziyi bir döngüyle okuyabilirsiniz.

```
 // 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
```

position özelliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

position özelliği, okuma veya yazma sırasında ByteArray öğesini dizine alan işaretçinin geçerli konumunu saklar. position özelliğinin başlangıç değeri, aşağıdaki kodda da gösterildiği gibi 0'dır (sıfır):

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

Bir ByteArray öğesinden okuma veya ByteArray öğesine yazma işlemi gerçekleştirdiğinizde, kullandığınız yöntem, position özelliğini okunan veya yazılan son baytın hemen ardından konuma işaret edecek şekilde güncelleştirir. Örneğin aşağıdaki kod ByteArray öğesine bir dize yazar ve sonrasında position özelliği, ByteArray öğesindeki dizenin hemen ardından bayta işaret eder.

```
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
```

Benzer şekilde, bir okuma işlemi de okunan bayt sayısını position özelliğine göre artırır.

```
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!
```

Söz konusu uzaklıkta okumak veya yazmak için position özelliğini ByteArray öğesinde belirli bir konuma ayarlayabilirsiniz.

bytesAvailable ve length özellikleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`length` ve `bytesAvailable` özellikleri, ByteArray öğesinin ne kadar süreceğini ve geçerli konumdan sonuna kadar kaç baytı kaldığını gösterir. Aşağıdaki örnek bu özellikleri nasıl kullanabileceğinizi gösterir. Örnek, metnin bir Dizesini ByteArray öğesine yazar ve ardından, "a" ile veya son ile karşılaşana kadar (`bytesAvailable <= 0`) bir kerede bir bayt olacak şekilde ByteArray öğesini okur.

```
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
}
```

endian özelliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bilgisayarlar çok baytlı, yani saklanması için 1 bayttan fazla bellek gereken sayıları saklama yöntemlerine göre farklılık gösterebilir. Örneğin bir tam sayı belleğin 4 baytını veya 32 bitini kaplayabilir. Bazı bilgisayarlar öncelikle sayının en yüksek değerli baytını en düşük bellek adresinde saklar, diğer bilgisayarlar ise öncelikle en düşük değerli baytı saklar. Bilgisayarın veya bayt sıralamasının bu niteliği, *big endian* (önce en yüksek değerli bayt) veya *little endian* (önce en düşük değerli bayt) olarak adlandırılır. Örneğin 0x31323334 sayısı big endian ve little endian bayt sıralaması için aşağıdaki gibi saklanır, a0, 4 bayt arasında en düşük bellek adresini ve a3 en yüksek olanı temsil eder:

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

ByteArray sınıfının `endian` özelliği, işlediğiniz çok baytı sayılar için bu bayt sırasını belirtmenizi sağlar. Bu özellik için kabul edilebilir değerler `"bigEndian"` veya `"littleEndian"` değerleridir ve Endian sınıfı, `endian` özelliğini bu dizelerle ayarlamak için `BIG_ENDIAN` ve `LITTLE_ENDIAN` sabitlerini tanımlar.

compress() ve uncompress() yöntemleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`compress()` yöntemi sayesinde bir ByteArray öğesini, parametre olarak belirttiğiniz bir sıkıştırma algoritmasına göre sıkıştırabilirsiniz. `uncompress()` yöntemi sayesinde, sıkıştırılmış bir ByteArray öğesini sıkıştırma algoritmasına göre genişletebilirsiniz. `compress()` ve `uncompress()` yöntemlerini çağırdıktan sonra, bayt dizisinin uzunluğu yeni uzunluğa ve position özelliği sona ayarlanır.

CompressionAlgorithm sınıfı, sıkıştırma algoritmasını belirtmek için kullanabileceğiniz sabitleri tanımlar. ByteArray sınıfı deflate (yalnızca AIR), zlib ve lzma algoritmalarını destekler. Zlib sıkıştırılmış veri biçimi açıklaması için bkz. http://www.ietf.org/rfc/rfc1950.txt lzma algoritması Flash Player 11.4 ve AIR 3.4 için eklendi. Açıklaması için bkz. http://www.7-zip.org/7z.html.

Deflate sıkıştırma algoritması zlib, gzip ve bazı zip uygulamaları gibi çeşitli sıkıştırma biçimlerinde kullanılır. Deflate sıkıştırma algoritması açıklaması için bkz. http://www.ietf.org/rfc/rfc1951.txt.

Aşağıdaki örnek, `bytes` adındaki ByteArray öğesini lzma algoritması kullanarak sıkıştırır:

```
bytes.compress(CompressionAlgorithm.LZMA);
```

Aşağıdaki örnek sıkıştırılmış bir ByteArray öğesini deflate algoritması kullanarak genişletir:

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

Nesneleri okuma ve nesnelere yazma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`readObject()` ve `writeObject()` yöntemleri, serileştirilmiş Action Message Format (AMF) olarak kodlanmış şekilde ByteArray öğesinden bir nesne okur ve ByteArray öğesine bir nesne yazar. AMF, Adobe tarafından oluşturulan ve Netstream, NetConnection, NetStream, LocalConnection ve Paylaşılan Nesneleri de içeren çeşitli ActionScript 3.0 sınıfları tarafından kullanılan özel bir mesaj protokolüdür.

Tek baytlı tür işaretleyicisi, ardından gelen kodlanmış verilerin türünü açıklar. AMF aşağıdaki 13 veri türünü kullanır:

```
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
```

Kodlanmış veriler, işaretleyici null veya true ya da false gibi, hiçbir verinin kodlanmadığı tek bir olası değer temsil edene kadar tür işaretleyicisini izler.

AMF'nin iki sürümü vardır: AMF0 ve AMF3. AMF 0, karmaşık nesneleri başvuruya göre göndermeyi destekler ve uç noktaların nesne ilişkilerini geri yüklemesini sağlar. AMF 3, nesne özelliklerini ve dizeleri nesne başvurularına ek olarak başvuruya göre göndererek ve ActionScript 3.0 uygulamasında tanıtılan yeni veri türlerini destekleyerek AMF 0'ı iyileştirir. `ByteArray.objectEcoding` özelliği, nesne verilerinin kodlanmasında kullanılan AMF sürümünü belirtir. flash.net.ObjectEncoding sınıfı, AMF sürümünü belirtmeye yönelik sabitleri tanımlar: `ObjectEncoding.AMF0` ve `ObjectEncoding.AMF3`.

Aşağıdaki örnek, bir XML nesnesini ByteArray öğesine yazmak için `writeObject()` öğesini çağırır, ardından Deflate algoritmasını kullanarak öğeyi genişletir ve masaüstündeki `order` dosyasına yazar. Örnek, işlem bittiğinde AIR penceresinde “Sıra dosyası masaüstüne yazıldı!” mesajını görüntülemek için mesajını görüntüler.

```
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();
    }
}
```

`readObject()` yöntemi, serileştirilmiş AMF'deki bir nesneyi ByteArray öğesinden okur ve belirtilen türde bir nesnede saklar. Aşağıdaki örnek `order` dosyasını masaüstünden ByteArray öğesine (`inBytes`) okur, genişletir ve `orderXML` XML nesnesinde saklamak için `readObject()` öğesini çağırır. Örnek, her düğümü görüntüleme için bir metin alanına eklemek üzere `for each()` döngü yapısı kullanır. Örnek ayrıca `objectEncoding` özelliğinin değerini de, `order` dosyasının içeriğine yönelik bir üstbilgiyle birlikte görüntüler.

```
import flash.filesystem.*;
import flash.display.Sprite;
import flash.display.TextField;
import flash.utils.ByteArray;

public class ReadObjectExample extends Sprite
{
    public function ReadObjectExample()
    {
        var inBytes:ByteArray = new ByteArray();
        // define text area for displaying XML content
        var myTxt:TextField = new TextField();
        myTxt.width = 550;
        myTxt.height = 400;
        addChild(myTxt);
        //display objectEncoding and file heading
        myTxt.text = "Object encoding is: " + inBytes.objectEncoding + "\n\n" + "order file: \n\n";
        readFileIntoByteArray("order", inBytes);

        inBytes.position = 0; // reset position to beginning
        inBytes.uncompress(CompressionAlgorithm.DEFLATE);
        inBytes.position = 0;//reset position to beginning
        // read XML Object
        var orderXML:XML = inBytes.readObject();

        // for each node in orderXML
        for each (var child:XML in orderXML)
        {
            // append child node to text area
            myTxt.text += child + "\n";
        }
    }

    // read specified file into byte array
    private function readFileIntoByteArray(fileName:String, data:ByteArray):void
    {
        var inFile:File = File.desktopDirectory; // source folder is desktop
        inFile = inFile.resolvePath(fileName);  // name of file to read
        var inStream:FileStream = new FileStream();
        inStream.open(inFile, FileMode.READ);
        inStream.readBytes(data);
        inStream.close();
    }
}
```

ByteArray örneği: Bir .zip dosyasını okuma

Adobe AIR 1.0 ve üstü

Bu örnek, farklı türlerde birçok dosya içeren basit bir .zip dosyasının nasıl okunacağını gösterir. Bu işlemi her dosyanın meta verilerinden ilgili verileri ayıklayarak, her dosyayı bir ByteArray öğesine genişleterek ve dosyayı masaüstüne yazarak yapar.

Bir .zip dosyasının genel yapısı PKWARE Inc.in belirtimine dayalıdır, bu belirtim http://www.pkware.com/documents/casestudies/APPNOTE.TXT adresinde bulunmaktadır. İlki, .zip arşivindeki ilk dosyanın dosya üstbilgisi ve dosya verileridir, ardından her ek dosyanın dosya üstbilgisi ve dosya veri çifti gelir. (Dosya üstbilgisinin yapısı daha ileride anlatılmaktadır.) İkincisi, .zip dosyası isteğe bağlı olarak bir veri tanımlayıcı kaydı içerir (genellikle çıktı zip dosyası diske kaydedilmek yerine bellekte oluşturulduğunda). Üçüncü olarak, birçok ilave isteğe bağlı öğe bulunmaktadır: arşiv şifre çözme üstbilgisi, arşiv ek veri kaydı, merkezi dizin yapısı, Zip64 merkezi dizin kaydının sonu, Zip64 merkezi dizin bulucunun sonu ve merkezi dizin kaydının sonu.

Bu örnekteki kod yalnızca klasör içermeyen zip dosyalarının ayrıştırılması için yazılmıştır ve veri tanımlayıcı kayıtları beklemez. Son veri dosyasını izleyen tüm bilgileri yok sayar.

Her dosya için dosya üstbilgi biçimi şöyledir:

dosya üstbilgisi imzası	4 bayt
gerekli sürüm	2 bayt
genel amaçlı bit bayrağı	2 bayt
sıkıştırma yöntemi	2 bayt (8=DEFLATE; 0=SIKIŞTIRILMAMIŞ)
dosyanın son değiştirilme zamanı	2 bayt
dosyanın son değiştirilme tarihi	2 bayt
crc-32	4 bayt
sıkıştırılmış boyut	4 bayt
sıkıştırılmamış boyut	4 bayt
dosya adı uzunluğu	2 bayt
ek alan uzunluğu	2 bayt
dosya adı	değişken
ek alan	değişken

Dosya üstbilgisinin ardından gerçek dosya verileri gelir, bu veriler sıkıştırma yöntemi bayrağına bağlı olarak sıkıştırılmış veya sıkıştırılmamış olabilir. Dosya verileri sıkıştırılmamışsa bayrak 0'dır (sıfır), DEFLATE algoritması kullanılarak sıkıştırılmışsa 8'dir veya diğer sıkıştırma algoritmaları kullanılmışsa başka bir değerdir.

Bu örneğin kullanıcı arabirimi bir etiket ve metin alanından oluşur (`taFiles`). Uygulama aşağıdaki bilgileri .zip dosyasında karşılaştığı her dosyanın metin alanına yazar: dosya adı, sıkıştırılmış boyut ve sıkıştırılmamış boyut. Aşağıdaki MXML belgesi uygulamanın Flex versiyonuna ait kullanıcı arabirimini açıklıyor:

```
<?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>
```

Programın başlangıcı aşağıdaki görevleri gerçekleştirir:

- Gerekli sınıfları içeri aktarır

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- Kullanıcı arabirimini tanımlar Flash için

```
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);
```

- `bytes` ByteArray öğesini tanımlar

```
var bytes:ByteArray = new ByteArray();
```

- Dosya üstbilgisinden meta verileri saklamak için değişkenleri tanımlar

```
// 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;
```

- .zip dosyasını temsil etmek için File (`zfile`) ve FileStream (`zStream`) nesnelerini tanımlar ve dosyaların ayıklandığı .zip dosyasının konumunu belirtir—masaüstü dizininde “HelloAIR.zip” adlı bir dosya.

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

Flex'te, program kodu `init()` yöntemiyle başlar ve buna kök `mx:WindowedApplication` için `creationComplete` işleyicisi etiketi adı verilir.

```
// for Flex
private function init():void
{
```

Program .zip dosyasını OKUMA modunda açarak başlar.

```
zStream.open(zfile, FileMode.READ);
```

Ardından, sayısal alanların bayt sıralamasında en düşük değerli baytın önce geldiğini göstermek için `bytes` öğesinin `endian` özelliğini `LITTLE_ENDIAN` olarak ayarlar.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

Sonra `while()` ifadesi, dosya akışındaki geçerli konum dosya boyutundan büyük veya dosya boyutuna eşit olana kadar devam eden bir döngü başlatır.

```
while (zStream.position < zfile.size)
{
```

Döngüdeki ilk ifade dosya akışının ilk 30 baytını ByteArray `bytes` öğesine okur. İlk 30 bayt, ilk dosya üstbilgisinin sabit boyutlu parçasını oluşturur.

```
// read fixed metadata portion of local file header
zStream.readBytes(bytes, 0, 30);
```

Sonra kod, 30 baytlık üstbilginin ilk baytlarından bir tam sayı (`signature`) okur. ZIP biçimi tanımı, her dosya üstbilgi imzasının onaltılık değer `0x04034b50` olduğunu belirtir; imza farklıysa bu, kodun, .zip dosyasının dosya kısmının ötesine taşındığı ve ayıklanacak dosya olmadığı anlamına gelir. Bu durumda kod, bayt sırasının sonunu beklemek yerine hemen `while` döngüsünden çıkar.

```
bytes.position = 0;
signature = bytes.readInt();
// if no longer reading data files, quit
if (signature != 0x04034b50)
{
    break;
}
```

Kodun sonraki kısmı üstbilgi baytını uzaklık konumu 8'de okur ve değeri `compMethod` değişkeninde saklar. Bu bayt, bu dosyayı sıkıştırmak için kullanılan sıkıştırma yöntemini gösteren bir değer içerir. Birçok sıkıştırma yöntemi kullanılabilir, ancak uygulamada neredeyse tüm .zip dosyaları DEFLATE sıkıştırma algoritmasını kullanmaktadır. Geçerli dosya DEFLATE sıkıştırmayla sıkıştırıldıysa `compMethod` 8'dir; dosya sıkıştırılmamışsa, `compMethod` 0'dır.

```
bytes.position = 8;
compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

İlk 30 baytın ardından, üstbilginin dosya adını ve büyük olasılıkla ek bir alan içeren üstbilginin değişken uzunluklu kısmı gelir. Değişken `offset` bu kısmın boyutunu saklar. Boyut, uzaklık 26 ve 28'deki üstbilgilerden okunan dosya adı uzunluğu ve ek alan uzunluğu toplanarak hesaplanır.

```
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
```

Sonra program `offset` değişkeninde saklanan bayt sayısı için dosya üstbilgisinin değişken uzunluklu kısmını okur.

```
// read variable length bytes between fixed-length header and compressed file data
zStream.readBytes(bytes, 30, offset);
```

Program, dosya adını üstbilginin değişken uzunluk kısmından okur ve dosyanın sıkıştırılmış (zip) ve sıkıştırılmamış (orijinal) boyutlarıyla birlikte metin alanında görüntüler.

```
// 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';
```

Örnek, dosyanın geri kalanını sıkıştırılmış boyut tarafından belirtilen uzunluk için dosya akışından `bytes` öğesine okur ve ilk 30 bayttaki dosya üstbilgisinin üzerine yazar. Dosya sıkıştırılmış olmasa bile sıkıştırılmış boyut doğrudur, çünkü bu durumda sıkıştırılmış boyut dosyanın sıkıştırılmamış boyutuna eşittir.

```
    // 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);
```

Sonra örnek, sıkıştırılmış dosyayı genişletir ve çıktı dosyası akışına yazmak için `outfile()` işlevini çağırır. `outfile()` öğesine dosya verilerini içeren dosya adını ve bayt sırasını iletir.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

Daha önce bahsedilen örnekte, `bytes.uncompress(CompressionAlgorithm.DEFLATE)` öğesi yalnızca AIR uygulamalarında çalışır. Hem AIR hem de Flash Player için sönmüş verinin sıkıştırmasını açmak için ByteArray öğesinin `inflate()` işlevini çağırın.

Kapatma ayraçları, `outFile()` yöntemi haricinde, `while` döngüsünün, `init()` yönteminin ve uygulama kodunun sonunu gösterir. Çalıştırma `while` döngüsünün başlangıcına kadar geriye doğru ilerler ve başka bir dosyayı ayıklayarak veya son dosya işlenmişse .zip dosyasını işlemeyi sonlandırarak .zip dosyasında bulunan sonraki baytları işlemeye devam eder.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

`outfile()` işlevi bir çıktı dosyasını, dosyaya `filename` parametresi tarafından sağlanan ismi vererek masaüstünde YAZMA modunda açar. Ardından `data` parametresindeki dosya verilerini çıktı dosyası akışına (`outStream`) yazar ve dosyayı kapatır.

```
// 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();
}
```

Bölüm 42: Ağ iletişimi ve iletişim temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player veya AIR'de uygulamalar oluştururken, çoğunlukla uygulamanızın dışındaki kaynaklara erişme ihtiyacı duyarsınız. Örneğin, bir görüntü için İnternet web sunucusuna talep gönderebilir ve karşılığında görüntü verilerini alabilirsiniz. Ya da bir uygulama sunucusuyla bağlantısı bulunan priz bağlantısı üzerinden ileri-geri seri nesneler gönderebilirsiniz. Flash Player ve AIR API'leri, uygulamanızın bu alışverişe katılmasına izin veren birçok sınıf sağlar. Bu API'ler UDP, TCP, HTTP, RTMP ve RTMFP gibi iletişim kuralları için IP tabanlı ağ kurmayı destekler.

Bir ağ boyunca veri gönderip almak için aşağıdaki sınıflar kullanılabilir:

Sınıf	Desteklenen veri biçimleri	İletişim kuralları	Açıklama
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	Desteklenen veri türlerini yükler ve verileri bir görüntüleme nesnesine dönüştürür. Bkz. "Görüntüleme içeriğini dinamik olarak yükleme" sayfa 201.
URLLoader	Herhangi biri (metin, XML, ikili, vs.)	HTTP, HTTPS	Gelişigüzel veri biçimleri yükler. Uygulamanız verileri yorumlamaktan sorumludur. Bkz. "URLLoader sınıfını kullanma" sayfa 824
FileReference	Herhangi bir	HTTP	Dosyaları yükler ve indirir. Bkz. "FileReference sınıfını kullanma" sayfa 659
NetConnection	Video, ses, ActionScript Mesaj Biçimi (AMF)	HTTP, HTTPS, RTMP, RTMFP	Video, ses ve uzak nesne akışlarına bağlanır. Bkz. "Videoyla çalışma" sayfa 478.
Ses	Ses	HTTP	Desteklenen ses biçimlerini yükler ve oynatır. Bkz. "Harici ses dosyalarını yükleme" sayfa 447.
XMLSocket	XML	TCP	XMLSocket sunucusuyla XML mesajı alışverişi yapar. Bkz. "XML soketleri" sayfa 811.
Socket	Herhangi bir	TCP	Bir TCP soket sunucusuna bağlanır. Bkz. "İkili istemci soketleri" sayfa 806.
SecureSocket (AIR)	Herhangi bir	SSLv3 veya TLSv1'e sahip TCP	SSL veya TLS güvenliği gerektiren bir TCP soket sunucusuna bağlanır. Bkz. "Güvenli istemci soketleri (AIR)" sayfa 806.
ServerSocket (AIR)	Herhangi bir	TCP	Gelen TCP soket bağlantıları için sunucu olarak hareket eder. Bkz. "Sunucu soketleri" sayfa 814.
DatagramSocket (AIR)	Herhangi bir	UDP	UDP paketleri gönderip alır. Bkz. "UDP soketleri (AIR)" sayfa 816

Çoğu zaman, bir web uygulaması oluştururken kullanıcının uygulama durumu hakkında kalıcı bilgiler depolamak faydalıdır. HTML sayfaları ve uygulamaları bu amaçla tipik olarak çerezler kullanır. Flash Player'da, aynı amaç için SharedObject sınıfını kullanabilirsiniz. Bkz. "Paylaşılan nesneler" sayfa 709. (SharedObject sınıfı AIR uygulamalarında kullanılabilir, ancak düzenli bir dosyaya sadece veri kaydederken daha az sınırlama mevcuttur.)

Flash Player veya AIR uygulamasının aynı bilgisayardaki başka bir Flash Player veya AIR uygulamasıyla iletişim kurması gerekirse, LocalConnection sınıfını kullanabilirsiniz. Örneğin, aynı web sayfasındaki iki (veya daha fazla) SWF, birbiriyle iletişim kurabilir. Benzer şekilde, bir web sayfasında çalışan SWF bir AIR uygulamasıyla iletişim kurabilir. Bkz. "Diğer Flash Player ve AIR örneklerine bağlanma" sayfa 839.

Yerel bilgisayardaki SWF dışı diğer işlemlerle iletişim kurmanız gerektiğinde, AIR 2'ye eklenen NativeProcess sınıfını kullanın. NativeProcess sınıfı AIR uygulamanızın başlamasına ve diğer uygulamalarla iletişim kurmasına imkan verir. Bkz. "AIR uygulamasında yerel işlemlere bağlanma" sayfa 846

AIR uygulamasının çalışmakta olduğu bilgisayarın ağ ortamı hakkında bilgiye ihtiyaç duyduğunuzda, aşağıdaki sınıfları kullanabilirsiniz:

- NetworkInfo—Kullanılabilir ağ arabirimleri hakkında bilgisayarın IP adresi gibi bilgiler sağlar. Bkz. "Ağ arabirimleri" sayfa 799.
- DNSResolver—DNS kayıtlarına bakmanıza imkan verir. Bkz. "DNS (Alan Adı Sistemi) kayıtları" sayfa 802.
- ServiceMonitor—Bir sunucunun kullanılabilirliğini izlemenize imkan verir. Bkz. "Hizmet izleme" sayfa 800.
- URLMonitor—Bir kaynağın belli bir URL'deki kullanılabilirliğini izlemenize imkan verir. Bkz. "HTTP izleme" sayfa 801.
- SocketMonitor ve SecureSocketMonitor—Bir kaynağın bir soketteki kullanılabilirliğini izlemenize imkan verir. Bkz. "Soket izleme" sayfa 802.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi ağ ve iletişim kodu programlarken karşılaşacağınız önemli terimleri içerir.

Harici veriler Uygulamanın dışında belirli bir şekilde saklanan ve gerektiğinde uygulamaya yüklenen veriler. Bu veriler, doğrudan yüklenen bir dosyada saklanabildiği gibi, bir sunucuda çalıştırılan komut dosyaları veya programların çağrılmasıyla alınan bir veritabanında ya da başka bir formda saklanabilir.

URL olarak kodlanmış değişkenler URL olarak kodlanmış biçim, çeşitli değişkenleri (değişken ad ve değer çiftleri) tek bir metin dizesi halinde temsil etmenin bir yolunu sunar. Ayrı ayrı değişkenler name=value biçiminde yazılır. Her değişken (her ad-değer çifti) ampersan karakteri tarafından şu şekilde ayrılır: variable1=value1&variable2=value2. Bu şekilde, sonsuz sayıda değişken tek bir mesaj olarak gönderilebilir.

MIME türü Belirli bir İnternet iletişimi dosyasının türünü tanımlamak için kullanılan standart bir kod. Her dosya, kendisini tanımlamak için kullanılan belirli bir kod içerir. Dosya veya mesaj gönderilirken, bilgisayar (örn. web sunucusu veya kullanıcının Flash Player ya da AIR örneği) gönderilen dosya türünü belirtir.

HTTP Köprü Metni Aktarım Protokolü—web sayfalarını ve Internet üzerinden gönderilen diğer çeşitli içerik türlerini dağıtmak için standart biçim.

İstek yöntemi Bir uygulama (bir AIR uygulaması veya bir web tarayıcısı gibi) web sunucusuna mesaj (HTTP isteği adı verilir) gönderdiğinde, gönderilen tüm veriler iki yöntemden biri kullanılarak isteğe gömülebilir; bunlar GET ve POST olmak üzere iki istek yöntemidir. Sunucu ucunda, isteği alan programın verileri bulmak için isteğin ilgili bölümüne bakması gerekir, bu nedenle uygulamanızdan veri göndermek için kullanılan istek yönteminin, sunucuda o veriyi okumak için kullanılan istek yöntemiyle eşleşmesi gerekir.

Soket bağlantısı İki bilgisayar arasındaki iletişim için kalıcı bir bağlantı.

Karşıya yükleme Başka bir bilgisayara dosya gönderme.

İndirme Başka bir bilgisayardan dosya alma.

Ağ arabirimleri

Adobe AIR 2 ve üstü

NetworkInfo nesnesini uygulamanızda kullanabileceğiniz mevcut donanım ve yazılım ağlarını keşfetmek için kullanabilirsiniz. NetworkInfo nesnesi bir *singleton* nesnesidir. Bu nedenle bir tane oluşturmanıza gerek yoktur. NetworkInfo nesnesine ulaşmak için `onun yerine networkInfo` statik sınıf özelliğini kullanın. NetworkInfo nesnesi `ayrıca, kullanılabilir arabirimlerden biri de i ti inde networkChange` olayı gönderir.

NetworkInterface nesnelerinin listesi için`findInterfaces()` yöntemini çağırın. Listedeki her Each NetworkInterface nesnesi kullanılabilir arabirimlerden birini tanımlar. NetworkInterface nesnesi IP adresi, donanım adresi, maksimum iletim birimi ve arabirimin etkin olup olmadığıyla ilgili bilgiler sağlar.

Aşağıdaki kod örneği, istemci bilgisayarın her arabiriminin NetworkInterface özelliklerini izler:

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

Daha fazla bilgi için bkz.:

- NetworkInfo
- NetworkInterface
- InterfaceAddress
- Flexpert: Detecting the network connection type with Flex 4.5 (Flex 4.5 ile ağ bağlantısı türünü belirleme)

Ağ bağlanabilirliği değişimleri

Adobe AIR 1.0 ve üstü

AIR uygulamanız, güvenli olmayan ve değişken ağ bağlantıları içeren ortamlarda çalışabilir. Adobe AIR, bir uygulamanın çevrimiçi kaynaklara yönelik bağlantıları yönetmesine yardımcı olmak için, ağ bağlantısının her kullanılabilir veya kullanılamaz oluşunda bir ağ değişikliği olayı gönderir. Hem NetworkInfo nesnesi hem de uygulamanın NativeApplication nesnesi, `networkChange` olayını gönderir. Bu olaya tepki vermek için bir dinleyici ekleyin:

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

Ve bir olay işleyici işlevi tanımlayın:

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

`networkChange` olayı, tüm ağ etkinliğinde gerçekleşen bir değişikliği değil, yalnızca bir ağ bağlantısının değiştiğini gösterir. AIR, ağ değişikliğinin anlamını yorumlamaya çalışmaz. Ağa bağlı bir bilgisayar birçok gerçek ve sanal bağlantıya sahip olabilir. Bu nedenle bir bağlantının kaybedilmesi, kaynağın kaybedilmesi anlamına gelmeyebilir. Diğer yandan, yeni bağlantılar da iyileştirilmiş kaynak kullanılabilirliğini garanti etmez. Bazen yeni bir bağlantı, önceden kullanılabilir olan bir kaynağa erişimi bloke bile edebilir (örneğin VPN'e bağlanıldığında).

Genellikle, bir uygulamanın uzak bir kaynağa bağlanıp bağlanamayacağını belirlemenin tek yolu, denemektir. Hizmet izleme çerçevesi, ağ bağlanabilirliğindeki değişikliklere yanıt vermek için belirli bir ana bilgisayara olay bazlı araçlar sağlar.

Not: *Servis izleme çerçevesi, sunucunun bir isteği kabul edilebilir biçimde yanıtlayıp yanıtlamadığını saptar. Başarılı bir denetim, tam bağlanabilirliği garanti etmez. Ölçeklenebilir web servisleri, trafiği bir dizi web sunucusuna yönlendirmek için genellikle önbelleğe alma ve yük dengeleme gereçlerini kullanır. Bu durumda, servis sağlayıcıları ağ bağlantısına ilişkin yalnızca kısmi bir tanı sağlar.*

Hizmet izleme

Adobe AIR 1.0 ve üstü

Servis izleme çerçevesi, AIR çerçevesinden ayrı olarak aircore.swc dosyasında bulunur. Çerçeveyi kullanmak için, aircore.swc dosyası, inşa etme işlemine dahil edilmelidir.

Adobe® Flash® Builder bunu otomatik olarak dahil eder.

ServiceMonitor sınıfı, ağ servislerinin izlenmesi için çerçeveyi uygular ve servis izlemeleri için temel işlevler sağlar. Varsayılan olarak, bir ServiceMonitor sınıfı örneği, ağ bağlantısına ilişkin olaylar gönderir. ServiceMonitor nesnesi bu olayları, örnek oluşturulduğunda ve çalışma zamanının bir ağ değişimi algıladığı zamanlarda gönderir. Ayrıca, genel ağ bağlantısı olaylarına bakılmaksızın milisaniye cinsinden belirli bir aralıkta bağlantıyı kontrol etmek için ServiceMonitor örneğinin `pollInterval` özelliğini ayarlayabilirsiniz. ServiceMonitor nesnesi, `start()` yöntemi çağrılana kadar ağ bağlantısını kontrol etmez.

ServiceMonitor sınıfının alt sınıfı olan URLMonitor sınıfı, belirli bir URLRequest için HTTP bağlantısındaki değişiklikleri saptar.

ServiceMonitor sınıfının başka bir alt sınıfı olan SocketMonitor sınıfı ise, belirli bir bağlantı noktasındaki belirli bir ana bilgisayara ilişkin bağlantıdaki değişiklikleri saptar.

Not: *AIR 2'den önce, servis izleme çerçevesi servicemonitor.swc kütüphanesinde yayınlanıyordu. Bu kütüphane artık çıkartılmıştır. Onun yerine aircore.swc kütüphanesini kullanın.*

Flash CS4 ve CS5 Professional

Bu sınıfları Adobe® Flash® CS4 veya CS5 Professional'da kullanmak için:

1. File > Yayınlama ayarları komutunu seçin.
2. ActionScript 3.0 için Ayarlar düğmesini tıklatın. Kütüphane Yolu'nu seçin.
3. SWC'ye Gözat'ı tıklatın ve Flash Professional yükleme klasöründeki AIK klasörüne göz atın.
4. Bu klasörde, /frameworks/libs/air/aircore.swc (AIR 2 için) veya /frameworks/libs/air/servicemonitor.swc (AIR 1.5 için) öğesini bulun.
5. Tamam düğmesini tıklatın.
6. Aşağıdaki içe aktarma ifadesini ActionScript 3.0 kodunuza ekleyin:

```
import air.net.*;
```

Flash CS3 Professional

Adobe® Flash® CS3 Professional'da bu sınıfları kullanmak için ServiceMonitorShim bileşenini Bileşenler panelinden Kütüphane öğesine sürükleyin ve daha sonra aşağıdaki `import` ifadesini ActionScript 3.0 kodunuza ekleyin:

```
import air.net.*;
```

HTTP izleme

Adobe AIR 1.0 ve üstü

URLMonitor sınıfı, bağlantı noktası 80'de (HTTP iletişimi için tipik bağlantı noktası) belirli bir adrese HTTP isteklerinin yapılıp yapılamayacağını belirler. Aşağıdaki kod, Adobe web sitesindeki bağlantı değişikliklerini saptamak için bir URLMonitor sınıfı örneğini kullanır:

```
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);
}
```

Soket izleme

Adobe AIR 1.0 ve üstü

AIR uygulamaları, itme modeli bağlanabilirliği için soket bağlantıları da kullanabilir. Güvenlik duvarları ve ağ yönlendiricileri, güvenlik nedenleriyle genellikle yetkisiz bağlantı noktalarındaki ağ iletişimini kısıtlar. Bu nedenle geliştiriciler, kullanıcıların soket bağlantıları kuramayabileceğini göz önünde bulundurmalıdır.

Aşağıdaki kod, Adobe web sitesindeki bağlantı değişikliklerini saptamak için bir URLMonitor sınıfı örneğini kullanır. İzlenen bağlantı noktası, IRC'de yaygın bir bağlantı noktası olan 6667'dir:

```
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);
}
```

Soket sunucusu güvenli bir bağlantı gerektiriyorsa, SocketMonitor yerine SecureSocketMonitor sınıfını kullanabilirsiniz.

DNS (Alan Adı Sistemi) kayıtları

Adobe AIR 2.0 ve üstü

DNSResolver sınıfını kullanarak DNS kaynak kayıtlarını arayabilirsiniz. DNS kaynak kayıtları alan adının IP adresi ve IP adresinin alan adı gibi bilgiler verir. Aşağıdaki türlerdeki DNS kaynak kayıtlarını arayabilirsiniz:

- ARecord—Bir sunucunun IPv4 adresi
- AAAARecord—Bir ana bilgisayarın IPv6 adresi
- MXRecord (mail exchange record)—ana bilgisayarın mesaj alışverişi kaydı.
- PTRRecord—bir IP adresinin internet sunucu adresi
- SRVRecord—bir hizmetin hizmet kaydıdır.

Bir kayıt aramak için sorgulama dizesini kayıt türünü gösteren sınıf nesnesini DNSResolver nesnesinin `lookup()` yöntemine geçirin. Sorgulama dizesi kayıt türüne göre değişir:

Kayıt sınıfı	Sorgulama dizesi	Sorgulama dizesi örneği
ARecord	ana bilgisayar adı	“example.com”
AAAARecord	ana bilgisayar adı	“example.com”
MXRecord	ana bilgisayar adı	“example.com”
PTRRecord	IP adresi	“208.77.188.166”
SRVRecord	Service identifier: _service._protocol.host	“_sip._tcp.example.com”

Aşağıdaki kod örneği ana bilgisayarın “example.com” IP adresini arar.

```
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 );
        }
    }
}
```

Daha fazla bilgi için bkz.:

- DNSResolver
- DNSResolverEvent
- ARecord

- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

Bölüm 43: Soketler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Soket, iki bilgisayar işlemi arasında kurulan bir ağ bağlantısı türüdür. Tipik olarak, işlemler aynı İnternet İletişim Kuralı (IP) ağına bağlı iki farklı bilgisayarda çalışır. Ancak, bağlı işlemler özel "yerel ana bilgisayar" IP adresi kullanarak aynı bilgisayarda çalışıyor olabilir.

Adobe Flash Player istemci tarafı Taşıma Kontrolü İletişim Kuralı (TCP) soketlerini destekler. Bir Flash Player uygulaması, soket sunucusu olarak hareket eden başka bir işleme bağlanabilirken, diğer işlemlerden gelen bağlantı taleplerini kabul edemez. Diğer bir deyişle, bir Flash Player uygulaması TCP sunucusuna bağlanabilir, ancak bir tane gibi hareket edemez.

Flash Player API'si XMLSocket sınıfını da içerir. XMLSocket sınıfı, o iletişim kuralını anlayan bir sunucuyla XML mesajları alışverişi yapmaya imkan veren Flash Player'a özel bir iletişim kuralı kullanır. XMLSocket sınıfı ActionScript 1'de ortaya koyulmuştur ve hala geriye dönük uyumluluk sağlaması için destekleniyor. Genel olarak Socket sınıfı, Flash XMLSockets ile iletişim kurmak için özellikle oluşturulmuş bir sunucuya bağlanmadığınız sürece yeni uygulamalar için kullanılmalıdır.

Adobe AIR, soket tabanlı ağ programlaması için ilave sınıflar ekler. AIR uygulamaları ServerSocket sınıfıyla birlikte TCP soket sunucuları gibi hareket edebilir ve SecureSocket sınıfıyla birlikte SSL ya da TLS güvenliği gerektiren soket sunuculara bağlanabilir. AIR uygulamaları DatagramSocket sınıfıyla birlikte Evrensel Datagram İletişim Kuralı (UDP) mesajlarını da gönderip alabilir.

Daha fazla Yardım konusu

flash.net paketi

"Soketlere bağlanma" sayfa 1080

TCP soketleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İletim Denetimi Protokolü (TCP), kalıcı bir ağ bağlantısı üzerinden mesaj alışverişi yapılmasını sağlar. TCP, gönderilen mesajların doğru sırada alınacağını garanti eder (büyük ağ sorunlarını engeller). TCP bağlantıları bir "istemci" ve bir "sunucu" gerektirir. Flash Player istemci soketleri oluşturabilir. Adobe AIR ayrıca sunucu soketleri oluşturabilir.

Aşağıdaki ActionScript API'leri TCP bağlantıları sağlar:

- Soket — bir istemci uygulamasının sunucuya bağlanmasına izin verir. Socket sınıfı gelen bağlantıları dinleyemez.
- SecureSocket (AIR) — bir istemci uygulamasının güvenilir bir sunucuya bağlanmasına ve şifreli bağlantılara girmesine izin verir.
- ServerSocket (AIR) — bir uygulamanın gelen bağlantıları dinlemesine ve sunucu olarak davranmasına izin verir.
- XMLSocket — bir istemci uygulamasının XMLSocket sunucusuna bağlanmasına izin verir.

İkili istemci soketleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İkili soket bağlantısı, istemci ve sunucunun özellikle XML paketleri değişmesini gerektirmemesi dışında, XML soketine benzer. Bunun yerine, bağlantı verileri ikili bilgi halinde aktarabilir. Bu da, arasında posta sunucuları (POP3, SMTP ve IMAP) ve haber sunucularının da (NNTP) yer aldığı çok çeşitli hizmetlere bağlanmanızı sağlar.

Socket sınıfı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Socket sınıfı soket bağlantıları yapabilmenizi ve ham ikili veriler okuyup yazabilmenizi sağlar. Socket sınıfı, ikili protokoller kullanan sunucularla birlikte çalışıldığında yararlıdır. İkili soket bağlantıları kullanarak, POP3, SMTP, IMAP ve NNTP gibi farklı İnternet protokolleriyle etkileşime izin veren bir kod yazabilirsiniz. Bu etkileşim, uygulamalarınızın posta ve haber sunucularına bağlanmasını sağlar.

Flash Player, doğrudan bir sunucunun ikili protokolünü kullanarak o sunucuyla arabirim oluşturabilir. Bazı sunucular big-endian bayt sırasını, bazıları little-endian bayt sırasını kullanır. “Ağ bayt sırası” big-endian olduğundan, İnternet'teki çoğu sunucu big-endian bayt sırasını kullanır. Intel® x86 mimarisi tarafından kullanıldığından, little-endian bayt sırası sık kullanılır. Veri gönderen ve alan sunucunun bayt sırasıyla eşleşen endian bayt sırasını kullanmalısınız. IDataInput ve IDataOutput arabirimleri tarafından gerçekleştirilen tüm işlemler ve bu arabirimleri uygulayan sınıflar (ByteArray, Socket ve URLStream) varsayılan olarak big-endian biçiminde, başka bir deyişle, en önemli bayt önce olacak şekilde kodlanır. Bu varsayılan bayt sırası Java ve resmi ağ bayt sırası ile eşleşecek şekilde seçilir. Kullanılan big-endian veya little-endian bayt sırasını değiştirmek için, `endian` özelliğini `Endian.BIG_ENDIAN` veya `Endian.LITTLE_ENDIAN` olarak ayarlayabilirsiniz.

Soket sınıfı, (flash.utils paketinde yer alan) IDataInput ve DataOutput arabirimleri tarafından tanımlanan tüm yöntemleri üstlenir. Soketten yazmak ve okuma yapmak için bu yöntemler kullanılmalıdır.

Daha fazla bilgi için bkz.:

- Socket
- IDataInput
- IDataOutput
- socketData event

Güvenli istemci soketleri (AIR)

Adobe AIR 2 ve üstü

SecureSocket sınıfını, Güvenli Soket Katmanı sürüm 4 (Secure Sockets Layer - SSLv4) veya Taşıma Katmanı Güvenliği sürüm 1 (Transport Layer Security - TLSv1) kullanan soket sürücülerine bağlanmak için kullanabilirsiniz. Güvenli bir soketin üç avantajı vardır: sunucu onaylama, veri bütünlüğü ve mesaj gizliliği. Çalışma zamanı, kullanıcının güvenilir deposunda bir sunucu sertifikasını ve kökle veya ara sertifika yetkisiyle olan ilişkisini kullanarak bir sunucuyu yetkilendirir. Çalışma zamanı, veri bütünlüğü ve mesaj gizliliği sağlamak için SSL ve TLS protokol uygulamaları tarafından kullanılan şifreleme algoritmalarını temel alır.

SecureSocket nesnesini kullanarak sunucuya bağlandığınızda çalışma zamanı sertifika güven deposu kullanarak sunucu sertifikasını doğrular. Windows ve Mac üzerinde, işletim sistemi bir güven deposu sağlar. Linux'ta uygulamasının çalışma zamanı, kendi güven deposunu sağlar.

Sunucu sertifikası geçerli veya güvenilir değilse çalışma zamanı `ioError` olayı gönderir. Doğrulamanın başarısız olmasının nedenini kontrol etmek için SecureSocket nesnesinin `serverCertificateStatus` özelliğini kontrol edin. Geçerli ve güvenilir sertifikası olmayan bir sunucuya bağlanmak için gereken bir koşul yoktur.

CertificateStatus sınıfı, olası doğrulama sonuçlarını gösteren dize kalıp deyimlerini tanımlar:

- Süresi dolmuş - sertifikanın geçerlilik süresi dolmuş.
- Geçersiz - bir sertifikanın geçersiz olduğunu gösteren birkaç neden var. Örneğin, sertifika değiştirilmiş, bozulmuş olabilir veya yanlış türde bir sertifika olabilir.
- Geçersiz zincir - sunucunun sertifika zincirinde bir veya birden fazla sertifika geçersiz.
- Ana uyumsuzluk — Ana bilgisayarın adı ve sertifikanın ortak adı eşleşmiyor. Bir başka deyişle, sunucu yanlış sertifikayı kullanıyor.
- İptal edilmiş — sunulan sertifika yetkisi, sertifikayı iptal etti.
- Güvenilir — sertifika geçerli ve doğrulanmış. Bir SecureSocket nesnesi yalnızca geçerli, güvenilir bir sertifika kullanan bir sunucuya bağlanabilir.
- Bilinmiyor - SecureSocket nesnesi sertifikayı henüz doğrulamamış. `serverCertificateStatus` özelliğini, `connect()` öğesini çağırmanızdan önce ve `connect` veya bir `ioError` olayı gönderilmeden önce bu değer konumuna sahiptir.
- Güvenilmeyen imzacılar — sertifika, istemci bilgisayarın güven deposundaki güvenilir köke "zincir" bağlamaz.

SecureSocket nesnesi ile bağlanmak, güvenilir bir iletişim kuralı kullanan ve geçerli, güvenilir bir sertifikası bulunan bir sunucu gerektirir. Diğer açılardan SecureSocket nesnesi kullanma Soket nesnesi kullanmayla aynıdır.

SecureSocket nesnesi tüm platformlarda desteklenmez. SecureSocket sınıfının `isSupported` özelliğini, çalışma zamanının SecureSocket nesnesinin geçerli istemci bilgisayarda kullanılmasını destekleyip desteklemediğini sınamak amacıyla kullanın.

Daha fazla bilgi için bkz.:

- SecureSocket
- CertificateStatus
- IDataInput
- IDataOutput
- socketData event

TCP socket örneği: Telnet istemcisi oluşturma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Telnet örneği, uzak bir sunucuya bağlanıp Socket sınıfını kullanarak veri iletilmesi tekniklerini göstermektedir. Örnekte şu teknikler gösterilmektedir:

- Socket sınıfını kullanarak özel bir telnet istemcisi oluşturma
- ByteArray nesnesini kullanarak uzak sunucuya metin gönderme
- Uzak sunucudan alınan verileri işleme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Telnet uygulama dosyaları, Samples/Telnet klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
TelnetSocket.fla veya TelnetSocket.mxml	Flex (MXML) veya Flash (FLA) için kullanıcı arabirimi içeren ana uygulama dosyası.
TelnetSocket.as	Kullanıcı arabirimi mantığını sağlayan belge sınıfı (yalnızca Flash).
com/example/programmingas3/Telnet/Telnet.as	Uygulama için, uzak sunucuya bağlanma, veri gönderme, alma ve görüntüleme gibi Telnet istemcisi işlevlerini sağlar.

Telnet soket uygulamasına genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Ana TelnetSocket.mxml dosyası, uygulamanın tamamı için kullanıcı arabirimi (UI) oluşturmaktan sorumludur.

Bu dosya, belirtilen sunucuya bağlanmak için kullanıcı arabirimine ek olarak iki yöntemi tanımlar: `login()` ve `sendCommand()`.

Aşağıdaki kod, ana uygulama dosyasında ActionScript'i listeler:

```
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 = "";
}
```

Birinci kod satırı, özel com.example.programmingas.socket paketinden Telnet sınıfını içe aktarır. İkinci kod satırı, daha sonra `connect()` yöntemi tarafından başlatılacak olan `telnetClient` adında bir Telnet sınıfı örneğini bildirir. Ardından, `connect()` yöntemi bildirilir ve önceden bildirilmiş `telnetClient` değişkenini başlatır. Bu yöntem, kullanıcı tanımlı telnet sunucu adını, telnet sunucu bağlantı noktasını ve görüntüleme listesindeki TextArea bileşeninin bir başvurusunu iletir, bu başvuru, soket sunucusundan gelen metin yanıtlarını görüntülemek için kullanılır. `connect()` yönteminin son iki satırı, Panel için `title` özelliğini ayarlar ve Panel bileşenini etkinleştirir, böylece kullanıcının uzak sunucuya veri göndermesine olanak sağlanır. Kullanıcının komutlarını bir ByteArray nesnesi olarak uzak sunucuya göndermek için, ana uygulama dosyasındaki son yöntem olan `sendCommand()` kullanılır.

Telnet sınıfına genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Telnet sınıfı, uzak Telnet sunucusuna bağlanılmasından ve veri gönderilmesinden/alınmasından sorumludur.

Telnet sınıfı şu özel değişkenleri bildirir:

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

Birinci değişken olan `serverURL`, bağlanılacak kullanıcı tanımlı sunucu adresini içerir.

İkinci değişken olan `portNumber`, Telnet sunucusunun geçerli olarak üzerinde çalıştığı bağlantı noktası numarasıdır. Varsayılan olarak Telnet hizmeti, bağlantı noktası 23'te çalışır.

Üçüncü değişken olan `socket`, `serverURL` ve `portNumber` değişkenleri tarafından tanımlanmış sunucuya bağlanmayı deneyen bir Socket örneğidir.

Dördüncü değişken olan `ta`, Sahne Alanı'ndaki bir TextArea bileşeni örneğine başvurudur. Bu bileşen, uzak Telnet sunucusundan gelen yanıtları veya olası hata mesajlarını görüntülemek için kullanılır.

Son değişken olan `state`, Telnet istemcinizin hangi seçenekleri desteklediğini belirlemek için kullanılan sayısal bir değişkendir.

Daha önce gördüğünüz gibi, Telnet sınıfının yapıcı işlevi, ana uygulama dosyasındaki `connect()` yöntemi tarafından çağrılır.

Telnet yapıcısı üç parametre alır: `server`, `port` ve `output`. `server` ve `port` parametreleri, sunucu adını ve Telnet sunucusunun çalıştığı bağlantı noktası numarasını belirtir. Son parametre olan `output`, sunucu çıktısını kullanıcılara görüntüleneceğini Sahne alanındaki Textarea bileşen örneği için bir başvurudur.

```
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();
    }
}
```

Sokete veri yazma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Verileri bir soket bağlantısına yazmak için Soket sınıfındaki yazma yöntemlerinden herhangi birini çağırın. Bu yöntemler arasında `writeBoolean()`, `writeByte()`, `writeBytes()`, `writeDouble()` ve diğer yöntemler yer alır. Daha sonra verileri, `flush()` yöntemini kullanarak çıktı arabelleğinde boşaltın. Telnet sunucusunda, bayt dizisini parametre olarak alıp çıktı arabelleğine gönderen `writeBytes()` yöntemi kullanılarak soket bağlantısına veriler yazılır. `writeBytesToSocket()` yöntemi şöyledir:

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

Bu yöntem, ana uygulama dosyasının `sendCommand()` yöntemi tarafından çağrılır.

Soket sunucusundan mesajları görüntüleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Soket sunucusundan her mesaj alındığında veya bir olay gerçekleştiğinde, özel `msg()` yöntemi çağrılır. Bu yöntem, Sahne Alanı'nda TextArea öğesine bir dize ekler ve TextArea bileşeninin alta gitmesini sağlayan özel bir `setScroll()` yöntemini çağırır. `msg()` yöntemi şöyledir:

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

TextArea bileşeninin içeriklerini otomatik olarak kaydırmadıysanız, kullanıcıların sunucudan gelen en son yanıtı görmek için metin alanında kaydırma çubuklarını elle sürüklemesi gerekir.

TextArea bileşenini kaydırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`setScroll()` yöntemi, kullanıcının döndürülen metnin son satırını görebilmesi için TextArea bileşeninin içeriklerini dikey olarak kaydıran tek bir ActionScript satırı içerir. Aşağıdaki kod parçası, `setScroll()` yöntemini gösterir:

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

Bu yöntem, geçerli olarak görüntülenen karakterlerin en üst satırının satır numarası olan `verticalScrollPosition` özelliğini ayarlar ve bunu `maxVerticalScrollPosition` özelliğinin değerine ayarlar.

XML soketleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir XML soketi, belirgin bir şekilde kapatılana kadar açık kalan uzak sunucuya bağlantı kurmanıza izin verir. Sunucu ve istemci arasında, XML gibi dize verisi alış verişi yapabilirsiniz. XML soket sunucusu kullanmanın sağladığı faydalardan biri, istemcinin açık bir şekilde veri istemek zorunda kalmamasıdır. Sunucu istek beklemeden veri gönderebilir ve bağlı durumda olan her istemciye veri gönderebilir.

Flash Player'da ve uygulama sanal alanı dışındaki Adobe AIR içeriğinde, XML soketi bağlantıları hedef sunucuda bir soket ilke dosyası olmasını gerektirir. Daha fazla bilgi için, bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062 ve "Soketlere bağlanma" sayfa 1080.

XMLSocket sınıfı güvenlik duvarları arasından otomatik olarak tünel oluşturamaz, çünkü Real-Time Messaging Protocol'ün (RTMP) aksine XMLSocket, HTTP tüneli özelliğine sahip değildir. HTTP tüneli kullanmanız gerekiyorsa, bunun yerine Flash Remoting veya Flash Media Server (RTMP'yi destekler) kullanabilirsiniz.

Şu kısıtlamalar Flash Player veya uygulama güvenlik sanal alanı dışındaki AIR uygulamasındaki içeriğin bir XMLSocket nesnesini sunucuya bağlanmak için nasıl ve nerede kullanabileceğini belirtir:

- Uygulama güvenlik sanal alanının dışındaki içerik için `XMLSocket.connect()` yöntemi yalnızca 1024'ten büyük veya 1024'e eşit TCP bağlantı noktası numaralarına bağlanabilir. Bu kısıtlamanın bir sonucu, `XMLSocket` nesnesiyle iletişim kuran sunucu arka plan programlarının da 1024'ten büyük veya 1024'e eşit bağlantı noktası numaralarına atanmak zorunda olmasıdır. 1024'ün altındaki bağlantı noktası numaraları genellikle FTP (21), Telnet (23), SMTP (25), HTTP (80) ve POP3 (110) gibi sistem servisleri tarafından kullanılır, bu yüzden XMLSocket nesneleri güvenlik nedenleriyle bu bağlantı noktalarından engellenir. Bağlantı noktası numara kısıtlaması, bu kaynaklara uygun olmayan biçimlerde erişilmesi ve kaynakların kötüye kullanılması olasılığını azaltır.
- Uygulama güvenlik sanal alanının dışındaki içerik için, `XMLSocket.connect()` yöntemi yalnızca içeriğin yer aldığı etki alanının aynısında bulunan bilgisayarlara bağlanabilir. (Bu kısıtlama `URLLoader.load()` için güvenlik kurallarının aynısıdır.) İçeriğin yer aldığı etki alanından farklı bir etki alanında çalışan sunucu arka plan programına bağlanmak için, sunucuda belirli etki alanlarından erişime izin veren bir etki alanları arası politika dosyası oluşturabilirsiniz. Etki alanları arası ilke dosyaları hakkında ayrıntılar için bkz. "AIR güvenliği" sayfa 1088.

Not: *XMLSocket nesnesiyle iletişim kurmak üzere bir sunucu kurmak zor olabilir. Uygulamanız gerçek zamanlı etkileşim gerektirmiyorsa, XMLSocket sınıfı yerine URLLoader sınıfını kullanın.*

Soket bağlantısı üzerinden bir sunucuya XML aktarmak ve bu sunucudan XML almak için XMLSocket sınıfının `XMLSocket.connect()` ve `XMLSocket.send()` yöntemlerini kullanabilirsiniz. `XMLSocket.connect()` yöntemi, web sunucusu bağlantı noktasıyla bir soket bağlantısı oluşturur. `XMLSocket.send()` yöntemi, XML nesnesini soket bağlantısında belirtilen sunucuya iletir.

`XMLSocket.connect()` yöntemini çağırdığınızda, uygulama sunucuya bir TCP/IP bağlantısı açar ve aşağıdaki durumlardan biri oluşana kadar bağlantıyı açık tutar:

- XMLSocket sınıfı `XMLSocket.close()` yönteminin çağrılması.
- XMLSocket sınıfına başvuru bulunmaması.
- Flash Player uygulamasından çıkılır.
- Bağlantının kopması (örneğin, modem bağlantısının kesilmesi).

XMLSocket sınıfıyla bir sunucuya bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir soket bağlantısı oluşturmak için, soket bağlantı isteğini bekleyecek ve Flash Player veya AIR uygulamasına yanıt gönderecek bir sunucu tarafı uygulaması oluşturmalısınız. Bu tür bir sunucu tarafı uygulaması AIR'de veya Java, Python veya Perl gibi başka bir programlama dilinde yazılabilir. XMLSocket sınıfını kullanmak için sunucu bilgisayarı, XMLSocket sınıfı tarafından kullanılan basit protokolü anlayan bir arka plan programı çalıştırmalıdır:

- XML mesajları, tam çift yönlü bir TCP/IP akış soket bağlantısı üzerinden gönderilir.
- Her XML mesajı, sıfır (0) baytıyla sonlanan tam bir XML belgesidir.
- Tek bir XMLSocket bağlantısı üzerinden sınırsız sayıda XML mesajı gönderilebilir ve alınabilir.

Java XML soket sunucusu oluşturma ve bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki kod, gelen bağlantıları kabul eden ve komut istemi penceresinde alınan mesajları görüntüleyen, Java'da yazılmış basit bir XMLSocket sunucusunu gösterir. Sunucunuzu komut satırından başlatırken farklı bir bağlantı noktası numarası belirtebiliyor olsanız da, yeni bir sunucu varsayılan olarak yerel bilgisayarınızın 8080 bağlantı noktasında oluşturulur.

Yeni bir metin belgesi oluşturun ve şu kodu ekleyin:

```
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);
    }
}
```

Belgeyi sabit diskinize SimpleServer.java olarak kaydedin ve SimpleServer.class adlı bir Java sınıfı dosyası oluşturan Java derleyicisini kullanarak derleyin.

Bir komut istemini açarak ve `java SimpleServer` yazarak XMLSocket sunucusunu başlatabilirsiniz. SimpleServer.class dosyası yerel bilgisayarınızda veya ağda herhangi bir yerde konumlandırılabilir; web sunucunuzun kök dizinine yerleştirilmesi gerekmez.

Dosyalar Java sınıf yolu içinde bulunmadığından sunucuyu başlatamıyorsanız, sunucuyu şununla başlatmayı deneyin: `java -classpath . SimpleServer`.

Uygulamanızdan XMLSocket öğesine bağlanmak için, aşağıdaki gibi, yeni bir XMLSocket sınıfı örneği oluşturmanız ve bir ana bilgisayar adı ve bağlantı noktası numarası iletirken `XMLSocket.connect()` yöntemini çağırmanız gerekir:

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

Sunucudan her veri aldığınızda, `data` olayı (`flash.events.DataEvent.DATA`) gönderilir:

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

XMLSocket sunucusuna veri göndermek için, `XMLSocket.send()` yöntemini kullanıp bir XML nesnesi veya dizesi iletirsiniz. Flash Player, sağlanan parametreyi bir String nesnesine dönüştürür ve içeriği bir sıfır (0) baytın takip ettiği XMLSocket sunucusuna gönderir:

```
xmlsock.send(xmlFormattedData);
```

`XMLSocket.send()` yöntemi, verilerin başarıyla aktarılıp aktarılmadığını gösteren bir değer döndürmez. Veri gönderilmeye çalışılırken bir hata oluştuysa, bir IOError hatası verilir.

XML soket sunucusuna gönderdiğiniz her mesajın yeni satır (`\n`) karakteriyle sonlandırılması gerekir.

Daha fazla bilgi için bkz. XMLSocket

Sunucu soketleri

Adobe AIR 2 ve üstü

Diğer işlemlerin İletim Denetimi Protokolü (TCP) soketi kullanarak uygulamanıza bağlanmalarını sağlamak için ServerSocket sınıfını kullanın. Bağlanma işlemi yerel bir bilgisayarda veya ağa bağlanmış başka bir bilgisayarda çalıştırılabilir. Bir ServerSocket nesnesi bir bağlantı talebi aldığında bir `connect` olayı gönderir. ServerSocketConnectEvent olayı, Soket nesnesi içeren bir olayla gönderildi. Bu Soket nesnesini sonradan diğer işlemlerle iletişim kurmak için kullanabilirsiniz.

Gelen soket bağlantılarını dinlemek için:

1. Bir ServerSocket nesnesi oluşturun ve yerel bir bağlantı noktasına bağlayın
2. `connect` olayı için olay dinleyicileri ekleyin.
3. `listen()` yöntemini çağırın
4. Gelen her bağlantı için Soket nesnesi sağlayan `connect` olayını yanıtlayın.

ServerSocket nesnesi, siz `close()` yöntemini çağırıncaya kadar ek bağlantıları dinlemeye devam eder.

Aşağıdaki kod örneğinde soket sunucu uygulamasının nasıl oluşturulduğu gösterilmektedir. Örnek, bağlantı noktası 8087'de gelen bağlantıları dinler. Bir bağlantı alındığında örnek tarafından bir mesaj gönderilir ("Bağlandı" dizesi). istemci soketine Bu nedenle sunucu, istemciden geri gelen tüm iletileri ikinci kez gösterir.

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

Daha fazla bilgi için bkz.:

- ServerSocket
- ServerSocketConnectEvent
- Socket

UDP soketleri (AIR)

Adobe AIR 2 ve üstü

Evrensel Veri Protokolü (UDP), ağ bağlantısı üzerinden mesaj alışverişi yapılmasını sağlar. UDP, mesajların sırayla veya hepsinin dağıtıldığını garanti etmez. UDP ile, işletim sisteminin ağ kodu mesajları sıralamada (marshaling), izlemede ve onaylamada genellikle daha az zaman harcar. Bu nedenle UDP mesajları genellikle hedef uygulamaya TCP mesajlarınınkinden daha kısa bir gecikmeyle ulaşır.

UDP soket iletişimi, bir oyunda konum güncellemeleri veya sesli sohbet uygulamasındaki ses paketleri gibi gerçek zamanlı bilgi göndermek zorunda olduğunuz zamanlarda yararlıdır. Bu tür uygulamalarda bazı veri kayıpları kabul edilebilirdir. İletim gecikme süresinin düşük olması, iletilerin varmalarının garantili olmasından daha önemlidir. Tüm diğer amaçlar açısından TCP soketleri daha iyi bir tercihtir.

AIR uygulamanız DatagramSocket ve DatagramSocketDataEvent sınıflarıyla UDP mesajları gönderir ve alır. Bir UDP mesajı göndermek veya almak için:

1. Bir DatagramSocket nesnesi oluşturun
2. `data` olayı için olay dinleyici ekleyin
3. Soketi, `bind()` yöntemini kullanarak yerel bir IP adresine ve bağlantı noktasına bağlayın.
4. Mesajları, `send()` yöntemini çağırarak, IP adresinde ve hedef bilgisayarın bağlantı noktasına geçirerek gönderin.
5. `data` olayına cevap vererek iletileri alın. DatagramSocketDataEvent nesnesi, mesaj verileri içeren bir ByteArray nesnesinin bulunduğu bir olay için göndermiştir.

Aşağıdaki kod örneği, bir uygulamanın nasıl UDP mesajları gönderdiğini ve aldığını göstermektedir. Örnek, hedef bilgisayara "Merhaba" dizesinin yer aldığı tek mesaj gönderir. Alınan herhangi bir mesajın içeriğini de izler.

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

UDP (user data protocol: kullanıcı veri protokolü) soketlerini kullanırken aşağıdaki noktaları göz önünde bulundurmayı unutmayın.

- Tek bir veri paketi, gönderen ve alıcı arasındaki ağ arabiriminin veya herhangi bir ağ düğümlerinin en küçük maksimum iletim biriminden (MTU) daha büyük olamaz. send() yöntemine geçirilen ByteArray nesnesindeki bütün veriler tek bir paket olarak iletildi. (TCP'de büyük mesajlar ayrı paketlere bölünür).
- Gönderen ve hedef arasında hiçbir anlaşma yoktur. Hedef mevcut değilse veya belirtilen bağlantı noktasında aktif bir dinleyici yoksa iletiler hata vermeden atılır.
- `connect()` yöntemini kullandığınızda diğer kaynaklardan gelen mesajlar göz ardı edilir. Bir UDP bağlantısı yalnızca uygun paket filtrelemesi sağlar. Bu, hedef adreste veya bağlantı noktasında geçerli olması gereken bir dinleme işlemi olmadığı anlamına gelmez.
- UDP trafiği bir ağı sıkıştırmaz. Ağ yöneticileri, ağ tıkanıklığı olduğunda hizmet kalitesi kontrollerini uygulama ihtiyacı duyabilirler. (TCP'nin, ağ sıkışıklığının etkisini azaltmak için yerleşik trafik kontrolü vardır.

Daha fazla bilgi için bkz.:

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

IPv6 adresleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player 9.0.115.0 ve sonrası, IPv6'yı (Internet Protokolü sürüm 6) destekler. IPv6, 128 bit adresleri destekleyen bir Internet Protokolü sürümüdür (32 bit adresleri destekleyen önceki IPv4 protokolündeki bir geliştirme). Ağ arabirimleriniz üzerinde IPv6'yı etkinleştirmeniz gerekebilir. Daha fazla bilgi için verileri barındıran işletim sisteminin Yardım bölümüne bakın.

IPv6, barındıran sistem üzerinde destekleniyorsa, köşeli ayraç ([]) içindeki URL'lerde şu şekilde sayısal IPv6 değişmez adreslerini belirtebilirsiniz:

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

Flash Player, şu kurallara göre değişmez IPv6 değerlerini döndürür:

- Flash Player, IPv6 adresleri için uzun dize formunu döndürür.
- IP değeri çift iki nokta kısaltma içermez.
- Onaltılık basamaklar yalnızca küçük harf olur.
- IPv6 adresler, köşeli ayraç ([]) içine alınır.
- Her adres dörtlüsü, başındaki sıfır atılmış şekilde 0 - 4 onaltılık basamak olarak verilir.
- Aşağıdaki istisnalar listesinde belirtilenler dışında, tümü sıfır olan bir adres dörtlüsü tek bir sıfır olarak (çift iki nokta olarak değil) verilir.

Flash Player'ın döndürdüğü IPv6 değerleri şu istisnaları içerir:

- Belirtilmemiş bir IPv6 adresi (tümü sıfır) [::] olarak verilir.
- Geri döngü veya yerel ana bilgisayar IPv6 adresi, [::1] olarak verilir.
- IPv4 eşleşmiş (IPv6'ya dönüştürülmüş) adresleri, [::ffff:a.b.c.d] olarak verilir, burada a.b.c.d, tipik bir IPv4 noktalı ondalık değeridir.
- IPv4 uyumlu adresler, [::a.b.c.d] olarak verilir, burada a.b.c.d, tipik bir IPv4 noktalı ondalık değeridir.

Bölüm 44: HTTP iletişimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe® AIR® ve Adobe® Flash® Player uygulamaları veri, görüntü, video yüklemek ve mesaj alışverişinde bulunmak üzere HTTP tabanlı sunucularla iletişim kurabilir.

Daha fazla Yardım konusu

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

Harici verileri yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, harici kaynaklardan veri yüklenmesine yönelik mekanizmalar içerir. Bu kaynaklar metin dosyaları gibi statik içerik veya bir web komut dosyası tarafından oluşturulan dinamik içerik olabilir. Veriler çeşitli yollarla biçimlendirilebilir ve ActionScript verilere ilişkin kod çözme ve erişim işlevleri sağlar. Veri alma işleminin parçası olarak, harici sunucuya veri de gönderebilirsiniz.

URLRequest sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici veri yükleyen çoğu API, gerekli ağ talebinin özelliklerini tanımlamak için URLRequest sınıfını kullanır.

URLRequest özellikleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir URLRequest nesnesinin aşağıdaki özelliklerini her güvenlik sanal alanında ayarlayabilirsiniz:

Özellik	Açıklama
contentType	URL isteğiyle gönderilen her verinin MIME içerik türü. Herhangi bir contentType ayarlanmazsa, değerler `application/x-www-form-urlencoded` olarak gönderilir.
data	URL isteği ile aktarılacak olan verileri içeren bir nesne.
digest	Adobe® Flash® Player önbelleğine saklanacak (veya buradan alınacak) imzalanmış Adobe platformu bileşenini benzersiz şekilde tanımlayan bir dize.
method	Bir GET veya POST gibi HTTP istek yöntemi. (AIR uygulama etki alanında çalışan içerik, `"GET"` veya `"POST"` dışındaki dizeleri `method` özelliği olarak belirleyebilir. Tüm HTTP fiilleri olanaklıdır ve `"GET"` varsayılan yöntemdir. Bkz. "AIR güvenliği" sayfa 1088.)
requestHeaders	HTTP istemine eklenecek HTTP istem başlıkları. Flash Player'da ve uygulama güvenliği sanal alanı dışında çalışan AIR içeriğinde bazı başlıkların ayarlanmasına yönelik izinlerin sınırlandırıldığını unutmayın.
url	İstenecek URL'yi belirtir.

AIR'de, URLRequest sınıfının yalnızca uygulama güvenliği sanal alanında çalışan AIR içeriğinin kullanabileceği ek özelliklerini ayarlayabilirsiniz. Uygulama sanal alanındaki içerik, yeni URL şemalarını (`file` ve `http` gibi standart şemalara ek olarak) kullanarak URL'leri de tanımlayabilir.

Özellik	Açıklama
followRedirects	Yeniden yönlendirmelerin takip edilip (`true`, varsayılan değer) edilmeyeceğini (`false`) belirtir. Bu yalnızca AIR uygulama sanal alanında desteklenir.
manageCookies	HTTP protokol yığınının bu istek için tanımlama bilgilerini yönetmesi gerekip (`true`, varsayılan değer) gerekmediğini (`false`) belirtir. Bu özelliği ayarlamak yalnızca AIR uygulama sanal alanında desteklenir.
authenticate	Kimlik doğrulama isteklerinin bu istek için işlenmesi gerekip (`true`) gerekmediğini belirtir. Bu özelliği ayarlamak yalnızca AIR uygulama sanal alanında desteklenir. Varsayılan, istekler için kimlik doğrulaması yapmaktır—sunucu kimlik bilgilerini gerektiriyorsa bu, kimlik doğrulama iletişim kutusunun görüntülenmesine neden olabilir. URLRequestDefaults sınıfını kullanarak kullanıcı adı ve şifreyi de ayarlayabilirsiniz—bkz. "URLRequest varsayılanlarını ayarlama (yalnızca AIR)" sayfa 821.
cacheResponse	Bu isteğe ilişkin yanıt verilerinin önbelleğe alınması gerekip gerekmediğini belirtir. Bu özelliği ayarlamak yalnızca AIR uygulama sanal alanında desteklenir. Varsayılan, yanıtı önbelleğe almaktır (`true`).
useCache	Bu URLRequest verileri almadan önce, yerel önbelleğe başvurulması gerekip gerekmediğini belirler. Bu özelliği ayarlamak yalnızca AIR uygulama sanal alanında desteklenir. Varsayılan (`true`), kullanılabiliyorsa yerel önbelleğe alınmış sürümü kullanmaktır.
userAgent	HTTP isteğinde kullanılacak kullanıcı aracısı dizesini belirtir.

Not: *HTMLLoader sınıfı, HTMLLoader nesnesi tarafından yüklenen içeriğe ait ayarlara ilişkin özellikler içerir. Ayrıntılar için bkz. "HTMLLoader sınıfı hakkında" sayfa 991.*

URLRequest varsayılanlarını ayarlama (yalnızca AIR)

Adobe AIR 1.0 ve üstü

URLRequestDefaults sınıfı URLRequest nesneleri için uygulamaya özel varsayılan ayarlarını tanımlamanızı sağlar. Örneğin, aşağıdaki kod `manageCookies` ve `useCache` özellikleri için varsayılan değerleri ayarlar. Tüm yeni URLRequest nesneleri normal varsayılanlar yerine bu özellikler için belirtilen değerleri kullanır:

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

Not: *URLRequestDefaults sınıfı Adobe AIR uygulamasında çalışan içerik için tanımlanır. Bu Flash Player'da desteklenmez.*

URLRequestDefaults sınıfı, belirli bir ana bilgisayar için kullanmak üzere varsayılan bir kullanıcı adı ve parola belirlemenizi sağlayan `setLoginCredentialsForHost()` yöntemini içerir. Yöntemin hostname parametresinde tanımlanan ana bilgisayar, `"www.example.com"` gibi bir etki alanı veya `"www.example.com:80"` gibi bir etki alanı ve bir bağlantı noktası numarası olabilir. `"example.com"`, `"www.example.com"` ve `"sales.example.com"` öğelerinden her birinin benzersiz ana bilgisayarlar olarak dikkate alındığını unutmayın.

Bu kimlik bilgileri, yalnızca sunucu onları istediğinde kullanılır. Kullanıcı kimliği doğrulanmışsa (örneğin, kimlik doğrulama iletişim kutusunu kullanarak), `setLoginCredentialsForHost()` yöntemini çağırmak, doğrulanmış kullanıcıyı değiştirmez.

Aşağıdaki kod, www.example.com adresine gönderilen taleplerde kullanılacak varsayılan kullanıcı adı ve şifreyi belirler:

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

URLRequestDefaults ayarları bir istisna haricinde yalnızca geçerli uygulama etki alanı için geçerlidir. `setLoginCredentialsForHost()` yöntemine aktarılan kimlik bilgileri, AIR uygulaması içindeki herhangi bir uygulama etki alanında yapılan talepler için kullanılır.

Daha fazla bilgi için, Adobe Flash Platformu için ActionScript 3.0 Başvurusu*'ndaki URLRequestDefaults sınıfına bakın.*

URI şemaları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki gibi standart URI şemaları herhangi bir güvenlik sanal alanından yapılan taleplerde kullanılabilir:

http: ve https:
Standart İnternet URL'leri için bunları kullanın (bir web tarayıcısında kullanıldıkları şekilde).

file:
Yerel dosya sisteminde bulunan bir dosyanın URL'sini belirtmek için `file:` kullanın. Örneğin:

```
file:///c:/AIR Test/test.txt
```

AIR'de, uygulama güvenlik sanal alanında çalışan içeriğe ilişkin URL'yi tanımlarken aşağıdaki şemaları da kullanabilirsiniz:

app:
Yüklü uygulamanın kök dizinine ilişkin bir yol belirlemek için `app:` öğesini kullanın. Örneğin, aşağıdaki yol yüklenen uygulama dizininin kaynaklar alt dizinine işaret eder:

```
app:/resources
```

Bir AIR uygulaması AIR Debug Launcher (ADL) kullanılarak başlatıldığında, uygulama dizini uygulama açıklayıcı dosyasını içeren dizindir.

`File.applicationDirectory` öğesi ile oluşturulan bir File nesnesi URL'si (ve `url` özelliği), `app` URI şemasını aşağıdaki gibi kullanır:

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:

Uygulamanın saklama dizinine ilişkin bir yol belirlemek için `app-storage:` öğesini kullanın. AIR her yüklenen uygulama için (ve kullanıcı), o uygulamaya özgü verilerin saklanmasına yönelik kullanışlı bir yer olan benzersiz bir uygulama depolama dizini tanımlar. Örneğin aşağıdaki yol, uygulama depolama dizininin ayarlar alt dizininde bulunan bir prefs.xml dosyasını işaret eder:

```
app-storage:/settings/prefs.xml
```

`File.applicationStorageDirectory` kullanılarak oluşturulan bir File nesnesine ilişkin URL (ve `url` özelliği) `app-storage` URI şemasını aşağıdaki şekilde kullanır:

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto:

`navigateToURL()` işlevine aktarılan URLRequest nesnelerindeki mailto şemasını kullanabilirsiniz. Bkz. "Başka bir uygulamada URL açma" sayfa 836.

FileStream veya Sound nesnesi gibi çeşitli nesnelere ilişkin URL isteği tanımlamak için bu URI şemalarından birini kullanan bir URLRequest nesnesini kullanabilirsiniz. Bu şemaları AIR'de çalışan HTML içeriğinde de kullanabilirsiniz; örneğin, bir `img` etiketinin `src` niteliğinde kullanabilirsiniz.

Ancak, AIR'ye özgü bu URL şemalarını (`app:` ve `app-storage:`) yalnızca uygulama güvenlik sanal alanındaki içerikte kullanabilirsiniz. Daha fazla bilgi için bkz. "AIR güvenliği" sayfa 1088.

URL değişkenlerini ayarlama

Değişkenleri URL dizesine doğrudan ekleyebilmenize rağmen, bir talep için gereken herhangi bir değişkeni tanımlamak için URLVariables sınıfını kullanmak daha kolay olabilir.

Bir URLVariables nesnesine parametre eklemenin üç yolu vardır:

- URLVariables yapıcısında
- `URLVariables.decode()` yöntemiyle
- URLVariables nesnesinin kendi içinde dinamik özellikler halinde

Aşağıdaki örnekte üç yöntem de gösterilmekte ve bir URLRequest nesnesine nasıl değişken atanacağı örneklendirilmektedir:

```
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;
```

Değişkenleri URLVariables yapıcısı içinde veya `URLVariables.decode()` yönteminde tanımlarken, bir URI dizesinde özel bir anlamı olan karakterleri URL kodlamasına tabi tuttuğunuzdan emin olun. Örneğin, bir parametre adı ya da değerde ampersan kullandığınızda, ampersanı `&`'den `%26`'ya değiştirerek kodlamanız gerekir, çünkü ampersan parametreler için bir sınırlayıcı olarak hareket eder. En üst seviye `encodeURIComponent()` işlevi bu amaçla kullanılabilir.

URLLoader sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

URLLoader sınıfı, bir sunucuya talep göndermenize ve verilen bilgilere ulaşmanıza izin verir. URLLoader sınıfını yerel dosya erişimine izin verilen bağlamlardaki (Flash Player dosya sistemiyle yerel sanal alanı ve AIR uygulaması sanal alanı gibi) yerel dosya sistemi üzerinde bulunan dosyalara erişmek için de kullanabilirsiniz. URLLoader sınıfı bir URL'den metin, ikili veri veya URL olarak kodlanmış değişkenler olarak veri indirir. URLLoader sınıfı `complete`, `httpStatus`, `ioError`, `open`, `progress` ve `securityError` gibi olaylar gönderir.

ActionScript 3.0 olay işleme modeli, `LoadVars.onData`, `LoadVars.onHTTPStatus` ve `LoadVars.onLoad` olay işleyicilerini kullanan ActionScript 2.0 modelinden büyük ölçüde farklıdır. ActionScript 3.0'da olayları işleme hakkında daha fazla bilgi edinmek için bkz. "Olayları işleme" sayfa 126

İndirilen veriler, indirme işlemi tamamlanana kadar kullanılamaz. `progress` olayının gönderilmesini dinleyerek indirmenin ilerlemesini (yüklenen baytlar ve toplam bayt) denetleyebilirsiniz. Ancak, bir dosya yeterince hızlı yüklenirse `progress` olayı gönderilmeyebilir. Bir dosya başarıyla yüklendiğinde `complete` olayı gönderilir. URLLoader `dataFormat` özelliğini ayarlayarak, verileri metin, ham ikili veri veya bir URLVariables nesnesi olarak alabilirsiniz.

`URLLoader.load()` yöntemi (ve isteğe bağlı olarak URLLoader sınıfının yapıcısı), URLRequest nesnesi olan tek bir parametre (`request`) alır. URLRequest nesnesi, hedef URL, istek yöntemi (`GET` veya `POST`), ek başlık bilgileri ve MIME türü gibi tek bir HTTP isteğinin tüm bilgilerini içerir.

Örneğin, sunucu tarafı komut dosyasına bir XML paketi yüklemek için aşağıdaki kodu kullanabilirsiniz:

```
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);
```

Önceki kod parçası, sunucuya gönderilecek XML paketini içeren `dataXML` adında bir XML örneği oluşturur. Örnek, URLRequest `contentType` özelliğini `"text/xml"` olarak ayarlar ve XML belgesini URLRequest `data` özelliğine atar. Son olarak, örnek bir URLLoader nesnesi oluşturup `load()` yöntemini kullanarak uzak komut dosyasına talebi gönderir.

URLStream sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

URLStream sınıfı indirme verileri geldiğinde bu verilere erişim sağlar. URLStream sınıfı ayrıca, indirmeyi tamamlamadan önce bir akışı kapatmanıza olanak verir. İndirilen veri, ham ikili veri olarak mevcuttur.

Bir URLStream nesnesinden veri okurken, okumadan önce yeterli verinin olup olmadığını belirlemek için `bytesAvailable` özelliğini kullanın. Mevcut olandan daha fazla veri okumaya çalışırsanız, EOFError istisnası atılır.

httpResponseStatus olayı (AIR)

Adobe AIR'de, URLStream sınıfı `httpStatus` olayına ek olarak bir `httpResponseStatus` olayı gönderir. `httpResponseStatus` olayı herhangi bir yanıt verisinden önce ulaştırılır. `httpResponseStatus` olayı (HTTPStatusEvent sınıfıyla ifade edilen), yanıtın döndüğü kaynak URL olan `responseURL` özelliğini ve yanıtın döndürdüğü yanıt başlıklarını temsil eden URLRequestHeader nesnesinin bir dizisi olan `responseHeaders` özelliğini içerir.

Harici belgelerden veri yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dinamik uygulamalar oluşturduğunuzda, harici dosyalardan veya sunucu tarafı komut dosyalarından veri yüklemek faydalı olabilir. Bu, uygulamanızı düzenlemeniz veya yeniden derlemeniz gerekmeden dinamik uygulamalar oluşturmanıza olanak sağlar. Örneğin, bir "günün ipucu" uygulaması oluşturuyorsanız, bir veritabanından rastgele bir ipucunu alıp her gün bir metin dosyasına kaydeden sunucu tarafı komut dosyası yazabilirsiniz. Daha sonra uygulamanız, her seferinde veritabanını sorgulamak yerine statik metin dosyasının içeriklerini yükleyebilir.

Aşağıdaki kod parçası, params.txt adında harici bir metin dosyasının içeriklerini yükleyen bir URLRequest ve URLLoader nesnesi oluşturur:

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

Varsayılan olarak, bir istek yöntemi tanımlamazsanız, Flash Player ve Adobe AIR uygulaması HTTP `GET` yöntemini kullanarak içeriği yükler. Verileri `POST` yöntemini kullanarak göndermek için, aşağıdaki kodun gösterdiği gibi `URLRequestMethod.POST` statik sabitini kullanarak `request.method` özelliğini `POST` olarak ayarlayın:

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

Çalışma zamanında yüklenen params.txt harici belgesi aşağıdaki verileri içerir:

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,Dece
mber&dayNames=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Dosya iki parametre içerir, `monthNames` ve `dayNames`. Her parametre, virgülle ayrılmış dizeler halinde ayrıştırılmış bir liste içerir. `String.split()` yöntemini kullanarak bu listeyi bir diziye ayrıştırabilirsiniz.

Harici veri dosyalarında değişken adı olarak ayrılmış sözcükleri veya dil yapılarını kullanmayın, aksi takdirde bu, kodunuzun okunmasını ve hatalarının ayıklanmasını daha güçleştirir.

Veriler yüklendiğinde `complete` olayı gönderilir ve aşağıdaki kodun gösterdiği gibi, harici belgenin içindekiler URLLoader `data` özelliğinde kullanıma hazırdır:

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

Uzak belge ad-değer çiftleri içeriyorsa, yüklenen dosyanın içeriğini aşağıdaki gibi ileterek URLVariables sınıfı aracılığıyla verileri ayrıştırabilirsiniz:

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

Harici dosyadan her ad-değer çifti URLVariables nesnesinde bir özellik olarak oluşturulmuştur. Önceki kod örneğinde yer alan değişkenler nesnesindeki her özellik, bir dize olarak değerlendirilir. Ad-değer çiftinin değeri bir öğeler listesiyse, dizeyi aşağıdaki gibi `String.split()` yöntemini çağırarak bir diziye dönüştürebilirsiniz:

```
var dayNameArray:Array = variables.dayNames.split(",");
```

Harici metin dosyalarından sayısal veriler yüklüyorsanız, `int()`, `uint()` veya `Number()` gibi bir üst düzey işlev kullanarak değerleri sayısal değerlere dönüştürün.

Uzak dosyanın içeriklerini bir dize olarak yükleyip yeni bir URLVariables nesnesi oluşturmak yerine, `URLLoader.dataFormat` özelliğini URLLoaderDataFormat sınıfında bulunan statik özelliklerden birine ayarlayabilirsiniz. `URLLoader.dataFormat` özelliği için üç olası değer şunlardır:

- `URLLoaderDataFormat.BINARY`—`URLLoader.data` özelliği, bir ByteArray nesnesinde saklanan ikili verileri içerir.
- `URLLoaderDataFormat.TEXT`—`URLLoader.data` özelliği, bir String nesnesinde metin içerir.
- `URLLoaderDataFormat.VARIABLES`—`URLLoader.data` özelliği, bir URLVariables nesnesinde saklanan URL olarak kodlanmış değişkenleri içerir.

Aşağıdaki kod, `URLLoader.dataFormat` özelliğinin `URLLoaderDataFormat.VARIABLES` olarak ayarlanmasının, yüklenen verileri nasıl otomatik olarak bir URLVariables nesnesine ayrıştırmanıza olanak sağladığını gösterir:

```
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);
        }
    }
}
```

Not: *`URLLoader.dataFormat` için varsayılan değer `URLLoaderDataFormat.TEXT` şeklindedir.*

Aşağıdaki örneğin gösterdiği gibi, harici bir dosyadan XML yüklemek, URLVariables yüklemekle aynıdır. Bir URLRequest örneği ile bir URLLoader örneği oluşturabilir ve bunları uzak bir XML belgesini indirmek için kullanabilirsiniz. Dosya tamamen indirildiğinde, `Event.COMPLETE` olayı gönderilir ve harici dosyanın içerikleri, XML yöntemlerini ve özelliklerini kullanarak ayrıştırabileceğiniz bir XML örneğine dönüştürülür.

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

Harici komut dosyalarıyla iletişim kurma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici veri dosyaları yüklemenin yanı sıra, sunucu tarafı komut dosyasına değişkenler gönderip sunucunun yanıtını işlemek için de URLVariables sınıfını kullanabilirsiniz. Bu örneğin, bir oyun programlıyor ve kullanıcının puanını yüksek puanlar listesine eklenip eklenmeyeceğinin hesaplanması için sunucuya göndermek ve hatta kullanıcının oturum açma bilgilerini doğrulama için sunucuya göndermek istiyorsanız, kullanışlı bir özelliktir. Sunucu tarafı komut dosyası kullanıcı adı ve parolayı işleyebilir, veritabanına karşı doğrulayabilir ve kullanıcının girdiği kimlik bilgilerinin geçerli olup olmadığına ilişkin teyit döndürebilir.

Aşağıdaki kod parçası, `name` adında yeni bir değişken oluşturan `variables` adında bir URLVariables nesnesi oluşturur. Daha sonra, değişkenlerin gönderileceği sunucu tarafı komut dosyasının URL'sini belirten bir URLRequest nesnesi oluşturulur. Daha sonra, değişkenleri bir HTTP `POST` isteği olarak göndermek üzere URLRequest nesnesinin `method` özelliğini ayarlarsınız. URLVariables nesnesini URL isteğine eklemek için, URLRequest nesnesinin `data` özelliğini önceden oluşturulan URLVariables nesnesine ayarlarsınız. Son olarak, URLLoader örneği oluşturulur ve isteği başlatan `URLLoader.load()` yöntemi çağrılır.

```
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);
}
```

Aşağıdaki kod, önceki örnekte kullanılan Adobe ColdFusion® greeting.cfm belgesinin içeriklerini barındırır:

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

Web hizmeti talepleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çok çeşitli HTTP tabanlı web hizmeti mevcuttur. Başlıca türleri şunlardır:

- REST
- XML-RPC
- SOAP

ActionScript 3'te bir web hizmetini kullanmak için bir URLRequest nesnesi oluşturun, URL değişkenlerini veya bir XML belgesini kullanarak web hizmeti çağrısını oluşturun ve bir URLLoader nesnesi kullanarak çağrıyı hizmete gönderin. Flex çerçevesi web hizmetlerini kullanmayı kolaylaştıran birçok sınıf içerir—özellikle karmaşık SOAP hizmetlerine erişirken faydalıdır. Flash Professional CS3'ten itibaren, Flex sınıflarını Flash Professional ile geliştirilen uygulamalarda ve Flash Builder'da geliştirilen uygulamalarda kullanabilirsiniz.

HTML tabanlı AIR uygulamalarında, URLRequest ve URLLoader sınıflarını veya Javascript XMLHttpRequest sınıfını kullanabilirsiniz. İsterseniz, Flex çerçevesinin web hizmeti bileşenlerini Javascript kodunuza açan bir SWF kütüphanesi de oluşturabilirsiniz.

Uygulamanız bir tarayıcıda çalışırken, web hizmetini barındıran sunucu aynı zamanda diğer etki alanlarından erişime izin veren bir çapraz etki alanı ilke dosyası barındırmıyorsa, web hizmetlerini yalnızca çağıran SWF ile aynı İnternet etki alanında kullanabilirsiniz. Bir çapraz etki alanı ilke dosyası bulunmadığında çoğunlukla kullanılan bir teknik de talepleri kendi sunucunuzdan devretmektir. Adobe Blaze DS ve Adobe LiveCycle web hizmeti devretmeyi destekler.

AIR uygulamalarında, web hizmeti çağrısı uygulama güvenliği sanal alanından geliyorsa bir çapraz etki alanı ilke dosyası gerekmez. AIR uygulamasının içeriği hiçbir zaman uzak bir etki alanından sunulmaz; bu yüzden, çapraz etki alanı ilkelerinin önlediği saldırı türlerine katılamaz. HTML tabanlı AIR uygulamalarında, uygulama güvenliği sanal alanındaki içerik çapraz etki alanı XMLHttpRequests oluşturabilir. İçeriğin iframe'e yüklü olması şartıyla, diğer güvenlik sanal alanlarındaki içeriğin de XMLHttpRequests oluşturmasına izin verebilirsiniz.

Daha fazla Yardım konusu

"Web sitesi denetimleri (ilke dosyaları)" sayfa 1062

Adobe BlazeDS

Adobe LiveCycle ES2

REST mimarisi

XML-RPC

SOAP iletişim kuralı

REST tarzı web hizmeti talepleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

REST tarzı web hizmetleri temel eylemleri belirleyecek HTTP yöntemi fiilleri ve eylem ayrıntılarını belirtecek URL değişkenleri kullanır. Örneğin, bir öğeyi alma talebi GET fiilini ve bir yöntem adı ile öğe kimliği belirtmek üzere URL değişkenlerini kullanabilir. Ortaya çıkan URL dizesi şunun gibi olabilir:

```
http://service.example.com/?method=getItem&id=d3452
```

REST tarzı bir web hizmetine ActionScript ile erişmek için, URLRequest, URLVariables ve URLLoader sınıflarını kullanabilirsiniz. Bir AIR uygulamasındaki Javascript kodunda bir XMLHttpRequest de kullanabilirsiniz.

ActionScript'te bir REST tarzı web hizmeti çağrısı programlamak tipik olarak aşağıdaki adımları içerir:

1. Bir URLRequest nesnesi oluşturun.
2. Talep nesnesinin hizmet URL'sini ve HTTP yöntem fiilini belirleyin.
3. Bir URLVariables nesnesi oluşturun.
4. Hizmet çağrısı parametrelerini variables nesnesinin dinamik özellikleri olarak ayarlayın.
5. variables nesnesini request nesnesinin data özelliğine atayın.
6. Bir URLLoader nesnesiyle çağrıyı hizmete gönderin.
7. URLLoader tarafından hizmet çağrısının tamamlandığını belirtmek üzere gönderilen `complete` olayını işleyin. Bir URLLoader nesnesi tarafından gönderilebilecek çeşitli hata olaylarını dinlemek de akıllıcadır.

Örneğin, çağrı parametrelerini talep edene geri ileten bir test yöntemini açığa çıkaran bir web hizmeti düşünün. Hizmeti çağırmak için aşağıdaki ActionScript kodu kullanılabilir:

```
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 );
}
```

Bir AIR uygulaması içindeki Javascript'te, aynı talebi XMLHttpRequest nesnesini kullanarak yapabilirsiniz:

```
<html>
<head><title>RESTful web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    //Create a conveninece object to hold the call properties
    var request = {};
    request.URL = "http://service.example.com/";
    request.method = "test.echo";
    request.HTTPmethod = "GET";
    request.parameters = {};
    request.parameters.api_key = "ABCDEF123";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestURL = makeURL( request );
    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, requestURL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerHTML = xmlhttp.responseText;
        }
    }
    xmlhttp.send(null);

    requestDisplay.innerHTML = requestURL;
}
//Convert the request object into a properly formatted URL
function makeURL( request )
{
    var url = request.URL + "?method=" + escape( request.method );
    for( var property in request.parameters )
    {
        url += "&" + property + "=" + escape( request.parameters[property] );
    }

    return url;
}
</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<div id="request"></div>
<h1>Result:</h1>
<div id="result"></div>
</body>
</html>
```

XML-RPC web hizmeti talepleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir XML-RPC web hizmeti tüm çağrı parametrelerini URL değişkenleri yerine bir XML belgesi olarak alır. Bir XML-RPC web hizmetiyle işlem yapmak için, doğru şekilde biçimlendirilmiş bir XML mesajı oluşturun ve HTTP `POST` yöntemini kullanarak web hizmetine gönderin. Bunun yanında, sunucunun talep verilerini XML olarak algılaması için, talebin `Content-Type` başlığını ayarlamanız gerekir.

Aşağıdaki örnekte, REST örneğinde gösterilen, ancak bu kez bir XML-RPC hizmeti olan aynı web hizmeti çağrısının nasıl kullanılacağı gösterilmektedir:

```
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 );
}
```

AIR içerisindeki WebKit E4X söz dizimini desteklemediğinden, önceki örnekte XML belgesini oluşturmada kullanılan yöntem Javascript kodunda çalışmaz. Bunun yerine, XML belgesini oluşturmak için DOM yöntemlerini veya belgeyi bir dize olarak oluşturup dizeyi XML'e dönüştürmek için Javascript DOMParser sınıfını kullanmanız gerekir.

Aşağıdaki örnek, bir XML-RPC mesajı oluşturmak için DOM yöntemleri, web hizmeti işlemini yapmak içinse bir XMLHttpRequest kullanır:

```
<html>
<head>
<title>XML-RPC web service request</title>
<script type="text/javascript">

function makeRequest()
{
    var requestDisplay = document.getElementById( "request" );
    var resultDisplay  = document.getElementById( "result" );

    var request = {};
    request.URL = "http://services.example.com/xmlrpc/";
    request.method = "test.echo";
    request.HTTPmethod = "POST";
    request.parameters = {};
    request.parameters.api_key = "123456ABC";
    request.parameters.message = "Able was I ere I saw Elba.";
    var requestMessage = formatXMLRPC( request );

    xmlhttp = new XMLHttpRequest();
    xmlhttp.open( request.HTTPmethod, request.URL, true);
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            resultDisplay.innerText = xmlhttp.responseText;
        }
    }
    xmlhttp.send( requestMessage );

    requestDisplay.innerText = xmlToString( requestMessage.documentElement );
```

```
}

//Formats a request as XML-RPC document
function formatXMLRPC( request )
{
    var xmldoc = document.implementation.createDocument( "", "", null );
    var root = xmldoc.createElement( "methodCall" );
    xmldoc.appendChild( root );
    var methodName = xmldoc.createElement( "methodName" );
    var methodString = xmldoc.createTextNode( request.method );
    methodName.appendChild( methodString );

    root.appendChild( methodName );

        var params = xmldoc.createElement( "params" );
        root.appendChild( params );

        var param = xmldoc.createElement( "param" );
        params.appendChild( param );
        var value = xmldoc.createElement( "value" );
        param.appendChild( value );
        var struct = xmldoc.createElement( "struct" );
        value.appendChild( struct );

        for( var property in request.parameters )
        {
            var member = xmldoc.createElement( "member" );
            struct.appendChild( member );

            var name = xmldoc.createElement( "name" );
            var paramName = xmldoc.createTextNode( property );
            name.appendChild( paramName )
            member.appendChild( name );

            var value = xmldoc.createElement( "value" );
            var type = xmldoc.createElement( "string" );
            value.appendChild( type );
            var paramValue = xmldoc.createTextNode( request.parameters[property] );
            type.appendChild( paramValue )
            member.appendChild( value );
        }
    return xmldoc;
}

//Returns a string representation of an XML node
function xmlToString( rootNode, indent )
{
    if( indent == null ) indent = "";
    var result = indent + "<" + rootNode.tagName + ">\n";
    for( var i = 0; i < rootNode.childNodes.length; i++)
    {
        if(rootNode.childNodes.item( i ).nodeType == Node.TEXT_NODE )
        {
            result += indent + "    " + rootNode.childNodes.item( i ).textContent + "\n";
        }
    }
    if( rootNode.childElementCount > 0 )
```

```
    {
        result += xmlToString( rootNode.firstElementChild, indent + "    " );
    }
    if( rootNode.nextElementSibling )
    {
        result += indent + "</" + rootNode.tagName + ">\n";
        result += xmlToString( rootNode.nextElementSibling, indent );
    }
    else
    {
        result += indent +"</" + rootNode.tagName + ">\n";
    }
    return result;
}

</script>
</head>
<body onload="makeRequest()">
<h1>Request:</h1>
<pre id="request"></pre>
<h1>Result:</h1>
<pre id="result"></pre>
</body>
</html>
```

SOAP web hizmeti talepleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SOAP genel XML-RPC web hizmeti konsepti üzerine kurulur ve daha karmaşık olmasına rağmen daha zengin bir yazılı veri aktarma yolu sunar. SOAP web hizmetleri tipik olarak web hizmeti çağrılarını, veri tiplerini ve servis URL'sini belirten bir Web Hizmeti Açıklama Dili dosyası (WSDL) sağlar. ActionScript 3, SOAP için doğrudan destek vermese de, bir SOAP XML mesajını "elle" hazırlayabilir, sunucuya gönderebilir ve ardından sonuçları ayrıştırabilirsiniz. Ancak, en basit SOAP web hizmeti haricindeki her şey için, mevcut bir SOAP kütüphanesi kullanarak önemli miktarda geliştirme zamanı kazanmanız muhtemeldir.

Flex çerçevesi SOAP web hizmetine erişim için kütüphanelere sahiptir. Flash Builder'da kütüphane, rpc.swc, Flex çerçevesinin bir parçası olduğu için Flex projelerine otomatik olarak dahil edilir. Flash Professional'da, Flex framework.swc ve rpc.swc'yi kütüphane yoluna ekleyebilir, ardından ActionScript'le Flex sınıflarına erişebilirsiniz.

Daha fazla Yardım konusu

Flash Professional'da Flex web hizmeti bileşenini kullanma

Cristophe Coenraets: Real-time Trader Desktop for Android

Başka bir uygulamada URL açma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir web tarayıcısında veya başka bir uygulamada URL açmak için `navigateToURL()` işlevini kullanabilirsiniz. AIR'de çalışan içerik için, `navigateToURL()` işlevi sayfayı varsayılan sistem web tarayıcısında açar.

Bu işlevin `request` parametresi olarak ilettiğiniz URLRequest nesnesi için yalnızca `url` özelliği kullanılır.

`navigateToURL()` işlevinin ilk parametresi olan `navigate` parametresi, bir URLRequest nesnesidir (bkz.“URLRequest sınıfını kullanma” sayfa 820). İkincisi, pencere adını belirtebileceğiniz isteğe bağlı bir `window` parametresidir. Örneğin, aşağıdaki kod www.adobe.com web sayfasını açar:

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

Not: *Çalışma zamanı, `navigateToURL()` işlevini kullanırken, POST yöntemini kullanan URLRequest nesnesinin (`method` özelliği `URLRequestMethod.POST` olarak ayarlanmış olan) GET yöntemini kullandığını kabul eder.*

`navigateToURL()` işlemini kullanırken, URI şemalarına `navigateToURL()` işlevini çağıran kodun güvenlik sanal alanı temelinde izin verilir.

Bazı API'ler, içeriği bir web tarayıcısında başlatmanıza olanak verir. Güvenlik nedenleriyle, AIR’de bu API’leri kullanırken bazı URI şemalarının kullanılmasına izin verilmez. İzin verilmeyen şemaların listesi, API'yi kullanan kodun güvenlik sanal alanına bağlıdır. (Güvenlik sanal alanlarına ilişkin ayrıntılar için bkz. “AIR güvenliği” sayfa 1088.)

Uygulama sanal alanı (Yalnızca AIR)

Herhangi bir URI şeması, AIR uygulaması korumalı alanında çalışan içerik tarafından başlatılmış URL'de kullanılabilir. URI şemasını işlemek için bir uygulama kaydedilmelidir. Aksi takdirde, istek hiçbir işlem gerçekleştirmez. Aşağıdaki şemalar birçok bilgisayar ve cihazda desteklenmektedir:

- `http:`
- `https:`
- `file:`
- `mailto:` — AIR, bu istekleri kayıtlı sistem posta uygulamasına yönlendirir.
- `sms:` — AIR, `sms:` isteklerini varsayılan metin mesajı uygulamasına yönlendirir. URL biçimi uygulamanın çalıştığı sistem kurallarıyla uyumlu olmalıdır. Örneğin, Android'de URI şeması küçük karakterlere sahip olmalıdır.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- `tel:` — AIR, `tel:` isteklerini varsayılan telefon arama uygulamasına yönlendirir. URL biçimi uygulamanın çalıştığı sistem kurallarıyla uyumlu olmalıdır. Örneğin, Android'de URI şeması küçük karakterlere sahip olmalıdır.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- `market:` — AIR, `market:` isteklerini Android cihazlarında genellikle desteklenen Market uygulamasına yönlendirir.

  ```
  navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
  navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
  ```

Uygulamalar, işletim sisteminin izin verdiği durumlarda özel URI şemaları tanımlayabilir ve kaydedebilir. AIR'den uygulama başlatmak için şemayı kullanarak bir URL oluşturabilirsiniz.

Uzak sanal alanlar

Aşağıdaki şemalara izin verilir. Bu şemaları web tarayıcısında kullandığınız gibi kullanın.

- `http:`
- `https:`
- `mailto:` — AIR, bu istekleri kayıtlı sistem posta uygulamasına yönlendirir.

Diğer URI şemalarına izin verilmez.

Dosyayla yerel sanal alan

Aşağıdaki şemalara izin verilir. Bu şemaları web tarayıcısında kullandığınız gibi kullanın.

- `file:`
- `mailto:` — AIR, bu istekleri kayıtlı sistem posta uygulamasına yönlendirir.

Diğer URI şemalarına izin verilmez.

Ağ iletişimiyle yerel sanal alan

Aşağıdaki şemalara izin verilir. Bu şemaları web tarayıcısında kullandığınız gibi kullanın.

- `http:`
- `https:`
- `mailto:` — AIR, bu istekleri kayıtlı sistem posta uygulamasına yönlendirir.

Diğer URI şemalarına izin verilmez.

Yerel güvenilir sanal alan

Aşağıdaki şemalara izin verilir. Bu şemaları web tarayıcısında kullandığınız gibi kullanın.

- `file:`
- `http:`
- https:
- `mailto:` — AIR, bu istekleri kayıtlı sistem posta uygulamasına yönlendirir.

Diğer URI şemalarına izin verilmez.

Bölüm 45: Diğer Flash Player ve AIR örneklerine bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

LocalConnection sınıfı, Adobe® AIR® uygulamalarının yanı sıra tarayıcıda çalışan SWF içeriği arasındaki iletişimi sağlar. LocalConnection sınıfını tarayıcıda çalışan AIR uygulaması ve SWF içeriği arasında iletişim kurmak için de kullanabilirsiniz. LocalConnection sınıfı Flash Player ve AIR örnekleri arasında veri paylaşabilen çok yönlü uygulamalar oluşturmanızı sağlar.

LocalConnection sınıfı hakkında

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

LocalConnection sınıfı, fscommand() yöntemini veya JavaScript uygulamasını kullanmadan diğer SWF dosyalarına talimatlar gönderebilen SWF dosyaları geliştirmenize olanak sağlar. LocalConnection nesneleri yalnızca aynı istemci bilgisayarında çalışan SWF dosyaları arasında iletişim kurabilir ancak bunlar farklı uygulamalarda çalışabilir. Örneğin, yerel bilgileri barındıran projektör ile tarayıcı tabanlı SWF uzaktan bağlanarak, bir tarayıcıda çalışan SWF dosyası ile bir projektörde çalışan SWF dosyası bilgi paylaşabilir. (Projektör, bağımsız bir uygulama olarak çalışabilecek formatta kaydedilmiş bir SWF dosyasıdır—başka bir deyişle, projektör çalıştırılabilir dosyanın içinde gömülü olduğundan Flash Player uygulamasının yüklenmesini gerektirmez.)

Farklı ActionScript sürümlerini kullanan SWF'ler arasında iletişim kurmak için LocalConnection nesneleri kullanılabilir:

- ActionScript 3.0 LocalConnection nesneleri, ActionScript 1.0 veya 2.0'da oluşturulmuş LocalConnection nesneleriyle iletişim kurabilir.
- ActionScript 1.0 veya 2.0 LocalConnection nesneleri, ActionScript 3.0'da oluşturulmuş LocalConnection nesneleriyle iletişim kurabilir.

Flash Player, farklı sürümlerdeki LocalConnection nesneleri arasındaki bu iletişim otomatik olarak işler.

LocalConnection nesnesini kullanmanın en basit yolu, yalnızca aynı etki alanında veya aynı AIR uygulamasında bulunan LocalConnection nesneleri arasındaki iletişime izin vermektir. Böylece, güvenlik meseleleriyle ilgili endişe duymanıza gerek kalmaz. Ancak, etki alanları arasındaki iletişime izin vermeniz gerekiyorsa, güvenlik önlemleri uygulamanın çeşitli yolları vardır. Daha fazla bilgi için, `http://help.adobe.com/tr_TR/FlashPlatform/reference/actionscript/3/flash/net/LocalConnection.htmlAdobe Flash Platformu için ActionScript 3.0 Ba vurusu`'nda listelenen `LocalConnection` sınıfındaki `allowDomain()` ile domain girişleri ve yönteminin connectionName parametresi hakkındaki tartışmaya bakın.

Tek bir SWF dosyası içinde veri alıp vermek için LocalConnection nesneleri kullanılabilir ancak Adobe bunu önermez. Bunun yerine, paylaşılan nesneleri kullanın.

LocalConnection nesnelerinize geri çağrı yöntemleri eklemenin üç yolu vardır:

- LocalConnection sınıfını alt sınıfa ayırın ve yöntemler ekleyin.
- `LocalConnection.client` özelliğini, yöntemleri uygulayan bir nesneye ayarlayın.

- LocalConnection öğesini genişleten ve dinamik olarak yöntemler ekleyen bir dinamik sınıf oluşturun.

Geri çağrı yöntemleri eklemenin birinci yolu, LocalConnection sınıfının genişletilmesidir. Yöntemleri LocalConnection örneğine dinamik olarak eklemek yerine, özel sınıf içinde tanımlarsınız. Aşağıdaki kodda bu yaklaşık gösterilmektedir:

```
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);
        }
    }
}
```

CustomLocalConnection sınıfının yeni bir örneğini oluşturmak için, şu kodu kullanabilirsiniz:

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

Geri çağrı yöntemleri eklemenin ikinci yolu, `LocalConnection.client` özelliğinin kullanılmasıdır. Bu işlem kapsamında, aşağıdaki kodun gösterdiği gibi, özel bir sınıf oluşturulur ve `client` özelliğine yeni bir örnek atanır:

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

`LocalConnection.client` özelliği, çağrılması gereken nesne geri çağrı yöntemlerini belirtir. Önceki kodda `client` özelliği, özel sınıfın yeni bir örneğine (CustomClient) ayarlanmıştı. `client` özelliğinin varsayılan değeri, geçerli LocalConnection örneğidir. Aynı yöntem kümesine sahip olan ancak farklı şekilde hareket eden iki veri işleyiciniz varsa (örneğin, bir penceredeki düğmenin ikinci bir penceredeki görünümü değiştirdiği bir uygulamada), `client` özelliğini kullanabilirsiniz.

CustomClient sınıfını oluşturmak için şu kodu kullanabilirsiniz:

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

Geri çağrı yöntemleri eklemenin üçüncü yolu olarak dinamik bir sınıf oluşturulması ve yöntemlerin dinamik olarak eklenmesi, aşağıdaki kodun gösterdiği gibi, önceki ActionScript sürümlerinde LocalConnection sınıfının kullanılmasına çok benzer:

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

Geri çağrı yöntemleri, aşağıdaki kod kullanılarak bu sınıfa dinamik olarak eklenebilir:

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

Kod çok taşınabilir bir öğe olmadığından, geri çağrı yöntemleri eklemenin bir önceki yolu önerilmez. Ayrıca, dinamik özelliklere erişim, mühürlenmiş özelliklere erişimden çok daha yavaş olduğundan, bu yerel bağlantı oluşturma yönteminin kullanılması performansla ilgili sorunlara yol açabilir.

isPerUser özelliği

`isPerUser` özelliği Flash Player (10.0.32) ve AIR (1.5.2) uygulamalarına, bir Mac bilgisayarda birden fazla kullanıcı oturum açtığında ortaya çıkan çakışmayı çözmek için eklenmiştir. Diğer işletim sistemlerinde, yerel bağlantı her zaman ayrı kullanıcılarla sınırlı olduğu için bu özellik yok sayılır. `isPerUser` özelliği yeni kodda `true` olarak ayarlanmalıdır. Ancak, varsayılan değer geriye dönük uyumluluk için geçerli olarak `false` şeklindedir. Varsayılan değer, çalışma zamanlarının gelecekteki sürümlerinde değiştirilebilir.

İki uygulama arasında mesaj gönderme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Farklı AIR uygulamaları arasında ve bir tarayıcıda çalıştırılan farklı Adobe® Flash® Player (SWF) arasındaki iletişimi sağlamak için LocalConnection sınıfını kullanırsınız. LocalConnection sınıfını tarayıcıda çalışan AIR uygulaması ve SWF içeriği arasında iletişim kurmak için de kullanabilirsiniz.

Örneğin, bir web sayfasında birden çok Flash Player örneğiniz olabilir veya bir Flash Player örneğinin, bir açılır penceredeki Flash Player örneğinden veriler almasını sağlayabilirsiniz.

Aşağıdaki kod, sunucu olarak görev yapan ve diğer uygulamalardan gelen LocalConnection çağrılarını kabul eden bir LocalConnection nesnesi tanımlar:

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

Bu kod, önce `lc` adlı bir LocalConnection nesnesi oluşturur ve `client` özelliğini `clientObject` nesnesine ayarlar. Başka bir uygulama bu LocalConnection örneğinde bir yöntem çağırdığında, çalışma zamanı `clientObject` nesnesinde bu yöntemi arar.

Belirtilen ada sahip bir bağlantınız zaten varsa, bağlantı girişiminin nesne zaten bağlı olduğu için başarısız olduğunu belirten bir Argument Error istisnası atılır.

Bir Flash Player örneği bu SWF dosyasına her bağlandığında ve belirtilen yerel bağlantı için herhangi bir yöntemi çağırmayı denediğinde, istek, `client` özelliği tarafından belirtilen ve CustomClient1 sınıfı olarak ayarlanan sınıfa gönderilir:

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

LocalConnection sunucusu oluşturmak için, `LocalConnection.connect()` yöntemini çağırın ve benzersiz bir bağlantı adı sağlayın. Belirtilen adla önceden bağlantınız varsa bir ArgumentError hatası oluşturulur ve bu hata, nesne zaten bağlanmış olduğundan bağlantı girişiminin başarısız olduğunu belirtir.

Aşağıdaki parçacık `conn1` adına sahip bir LocalConnection öğesinin nasıl oluşturulacağını gösterir:

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

İkincil uygulamadan birincil uygulamaya bağlanmak için ilk olarak, gönderen LocalConnection nesnesinde bir LocalConnection nesnesi oluşturmanız, daha sonra bağlantı adı ve yürütülecek yöntem adıyla birlikte `LocalConnection.send()` yöntemini çağırmanız gerekir. Örneğin, `doQuit` yöntemini önceden oluşturduğunuz LocalConnection nesnesine göndermek için aşağıdaki kodu kullanın:

```
sendingConnection.send("conn1", "doQuit");
```

Bu kod, var olan LocalConnection nesnesini `conn1` adlı bağlantıyla bağlar ve uzak uygulamadaki `doMessage()` yöntemini çağırır. Uzak uygulamaya parametre göndermek istiyorsanız, aşağıdaki parçacığın gösterdiği şekilde, `send()` yöntemindeki yöntem adından sonra ek argümanlar belirlersiniz:

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

Farklı etki alanlarında ve diğer AIR uygulamalarındaki içeriğe bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yalnızca belirli etki alanlarından iletişime izin vermek için, LocalConnection sınıfının `allowDomain()` veya `allowInsecureDomain()` yöntemlerini çağırın ve izin verilecek bir veya daha çok etki alanı adını onaylayarak, bu LocalConnection nesnesine erişme izni olan bir veya daha çok etki alanının listesini iletin.

Önceki ActionScript sürümlerinde `LocalConnection.allowDomain()` ve `LocalConnection.allowInsecureDomain()` öğeleri, geliştiriciler tarafından uygulanması ve bir Boolean değeri döndürmesi gereken geri çağrı yöntemleriydi. ActionScript 3.0'da `LocalConnection.allowDomain()` ve `LocalConnection.allowInsecureDomain()` öğesinin ikisi de, geliştiricilerin `Security.allowDomain()` ve `Security.allowInsecureDomain()` gibi çağırarak, izin verilecek bir veya daha fazla etki alanını iletebildiği yerleşik yöntemlerdir.

Flash Player 8, yerel SWF dosyaları üzerinde güvenlik kısıtlamaları getirmiştir. İnternet'e erişmesine izin verilen bir SWF dosyası aynı zamanda yerel dosya sistemine erişime sahip olamaz. `localhost` öğesini belirtirseniz, herhangi bir yerel SWF dosyası SWF dosyasına erişebilir. `LocalConnection.send()` yöntemi, çağıran kodun erişiminin olmadığı bir güvenlik sanal alanındaki SWF dosyasıyla iletişim kurmayı denerse, bir `securityError` olayı(`SecurityErrorEvent.SECURITY_ERROR`) gönderilir. Bu hatayı gidermek için, alıcının `LocalConnection.allowDomain()` yönteminde çağıranın etki alanını belirtebilirsiniz.

`LocalConnection.allowDomain()` ve `LocalConnection.allowInsecureDomain()` yönemlerine iletebileceğiniz iki özel değer vardır: `*` ve `localhost`. Yıldız değeri `(*)` tüm etki alanlarından erişime izin verir. `localhost` dizesi, yerel olarak yüklenen, ancak uygulama kaynak dizininin dışında bulunan içerikten uygulamaya yapılan çağrılara olanak verir.

`LocalConnection.send()` yöntemi, kodun erişime sahip olmadığı bir güvenlik sanal alanından bir uygulamayla iletişim kurma girişiminde bulunursa, `securityError` olayı(`SecurityErrorEvent.SECURITY_ERROR`) gönderilir. Bu hatayı gidermek için, alıcının `LocalConnection.allowDomain()` yönteminde çağıranın etki alanını belirtebilirsiniz.

Yalnızca aynı etki alanında bulunan içerik arasında iletişim uygulamak istiyorsanız, alt çizgiyle (_) başlamayan ve etki alanı adı belirtmeyen (örneğin, `myDomain:connectionName`) bir `connectionName` parametresi belirleyebilirsiniz. `LocalConnection.connect(connectionName)` komutundaki dizenin aynısını kullanın.

Farklı etki alanlarında bulunan içerik arasında iletişim uygularsanız, alt çizgiyle başlayan bir `connectionName` parametresi belirlersiniz. Alt çizginin belirtilmesi, alıcı LocalConnection nesnesine sahip içeriği etki alanları arasında daha iyi taşınabilir duruma getirir. İki olası durum şöyledir:

- `connectionName` için dize alt çizgiyle başlamıyorsa, çalışma zamanı üst etki alanı adını ve iki nokta karakterini içeren bir önek ekler (örneğin, `myDomain:connectionName`). Bu bağlantınızın diğer etki alanlarından aynı ada sahip bağlantılarla çakışmamasını sağlasa da, her gönderen LocalConnection nesnesi bu üst etki alanını belirtmelidir (örneğin, `myDomain:connectionName`). Alıcı LocalConnection nesnesini içeren HTML veya SWF dosyasını başka bir etki alanına taşıdığınızda, çalışma zamanı yeni üst etki alanını yansıtmak için öneki değiştirir (örneğin, `anotherDomain:connectionName`). Tüm gönderen LocalConnection nesneleri yeni üst etki alanını gösterecek biçimde elle düzenlenmelidir.
- `connectionName` için dize alt çizgiyle başlıyorsa (örneğin, `_connectionName`), çalışma zamanı dizeye önek eklemez. Bu, alıcı ve gönderen LocalConnection nesnelerinin `connectionName` için aynı dizeyi kullandığı anlamına gelir. Alıcı nesne tüm etki alanlarından bağlantıların kabul edileceğini belirtmek için `LocalConnection.allowDomain()` öğesini kullanıyorsa, alıcı LocalConnection nesnesini içeren HTML veya SWF dosyasını herhangi bir gönderen LocalConnection nesnesini değiştirmeden başka bir etki alanına taşıyabilirsiniz.

  `connectionName` kısmında alt çizgili adlar kullanmanın dezavantajı, örneğin iki uygulama aynı `connectionName` öğesini kullanarak bağlanmaya çalıştığında olduğu gibi çatışmaların ortaya çıkması olasılığıdır. Bununla ilgili ikinci dezavantaj da güvenlikle ilgilidir. Alt çizgi sözdizimini kullanan bağlantı adları, listeleme uygulamasının etki alanını tanımlamaz. Bu nedenlerle, etki alanıyla nitelenen adlar tercih edilir.

Adobe AIR

AIR uygulamasının güvenlik sanal alanında çalışan içerikle (AIR uygulamasıyla yüklenen içerik) bağlantı kurmak için bağlantı adının önüne AIR uygulamasını tanımlayan bir üst etki alanı adı eklemelisiniz. Üst etki alanı adı `app#` ile başlar ve uygulama adı, bir nokta (.) karakteri ve yayıncı kimliğiyle (tanımlandıysa) devam eder. Örneğin, uygulama kimliğine sahip bir `connectionName` parametresiyle kullanılacak uygun üst etki alanı adı `com.example.air.MyApp` şeklindeyken, yayıncı kimliği olmayan parametreyle kullanılacak olan ad şu şekildedir: `"app#com.example.air.MyApp"` Bu nedenle, temel bağlantı adı “appConnection” ise, `connectionName` parametresinde kullanılacak tüm dize şu şekildedir: `"app#com.example.air.MyApp:appConnection"` Uygulamanın yayıncı kimliği varsa, üst etki alanı adı dizesine bu kimlik de dahil edilmelidir: `"app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1"`.

Başka bir AIR uygulamasının yerel bağlantı üzerinden uygulamanızla iletişim kurmasına izin verdiğinizde, yerel bağlantı etki alanı adında iletilen LocalConnection nesnesinin `allowDomain()` yöntemini çağırmalısınız. AIR uygulaması için, bu etki alanı adı, bağlantı dizesinde olduğu gibi uygulama ve yayıncı kimliklerinden oluşturulur. Örneğin, gönderen AIR uygulaması `com.example.air.FriendlyApp` uygulama kimliği ve `214649436BD677B62C33D02233043EA236D13934.1` yayıncı kimliğine sahipse, bu uygulamanın bağlanmasına izin vermek için kullanacağınız etki alanı dizesi şudur:
`app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1`. (AIR 1.5.3'ten itibaren, tüm AIR uygulamaları yayıncı kimliğine sahip değildir.)

Bölüm 46: AIR uygulamasında yerel işlemlere bağlanma

Adobe AIR 2 ve üstü

Adobe AIR 2 sürümünden itibaren, AIR uygulamaları komut çizgisi yolu ile diğer yerel işlemleri çalıştırabilir ve onlarla iletişim kurabilir. Örneğin, bir AIR uygulaması bir işlemi çalıştırabilir ve standart girdi ve çıktı akışları yolu ile onunla iletişim kurabilir.

Yerel işlemlerle iletişim kurmak için, yerel yükleyici yoluyla yüklenecek bir AIR uygulaması paketleyin. Yerel yükleyicinin dosya türü, oluşturulduğu işletim sistemine özeldir:

- Mac OS'de bir DMG dosyasıdır.
- Windows'ta bir EXE dosyasıdır.
- Linux'ta bir RPM veya DEB paketidir.

Bu uygulamalar, uzantılı masaüstü profil uygulamaları olarak bilinir. ADT kullanarak `-package` komutunu çağırırken, `-target native` seçeneğini belirterek yerel bir yükleyici dosyası oluşturabilirsiniz.

Daha fazla Yardım konusu

flash.filesystem.File.openWithDefaultApplication()

flash.desktop.NativeProcess

Yerel işlem iletişimlerine genel bakış

Adobe AIR 2 ve üstü

Uzantılı masaüstü profilindeki bir AIR uygulaması, bir dosyayı komut satırı tarafından başlatılmış gibi çalıştırabilir. Yerel işlemin standart akışlarıyla iletişim kurabilir. Standart akış, standart girdi akışını (stdin), çıktı akışını (stdout) ve standart hata akışını (stderr) içerir.

Not: *Uzantılı masaüstü profilindeki uygulamalar, dosyaları ve uygulamaları `File.openWithDefaultApplication()` yöntemini kullanarak da başlatabilir. Ancak, bu yöntemin kullanılması, AIR uygulamasına standart akışa erişim sağlamaz. Daha fazla bilgi için, bkz. "Varsayılan sistem uygulamasıyla dosya açma" sayfa 688.*

Aşağıdaki kod örneği, uygulama dizinindeki test.exe uygulamasının nasıl başlatılacağını gösterir. Uygulama `"hello"` argümanını, komut çizgisi argümanı olarak iletir ve işlemin standart çıktı akışına bir olay dinleyicisi ekler:

```
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);
}
```

Yerel işlem başlatma ve kapatma

Adobe AIR 2 ve üstü

Yerel işlem başlatmak için, aşağıdakileri yapmak üzere bir NativeProcessInfo nesnesi ayarlayın.

- İmleci başlatmak istediğiniz dosyanın üzerine getirin
- Başlatıldığında işleme iletmek için komut çizgisi argümanlarını kaydedin (isteğe bağlı)
- İşlemin çalışma dizinini ayarlayın (isteğe bağlı)

Yerel işlemi başlatmak için, NativeProcessInfo nesnesini, NativeProcess nesnesinin `start()` yönteminin parametresi olarak iletin.

Örneğin, aşağıdaki kod örneği uygulama dizinindeki test.exe uygulamasının nasıl başlatılacağını gösterir. Uygulama `"hello"` argümanını iletir ve kullanıcının belgeler dizinini çalışma dizini olarak ayarlar:

```
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);
```

İşlemi sonlandırmak için, NativeProcess nesnesinin `exit()` yöntemini çağırın.

Bir dosyanın yüklü uygulamanızda çalıştırılabilir olmasını istiyorsanız, uygulamanızı paketlediğinizde dosya sisteminde çalıştırılabilir durumda olduğundan emin olun. (Mac ve Linux'ta gerekirse, çalıştırılabilir bayrağını ayarlamak için chmod kullanabilirsiniz.)

Yerel işlemle iletişim kurma

Adobe AIR 2 ve üstü

AIR uygulaması yerel bir işlem başlattığında, işlemin standart girdi, standart çıktı ve standart hata akışlarıyla iletişim kurabilir.

NativeProcess nesnesinin aşağıdaki özelliklerini kullanarak akışlara veri yazıp okuyabilirsiniz:

- `standardInput` —Standart girdi akış verilerine erişimi içerir.
- `standardOutput`—Standart çıktı akış verilerine erişim sağlar.
- `standardError` —Standart hata akış verilerine erişimi içerir.

Standart girdi akışına yazma

NativeProcess nesnesinin `standartInput` özelliğinin yazma yöntemini kullanarak standart veri akışına veri yazabilirsiniz. AIR uygulaması işleme veri yazarken, NativeProcess nesnesi `standartInputProgress` olaylarını gönderir.

Standart girdi akışına yazarken bir hata oluşursa, NativeProcess nesnesi bir `ioErrorStandardInput` olayı gönderir.

NativeProcess nesnesinin `closeInput()` yöntemini çağırarak girdi akışını kapatabilirsiniz. Girdi akışı kapandığında, NativeProcess nesnesi bir `standartInputClose` olayı gönderir.

```
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();
}
```

Standart çıktı akışından okuma

Bu özelliğin okuma yöntemlerini kullanarak standart çıktı akışından veri okuyabilirsiniz. AIR uygulaması işlemden çıktı akışı verisi alırken, NativeProcess nesnesi `standardOutputData` olaylarını gönderir.

Standart çıktı akışında bir hata oluşursa NativeProcess nesnesi bir `standardOutputError` olayı gönderir.

İşlem, çıktı akışını kapattığı zaman NativeProcess nesnesi `standartOutputClose` olayını gönderir.

Standart girdi akışından veri okurken, veriyi oluşturulduğu gibi okuduğunuzdan emin olun. Diğer bir deyişle, `standardOutputData` olayı için bir olay dinleyici ekleyin. `standardOutputData` olay dinleyicisinde, verileri NativeProcess nesnesinin `standardOutput` özelliğinden okuyun. Tüm verileri okumak için `standardOutputClose` olayını veya `exit` olayını beklemeyin. Yerel işlem veri oluşturduğunda verileri okumazsanız, ara bellek dolabilir veya veriler kaybedilebilir. Dolmuş bir ara bellek, daha fazla veri yazmak istediğinizde yerel işlemin gecikmesine neden olabilir. Ancak, `standardOutputData` olayı için bir olay dinleyici kaydetmezseniz, ara bellek dolmaz ve işlem gecikmez. Bu durumda, verilere erişemezsiniz.

```
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);
}
```

Standart hata akışından okuma

Bu özelliğin okuma yöntemlerini kullanarak standart hata akışından veri okuyabilirsiniz. AIR uygulaması işlemden hata akışı verisi okurken, NativeProcess nesnesi `standardErrorData` olaylarını gönderir.

Standart hata akışını yazarken bir hata oluşursa, NativeProcess nesnesi bir `standardErrorIoError` olayı gönderir.

İşlem, hata akışını kapattığı zaman NativeProcess nesnesi bir `standartErrorClose` olayı gönderir.

Standart hata akışından veri okurken, veriyi oluşturulduğu gibi okuduğunuzdan emin olun. Başka bir deyişle, `standardErrorData` olayı için bir olay dinleyici ekleyin. `standardErrorData` olay dinleyicisinde, verileri NativeProcess nesnesinin `standardError` özelliğinden okuyun. Tüm verileri okumak için `standardErrorClose` olayını veya `exit` olayını beklemeyin. Yerel işlem veri oluşturduğunda verileri okumazsanız, ara bellek dolabilir veya veriler kaybedilebilir. Dolmuş bir ara bellek, daha fazla veri yazmak istediğinizde yerel işlemin gecikmesine neden olabilir. Ancak, `standardErrorData` olayı için bir olay dinleyicisi kaydetmezseniz, ara bellek dolmaz ve işlem gecikmez. Bu durumda, verilere erişemezsiniz.

```
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);
}
```

Yerel işlem iletişimi için güvenlik etmenleri

Adobe AIR 2 ve üstü

Yerel işlem API'si kullanıcının sistemindeki her çalıştırılabilir öğeyi çalıştırabilir. Komut oluştururken ve çalıştırırken çok dikkatli olun. Çalıştırılacak komutun herhangi bir parçası harici bir kaynaktan gelirse, komutun çalıştırılmasının güvenli bir işlem olduğunu dikkatli bir şekilde doğrulayın. Aynı şekilde, AIR uygulamanız çalışan bir işleme iletilen her veriyi doğrulamalıdır.

Ancak, girdiyi doğrulamak zor olabilir. Bu zorlukları önlemek için, belirli API'leri olan yerel bir uygulama (Windows'ta bir EXE dosyası gibi) yazmak en iyi çözümdür. Bu API'ler yalnızca uygulama tarafından tanımlanan komutları işlemelidir. Örneğin uygulama, standart girdi akışı üzerinden yalnızca sınırlı bir talimat kümesini kabul edebilir.

Windows'taki AIR, .bat dosyalarını doğrudan çalıştırmanıza izin vermez. Komut yorumlayıcı uygulaması (cmd.exe) Windows .bat dosyalarını çalıştırır. Bir .bat dosyası çağırdığınızda, bu komut uygulaması komuta iletilen argümanları başlatılacak ek uygulamalar olarak yorumlayabilir. Argüman dizesine kötü niyetli amaçlarla fazladan karakter eklenmesi cmd.exe uygulamasının zararlı veya güvensiz bir uygulama çalıştırmasına neden olabilir. Örneğin, düzgün bir veri doğrulaması olmadan, AIR uygulamanız `myBat.bat myArguments c:/evil.exe` uygulamasını çağırabilir. Komut uygulaması toplu işleme dosyanızı çalıştırmanın yanı sıra evil.exe uygulamasını da başlatabilir.

Bölüm 47: Harici API'yi kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0 harici API'si (flash.external.ExternalInterface) ActionScript ve Adobe Flash Player'ın içinde çalıştığı kap uygulaması arasında doğrudan iletişime olanak tanır. Bir HTML sayfasında SWF belgesi ve JavaScript arasında etkileşim oluşturmak için ExternalInterface API'yi kullanın.

Kapsayıcı bir uygulamayla etkileşim kurmak için harici API'yi kullanabilir, bir HTML sayfasında ActionScript ve JavaScript arasında veri gönderebilirsiniz.

Bazı yaygın harici API görevleri şunlardır:

- Kap uygulaması hakkında bilgi alma
- Bir tarayıcı veya AIR masaüstü uygulaması içinde görüntülenen web sayfasında kod çağırmak için ActionScript'i kullanma
- Bir web sayfasından ActionScript kodunu çağırma
- Bir web sayfasından ActionScript kodu çağırmayı kolaylaştırmak için bir proxy oluşturma

Not: *Harici arabirim tartışması yalnızca bir SWF dosyasındaki ActionScript ile Flash Player'a başvuru veya SWF'nin yüklendiği örneği içeren kap uygulaması arasındaki iletişimi kapsamaktadır. Bir uygulama içinde Flash Player uygulamasının diğer kullanımları, bu belgenin kapsamı dışında kalır. Flash Player, bir tarayıcı eklentisi veya projektör (bağımsız uygulama) olarak kullanılmak üzere tasarlanmıştır. Diğer kullanım senaryoları sınırlı destek içerebilir.*

Harici API'yi AIR'de kullanma

AIR uygulaması harici bir kaba sahip olmadığından, bu harici arabirim genellikle geçerli değildir--veya genellikle gerekmez. AIR uygulamanız bir SWF dosyasını doğrudan yüklediğinde, uygulama kodu doğrudan SWF içerisindeki ActionScript koduyla iletişim kurabilir (güvenlik sanal alan sınırlamalarına tabidir).

Ancak, AIR uygulamanız bir HTMLLoeader nesnesi (veya Flex içerisinde bir HTML bileşeni) içerisindeki bir HTML sayfasını kullanarak SWF dosyası yüklediğinde, HTMLLoader nesnesi harici kap olarak görev yapar. Dolayısıyla, yüklenmiş SWF'deki ActionScript kodu ile HTMLLoeader'a yüklenmiş HTML sayfasındaki JavaScript kodu arasında iletişim kurmak için harici arabirimi kullanabilirsiniz.

Harici API'yi kullanma temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazı durumlarda (örneğin, bir SWF projektörü oluşturmak için Adobe®Flash® Professional kullanırsanız) SWF dosyası kendi başına çalışabilse de, çoğu durumda SWF dosyası başka bir uygulamanın içindeki bir öğe olarak çalıştırılır. Genellikle, SWF içeren kap bir HTML dosyasıdır; nadiren bir masaüstü uygulamasının kullanıcı arabiriminin tümü veya bir kısmı için SWF dosyası kullanılır.

Daha gelişmiş uygulamalarda çalıştıkça, SWF dosyası ile kap uygulaması arasında iletişim kurmanız gerektiğini fark edebilirsiniz. Örneğin, web sayfasının HTML'deki metni veya diğer bilgileri görüntülemesi ve grafik ya da video gibi dinamik görsel içerik görüntülemek için bir SWF dosyası içermesi genel bir durumdur. Bu tür bir durumda, kullanıcılar web sayfasındaki bir düğmeyi tıklattığında SWF dosyasındaki bir şeyi değiştirecek şekilde bunu ayarlamak isteyebilirsiniz. ActionScript, SWF dosyasındaki ActionScript ile kap uygulamasındaki diğer kod arasında bu iletişim türünü kolaylaştıran harici API adında bir mekanizma içerir.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi bu özellikle ilgili önemli terimler içerir:

Kap uygulaması Flash Player'ın içinde çalıştığı uygulama, Flash Player içeriği bulunduran web tarayıcısı ve HTML sayfası veya bir web sayfasında SWF'yi yükleyen AIR uygulaması gibi bir SWF dosyasıdır.

Projektör SWF içeriği ve Flash Player'ın gömülü bir sürümünü içeren bir çalıştırılabilir bir dosya. Flash Professional veya tek başına Flash Player kullanarak bir projektör dosyası oluşturabilirsiniz. Projektörler, CD-ROM ile SWF dosyalarını dağıtmak veya indirme boyutunun sorun oluşturmadığı ve SWF yazarının, kullanıcının bilgisayarında Flash Player yüklü olup olmadığına bakılmaksızın kullanıcının SWF dosyasını çalıştırabileceğinden emin olmak istediği benzer durumlarda yaygın olarak kullanılır.

Proxy Bir uygulamadaki (“harici uygulama”) kodu başka bir uygulama (“çağıran uygulama”) adına çağıran ve çağıran uygulamaya değerler döndüren bir aracı uygulama veya kod. Proxy, arasında şunların da yer aldığı bir çok nedenle kullanılabilir:

- Çağıran uygulamadaki yerel işlev çağrılarını, harici uygulama tarafından anlaşılan biçime dönüştürerek harici işlev çağrıları yapma işlemini basitleştirmek.
- Güvenlik kısıtlamalarına ve çağıranın harici uygulamayla doğrudan iletişim kurmasını önleyen diğer kısıtlamalara yönelik çözüm bulma.

Serileştirme Nesneleri veya veri değerlerini, iki programlama sistemi arasında (örn. Internet üzerinden veya tek bir bilgisayarda çalışan iki farklı uygulama arasında) mesajlardaki değerleri iletmek için kullanılabilir bir biçime dönüştürme.

Örneklerle çalışma

Bu bölümdeki kod örneklerinin çoğu, tam çalışma örnekleri veya değerleri kontrol eden kodlardan çok, gösterme amaçlı küçük kod listeleridir. Harici API kullanılması (tanım gereği), kap uygulamasındaki kodun yanı sıra ActionScript yazılmasını da gerektirdiğinden, örneklerin test edilmesi, kap (örneğin, SWF dosyasını içeren bir web sayfası) oluşturulmasını ve kap ile etkileşim kurmak için kod listelerinin kullanılmasını içerir.

ActionScript'ten JavaScript'e iletişim örneğini test etmek için:

1. Flash Professional'ı kullanarak yeni bir belge oluşturun ve belgeyi bilgisayarınıza kaydedin.
2. Ana menüden, Dosya > Yayınlama Ayarları seçeneklerini belirleyin.
3. Yayınlama Ayarları iletişim kutusunda, Biçimler sekmesinde Flash ve HTML onay kutularının seçilmiş olduğunu doğrulayın.
4. Yayınla düğmesini tıklatın. Böylece aynı klasörde ve belgeyi kaydetmek için kullandığınız aynı adla bir SWF dosyası ve HTML dosyası oluşturulur. Yayınlama Ayarları iletişim kutusunu kapatmak için Tamam'ı tıklatın.
5. HTML onay kutusunun seçimini kaldırın. HTML sayfası oluşturulduğuna göre, uygun JavaScript kodunu eklemek için bunu değiştireceksinizdir. HTML onay kutusunun seçiminin kaldırılması, siz HTML sayfasını değiştirdikten sonra Flash uygulamasının SWF dosyasını yayınlarken yeni bir HTML sayfasıyla değişikliklerinizi geçersiz kılmaması sağlanır.
6. Yayınlama Ayarları iletişim kutusunu kapatmak için Tamam'ı tıklatın.

7 SWF dosyasını yayınladığınızda Flash tarafından oluşturulan HTML dosyasını HTML veya metin düzenleyici uygulamasıyla açın. HTML kaynak koduna açma ve kapatma `script` etiketlerini ekleyin ve bunlara örnek kod listesindeki JavaScript kodunu kopyalayın:

```
<script>
// add the sample JavaScript code here
</script>
```

8 HTML dosyasını kaydedin ve Flash uygulamasına geri dönün.

9 Zaman Çizelgesi'nin Kare 1'inde anahtar kareyi seçin ve Eylemler panelini açın.

10 ActionScript kod listesini Komut Dosyası bölmesine kopyalayın.

11 SWF dosyasını yaptığınız değişikliklerle güncellemek için, ana menüden, Dosya > Yayınla seçeneklerini belirleyin.

12 Sayfayı görüntülemek ve ActionScript ile HTML arasındaki iletişimi test etmek için, bir web tarayıcısı kullanarak, düzenlediğiniz HTML sayfasını açın.

ActionScript'ten ActiveX kabına iletişim örneğini test etmek için:

1 Flash Professional'ı kullanarak yeni bir belge oluşturun ve belgeyi bilgisayarınıza kaydedin. Bunu, kap uygulamanızın SWF dosyasını bulmayı beklediği klasöre kaydetmek isteyebilirsiniz.

2 Ana menüden, Dosya > Yayınlama Ayarları seçeneklerini belirleyin.

3 Yayınlama Ayarları iletişim kutusunda, Biçimler sekmesinde yalnızca Flash onay kutusunun seçilmiş olduğunu doğrulayın.

4 Flash onay kutusunun yanındaki Dosya alanında, SWF dosyanızın yayınlanacağı klasörü seçmek için klasör simgesini tıklatın. SWF dosyanızın konumunu ayarlayarak, (örneğin) belgeyi tek bir klasörde tutabilir ancak yayınlanan SWF dosyasını, kap uygulamasının kaynak kodunu içeren klasör gibi başka bir klasöre koyabilirsiniz.

5 Zaman Çizelgesi'nin Kare 1'inde anahtar kareyi seçin ve Eylemler panelini açın.

6 Örneğin ActionScript kodunu, Komut Dosyası bölmesine kopyalayın.

7 SWF dosyasını yeniden yayınlamak için ana menüden, Dosya > Yayınla seçeneklerini belirleyin.

8 ActionScript ile kap uygulaması arasındaki iletişimi test etmek için kap uygulamanızı oluşturup çalıştırın.

Bir HTML sayfası ile iletişim kurma amacıyla harici API kullanmaya yönelik tam örnekler için aşağıdaki konuya bakın:

- "Harici API örneği: Bir web tarayıcısında ActionScript ve JavaScript arasındaki iletişim" sayfa 858

Bu örnekler, harici API kullanarak kod yazarken kullanmanız gereken, ActionScript ve kap hata denetleme kodu da dahil olmak üzere tam kodu içerir. Harici API kullanmaya ilişkin başka bir tam örnek için, ActionScript 3.0 Başvurusu'ndaki ExternalInterface sınıfının sınıf örneğine bakın.

Harici API gereksinimleri ve avantajları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici API, ActionScript'in, Flash Player için kap olarak hareket eden bir "harici uygulamada" (genellikle bir web tarayıcısı veya bağımsız projektör uygulaması) çalışan kod ile ActionScript arasındaki iletişime yönelik bir mekanizma sağlayan bölümüdür. ActionScript 3.0'da, harici API işlevi ExternalInterface sınıfı tarafından sağlanır. Flash Player 8'den önceki Flash Player sürümlerinde `fscommand()` eylemi, kap uygulamasıyla iletişimi yürütmek için kullanılıyordu. ExternalInterface sınıfı `fscommand()` için bir değiştirmedir.

Not: *Eski `fscommand()` işlevini kullanmanız gerekirse—örneğin, eski uygulamalarda uyumluluğu korumak veya üçüncü taraf SWF kap uygulaması ya da bağımsız Flash Player ile etkileşim kurmak için—bu, flash.system paketinde paket düzeyinde işlev olarak halen kullanılabilir.*

ExternalInterface sınıfı, bir HTML sayfasında ActionScript ve Flash Player'dan JavaScript'e kolaylıkla iletişim kurmanızı sağlayan bir alt sistemdir.

ExternalInterface sınıfı yalnızca şu koşullar altında kullanılabilir:

- Windows (5.0 ve sonrası) için tüm desteklenen Internet Explorer sürümlerinde
- Şu anda Firefox 1.0 ve sonrasını, Mozilla 1.7.5 ve sonrasını, Netscape 8.0 ve sonrasını ve Safari 1.3 ve sonrasını içeren NPRuntime arabirimini destekleyen herhangi bir tarayıcıda.
- SWF, HTMLLoader denetimi tarafından görüntülenen bir HTML sayfasında gömülü olduğunda bir AIR uygulamasında.

Diğer tüm durumlarda (örneğin bağımsız bir oynatıcı çalıştırılırken), `ExternalInterface.available` özelliği `false` değerini döndürür.

ActionScript'ten, HTML sayfasında bir JavaScript işlevini çağırabilirsiniz. Harici API, `fscommand()` öğesine göre daha gelişmiş işlevler sunar:

- Yalnızca `fscommand()` işleviyle kullanabildiğiniz işlevleri değil, JavaScript işlevini de kullanabilirsiniz.
- Herhangi bir ada sahip herhangi bir sayıda argüman iletebilirsiniz; bir komut ve tek bir dize argümanı iletmekle sınırlı değilsiniz Bu da API'nin `fscommand()` öğesinden daha fazla esnek olmasını sağlar.
- Çeşitli veri türlerini (örn. Boolean, Number ve String) iletebilirsiniz; String parametreleriyle sınırlı değilsiniz.
- Bir çağrının değerini alabilirsiniz ve bu değer hemen ActionScript'e döndürülür (yaptığınız çağrının döndürme değeri olarak).

Önemli: *Bir HTML sayfasında Flash Player örneğine verilen ad (`object` etiketinin `id` niteliği) kısa çizgi (`-`) veya JavaScript'te operatör olarak tanımlanan diğer karakterleri (örn. `+`, `*`, `/`, `\`, `.`, v.b.) ActionScript tarafından yapılan ExternalInterface çağrıları kap web sitesi Internet Explorer'da görüntülendiğinde çalışmaz. Ek olarak, Flash Player örneğini tanımlayan HTML etiketleri (`object` ve `embed` etiketleri) bir HTML `form` etiketinde yuvalanırlarsa, ActionScript tarafından yapılan ExternalInterface çağrıları çalışmaz.*

ExternalInterface sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript ile kap uygulaması arasındaki iletişim iki formdan birini alabilir: ActionScript, kapta tanımlı kodu (örn. bir JavaScript işlevi) çağırabilir veya kaptaki kod, çağrılabilir olarak belirlenen bir ActionScript işlevini çağırabilir. Her iki durumda da, çağrılmakta olan koda bilgi gönderilebilir ve çağrıyı yapan koda sonuçlar döndürülebilir.

Bu iletişimi kolaylaştırmak için, ExternalInterface sınıfı iki statik özellik ve iki statik yöntem içerir. Bu özellikler ve yöntemler, harici arabirim bağlantısı hakkında bilgi edinmek, ActionScript'ten kaptaki kodu çalıştırmak ve ActionScript işlevlerini kap tarafından çağrılabilir duruma getirmek için kullanılır.

Harici kap hakkında bilgi alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`ExternalInterface.available` özelliği, geçerli Flash Player uygulamasının harici arabirim sunan bir kapta bulunup bulunmadığını belirtir. Harici arabirim kullanılabilir durumdaysa, bu özellik `true` olur; aksi takdirde, `false` olur. ExternalInterface sınıfındaki diğer işlevlerden herhangi birini kullanmadan önce, her zaman şu şekilde geçerli kabın harici arabirim iletişimini destekleyip desteklemediğini kontrol etmeniz gerekir:

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

Not: *`ExternalInterface.available` özelliği, geçerli kabın ExternalInterface bağlantısını destekleyen bir türde olup olmadığını bildirir. Bu, size geçerli tarayıcıda JavaScript'in etkinleştirilmiş olup olmadığını bildirmez.*

`ExternalInterface.objectID` özelliği, Flash Player örneğinin benzersiz tanımlayıcısını (özellikle de Internet Explorer'da `object` etiketinin `id` niteliği veya NPRuntime arabirimini kullanan tarayıcılarda `embed` etiketinin `name` niteliği) belirlemenize olanak sağlar. Bu benzersiz kimlik, tarayıcıda geçerli SWF belgesini temsil eder ve örneğin, bir kap HTML sayfasında JavaScript işlevi çağrılırken SWF belgesine başvuru yapmak için kullanılabilir. Flash Player kabı bir web tarayıcısı değilse, bu özellik `null` olur.

ActionScript'ten harici kod çağırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`ExternalInterface.call()` yöntemi, kap uygulamasında kodu çalıştırır. Kap uygulamasında çağrılması için en az bir parametre ve işlevin adını içeren bir dize gereklidir. `ExternalInterface.call()` yöntemine iletilen tüm ek parametreler, işlev çağrısının parametreleri olarak kap boyunca iletilir.

```
// 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);
```

Kap bir HTML sayfasıysa, bu yöntem belirtilen ada sahip ve kapsayan HTML sayfasındaki bir `script` öğesinde tanımlanmış olması gereken JavaScript işlevini çağırır. JavaScript işlevinin döndürme değeri, ActionScript'e geri iletilir.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

Kap başka bir ActiveX kabıysa, bu yöntem Flash Player ActiveX denetiminin `FlashCall` olayını göndermesine neden olur. Belirtilen işlev adı ve tüm parametreler, Flash Player tarafından bir XML dizesine serileştirilir. Kap, olay nesnesinin `request` özelliğinde bu bilgilere erişebilir ve kendi kodunun nasıl çalıştırılacağını belirlemek için bu bilgileri kullanır. Kap kodu, ActionScript'e bir değer döndürmek için ActiveX nesnesinin `SetReturnValue()` yöntemini çağırarak sonucu (XML dizesine serileştirişmiş) o yöntemin bir parametresi olarak iletir. Bu iletişim için kullanılan XML biçimi hakkında daha fazla bilgi almak için bkz. "Harici API'nin XML biçimi" sayfa 857.

Kap bir web tarayıcısı da olsa veya başka bir ActiveX kabı da olsa, çağrı başarısız olursa ya da kap yöntemi bir döndürme değeri belirtmezse, `null` değeri döndürülür. Kapsayan ortam, çağıran kodun erişemediği bir güvenlik sanal alanına aitse, `ExternalInterface.call()` yöntemi bir SecurityError istisnası atar. Kapsayan ortamda `allowScriptAccess` için uygun bir değer ayarlayarak bu soruna geçici çözüm oluşturabilirsiniz. Örneğin, bir HTML sayfasında `allowScriptAccess` değerini değiştirmek için, `object` ve `embed` etiketlerinde uygun niteliği düzenlersiniz.

Kaptan ActionScript kodu çağırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kap yalnızca bir işlevde bulunan ActionScript kodunu çağırabilir—başka bir ActionScript kodu kap tarafından çağrılamaz. Kap uygulamasından bir ActionScript işlevi çağırmak için iki şey yapmanız gerekir: işlevi ExternalInterface sınıfıyla kaydetme ve sonra kabın kodundan çağırma.

İlk olarak, kap için kullanılabilir durumda olduğunu belirtmek üzere ActionScript işlevinizi kaydetmeniz gerekir. Aşağıdaki gibi, `ExternalInterface.addCallback()` yöntemini kullanın:

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

`addCallback()` yöntemi iki parametre alır. Birincisi olan String olarak işlev adı, işlevin kap tarafından bilinecek adıdır. İkinci parametre, kap tanımlı işlev adını çağırdığında çalıştırılacak olan gerçek ActionScript işlevidir. Bu adlar birbirinden farklı olduğundan, ActionScript işlevi farklı bir ada sahip olsa da, kap tarafından kullanılacak bir işlev adı belirtebilirsiniz. Bu özellikle de işlev adı bilinmediğinde, örneğin, adsız bir işlev belirtildiğinde veya çağrılacak işlev çalışma zamanında belirlendiğinde kullanışlıdır.

Bir ActionScript işlevi ExternalInterface sınıfıyla kaydolduktan sonra, kap gerçekten işlevi çağırabilir. Bunun yapılma şekli, kabın türüne göre değişiklik gösterir. Örneğin, ActionScript işlevi bir web tarayıcısındaki JavaScript kodunda, Flash Player tarayıcı nesnesinin bir yöntemiymiş gibi (başka bir deyişle, `object` veya `embed` etiketini temsil eden bir JavaScript yöntemiymiş gibi), kayıtlı işlev adı kullanılarak çağrılır. Başka bir deyişle, parametreler iletilir ve yerel bir işlev çağrılmış gibi bir sonuç döndürülür.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

Alternatif olarak, masaüstü uygulamasında çalışan bir SWF dosyasında ActionScript işlevi çağrılırken, kayıtlı işlev adı ve parametrelerin XML biçiminde bir dizeye serileştirilmesi gerekir. Daha sonra parametre olarak XML dizesiyle ActiveX denetiminin `CallFunction()` yöntemi çağrılarak gerçekten çağrı gerçekleştirilir. Bu iletişim için kullanılan XML biçimi hakkında daha fazla bilgi almak için bkz. "Harici API'nin XML biçimi" sayfa 857.

Çağıran bir tarayıcıdaki JavaScript kodu olduğunda doğrudan bir değer olarak veya çağıran bir ActiveX kabı olduğunda XML biçimli bir dize olarak serileştirilmiş şekilde, her durumda ActionScript işlevinin döndürme değeri kap koduna geri iletilir.

Harici API'nin XML biçimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript ile Shockwave Flash ActiveX denetimini barındıran bir uygulama arasındaki iletişim, işlev çağrılarını ve değerlerini kodlamak için belirli bir XML biçimini kullanır. Harici API tarafından kullanılan XML biçiminin iki bölümü vardır. Bir biçim, işlev çağrılarını temsil etmek için kullanılır. Başka bir biçim ise tek tek değerleri temsil etmek için kullanılır; bu biçim, işlev döndürme değerlerinin yanı sıra işlevlerdeki parametreler için de kullanılır. İşlev çağrılarının XML biçimi, ActionScript'e ve ActionScript'ten gelen çağrılar için kullanılır. ActionScript'ten gelen bir işlev için Flash Player, XML'i kaba iletir; kaptan gelen bir çağrı için Flash Player, kap uygulamasının kendisine bu biçimde bir XML dizesi iletmesini bekler. Aşağıdaki XML parçası, örnek bir XML biçimli işlev çağrısı göstermektedir:

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

Kök düğüm, `invoke` düğümüdür. İki niteliğe sahiptir: `name` öğesi, çağrılacak işlevin adını belirtir ve `returntype` her zaman `xml` olur. İşlev çağrısı parametreler içeriyorsa, `invoke` düğümü, alt düğümleri daha sonra açıklanan tek tek değer biçimi kullanılarak biçimlendirilmiş parametre değerleri olacak bir alt `arguments` düğümüne sahiptir.

İşlev parametreleri ve işlev döndürme değerleri gibi tek tek değerler, gerçek değerlere ek olarak veri türü bilgilerini de içeren bir biçimlendirme şemasını kullanır. Aşağıdaki tabloda, bu veri türünün değerlerini kodlamak için kullanılan ActionScript sınıfları ve XML biçimi listelenmektedir:

ActionScript sınıfı/değeri	**C# sınıfı/değeri**	**Biçim**	**Yorumlar**
null	null	<null/>	
Boolean `true`	bool true	<true/>	
Boolean `false`	bool false	<false/>	
Dize	dize	<string>string value</string>	
Number, int, uint	single, double, int, uint	`<number>27.5</number>` `<number>-12</number>`	

ActionScript sınıfı/değeri	C# sınıfı/değeri	Biçim	Yorumlar

Array (karma tür olabilen öğeler)	ArrayList veya object[] gibi karma tür öğelere izin veren bir koleksiyon	`<array>`
		`<property id="0">`
		`<number>27.5</number>`
		`</property>`
		`<property id="1">`
		`<string>Hello there!</string>`
		`</property>`
		`...`
		`</array>`
			`property` düğümü tek tek öğeleri tanımlar ve `id` niteliği sayısal, sıfır tabanlı dizindir.

Nesne	Dize anahtarları içeren bir HashTable gibi dize anahtarları ve nesne değerlerini içeren bir sözlük	`<object>`
		`<property id="name">`
		`<string>John Doe</string>`
		`</property>`
		`<property id="age">`
		`<string>33</string>`
		`</property>`
		`...`
		`</object>`
			`property` düğümü tek tek özellikleri tanımlar ve `id` niteliği özellik adıdır (dize).

Diğer yerleşik veya özel sınıflar		`<null/> or` `<object></object>`	ActionScript, diğer nesneleri null olarak veya boş bir nesne olarak kodlar. Her iki durumda da özellik değerleri kaybolur.

Not: *Bu tabloda, örnekleme yoluyla, ActionScript sınıflarına ek olarak eşdeğer C# sınıfları gösterilmektedir, harici API, herhangi bir programlama diliyle veya ActiveX denetimlerini destekleyen çalışma zamanıyla iletişim kurmak için kullanılabilir ve C# uygulamalarıyla sınırlı değildir.*

Bir ActiveX kabı uygulaması ile harici API kullanarak kendi uygulamalarınızı oluştururken, yerel işlev çağrılarını serileştirilmiş XML biçimine dönüştürme görevini gerçekleştirecek bir proxy yazmanız rahatlık sağlayabilir. C# uygulamasında yazılmış bir proxy sınıfı örneği için, bkz. ExternalInterfaceProxy sınıfının içinde.

Harici API örneği: Bir web tarayıcısında ActionScript ve JavaScript arasındaki iletişim

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu örnek uygulama, kişinin kendisiyle sohbet etmesine olanak sağlayan bir Anında Mesajlaşma uygulaması bağlamında, bir web tarayıcısında ActionScript ile JavaScript arasında iletişim kurulmasına yönelik uygun teknikleri gösterir. Mesajlar, web sayfasındaki bir HTML formu ile harici API kullanan bir SWF arabirimi arasında gönderilir. Bu örnekle gösterilen teknikler arasında şunlar yer alır:

- İletişim kurulmadan önce tarayıcının iletişim kurmak için hazır olduğunu doğrulayarak düzgün şekilde iletişimi başlatma
- Kabın harici API'yi destekleyip desteklemediğini kontrol etme
- ActionScript'ten JavaScript işlevlerini çağırarak parametreler iletme ve buna yanıt olarak değerler alma
- ActionScript yöntemlerini JavaScript tarafından çağrılabilir duruma getirme ve bu çağrıları gerçekleştirme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Introvert IM uygulama dosyaları Samples/IntrovertIM_HTML klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
IntrovertIMApp.fla veya IntrovertIMApp.mxml	Flash (FLA) veya Flex (MXML) için ana uygulama dosyası.
com/example/programmingas3/introvertIM/IMManager.as	ActionScript ile kap arasında iletişim kuran ve bu iletişimi yöneten sınıf
com/example/programmingas3/introvertIM/IMMessageEvent.as	Kaptan bir mesaj alındığında IMManager sınıfı tarafından gönderilen özel olay türü.
com/example/programmingas3/introvertIM/IMStatus.as	Değerleri, uygulamada seçilebilen farklı "kullanılabilirlik" durum değerlerini temsil eden numaralandırma.
html-flash/IntrovertIMApp.html veya html-template/index.template.html	Flash uygulaması için HTML sayfası (html-flash/IntrovertIMApp.html) veya Adobe Flex uygulaması için kap HTML sayfası oluşturmak üzere kullanılan şablon (html-template/index.template.html). Bu dosya, uygulamanın kap parçasını oluşturan tüm JavaScript işlevlerini içerir.

ActionScript tarayıcı iletişimi için hazırlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Harici API'nin en yaygın kullanımlarından biri, ActionScript uygulamalarının bir web tarayıcısıyla iletişim kurmasına olanak sağlamaktır. ActionScript yöntemleri harici API'yi kullanarak JavaScript'te yazılmış bir kod çağırabilir ve aksi durumda tam tersi olur. Tarayıcıların ve dahili olarak sayfa oluşturma şekillerinin karmaşık olması nedeniyle, bir SWF belgesinin HTML sayfasındaki birinci JavaScript çalıştırılmadan önce kendi geri çağrılarını kaydedeceğini garanti etmenin bir yolu yoktur. Bu nedenle, JavaScript'ten SWF belgesindeki işlevler çağrılmadan önce, SWF belgenizin, bağlantıları kabul etmeye hazır olduğunu HTML sayfasına bildirmek için her zaman HTML sayfasını çağırması gerekir.

Örneğin, IMManager sınıfı tarafından gerçekleştirilen adımlar dizisiyle, Introvert IM öğesi tarayıcının iletişim için hazır olup olmadığını belirler ve SWF dosyasını iletişim için hazırlar. Birinci adım (tarayıcının iletişim için ne zaman hazır olduğunu belirleme), şu şekilde IMManager yapıcısında gerçekleşir:

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

İlk olarak, kod `ExternalInterface.available` özelliğini kullanarak geçerli kapta harici API'nin kullanılabilir olup olmadığını kontrol eder. Harici API kullanılabilir durumdaysa, kod, iletişim kurma işlemini başlatır. Harici bir uygulamayla iletişim kurmayı denediğinizde güvenlik istisnaları ve başka hatalar oluşabildiğinden, kod bir `try` bloğunda sarılır. (Kısa olması için, karşılık gelen `catch` blokları çıkarılır.)

Daha sonra kod, burada listelenen `isContainerReady()` yöntemini çağırır:

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

Bunun sonucunda `isContainerReady()` yöntemi, şu şekilde, `isReady()` JavaScript işlevini çağırmak için `ExternalInterface.call()` yöntemini kullanır:

```
<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>
```

`isReady()` işlevi yalnızca `jsReady` değişkeninin değerini döndürür. Bu değişken başlangıçta `false` değerindedir; web sayfasının `onload` olayı tetiklendikten sonra, değişkenin değeri `true` olarak değiştirilir. Başka bir deyişle, sayfa yüklenmeden önce ActionScript `isReady()` işlevini çağırırsa, JavaScript `ExternalInterface.call("isReady")` için `false` değerini döndürür ve sonra da ActionScript `isContainerReady()` yöntemi `false` değerini döndürür. Sayfa yüklendikten sonra, JavaScript `isReady()` işlevi `true` değerini döndürür, böylece ActionScript `isContainerReady()` yöntemi de `true` değerini döndürür.

IMManager yapıcısına geri döndüğümüzde, kabın hazır olma durumuna bağlı olarak iki şeyden biri gerçekleşir. `isContainerReady()` öğesi `true` değerini döndürürse, kod yalnızca `setupCallbacks()` yöntemini çağırır ve bu da JavaScript ile iletişim kurma işlemini tamamlar. Diğer bir yandan, `isContainerReady()` öğesi `false` değerini döndürürse, işlem beklemeye alınır. Aşağıdaki gibi, bir Timer nesnesi oluşturulur ve bu nesneye her 100 milisaniyede bir `timerHandler()` yöntemini çağırması bildirilir:

```
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();
    }
}
```

`timerHandler()` yöntemi her çağrıldığında, bir kez daha `isContainerReady()` yönteminin sonucunu kontrol eder. Kap başlatıldıktan sonra, bu yöntem `true de erini döndürür.` Daha sonra kod, Timer öğesini durdurur ve tarayıcıyla iletişim kurma işlemini bitirmek için `setupCallbacks()` yöntemini çağırır.

ActionScript yöntemlerini JavaScript kullanımına sunma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Önceki örnekte gösterildiği gibi, kod tarayıcının hazır olduğunu belirledikten sonra `setupCallbacks()` yöntemi çağrılır. Bu yöntem, burada gösterildiği gibi, JavaScript'ten gelen çağrıları alacak şekilde ActionScript'i hazırlar:

```
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");
}
```

`setCallBacks()` yöntemi, JavaScript'ten çağrılabilir durumda olacak iki yöntemi kaydetmek için `ExternalInterface.addCallback()` öğesini çağırarak kapla iletişim kurmaya hazırlanma görevini bitirir. Bu kodda, birinci parametre olan yöntemin JavaScript tarafından bilinecek adı(`"newMessage"` ve `"getStatus"`), yöntemin ActionScript'teki adıyla aynıdır. (Bu durumda, farklı adlar kullanılmasının bir avantajı olmadığından kolaylık sağlamak için aynı ad yeniden kullanılmıştır.) Son olarak, kaba ActionScript işlevlerinin kaydedildiğini bildiren `setSWFIsReady()` JavaScript işlevini çağırmak için `ExternalInterface.call()` yöntemi kullanılır.

ActionScript'ten tarayıcıya iletişim

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Introvert IM uygulaması, kap sayfasındaki JavaScript işlevlerinin çağrılmasına ilişkin çok çeşitli örnekler gösterir. En basit durumda (`setupCallbacks()` yöntemindeki bir örnek), herhangi bir parametre iletilmeden veya karşılığında bir değer alınmadan `setSWFIsReady()` JavaScript işlevi çağrılır:

```
ExternalInterface.call("setSWFIsReady");
```

`isContainerReady()` yönteminin başka bir örneğinde, ActionScript `isReady()` işlevini çağırır ve karşılığında bir Boolean değeri alır:

```
var result:Boolean = ExternalInterface.call("isReady");
```

Ayrıca harici API kullanarak da JavaScript işlevlerine parametreler iletebilirsiniz. Örneğin, kullanıcı "dönüştürme ortağına" yeni bir mesaj gönderdiğinde çağrılan, IMManager sınıfının `sendMessage()` yöntemini göz önünde bulundurun:

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

Yine belirlenen JavaScript işlevini çağırmak için `ExternalInterface.call()` öğesi kullanılarak yeni mesaj tarayıcıya bildirilir. Ayrıca, mesajın kendisi de ek bir parametre olarak `ExternalInterface.call()` öğesine gönderilir ve sonra da parametre olarak `newMessage()` JavaScript işlevine iletilir.

JavaScript'ten ActionScript kodunu çağırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İletişimin iki yönlü olduğu varsayılır ve Introvert IM uygulaması da bir istisna değildir. Flash Player IM istemcisi mesaj göndermek için JavaScript'i çağırmakla kalmaz, HTML formu da mesaj göndermek için JavaScript kodunu çağırır ve SWF dosyasından bilgi ister. Örneğin, SWF dosyası iletişim kurmayı bitirdiğini ve iletişim kurmak için hazır olduğunu kaba bildirdiğinde tarayıcının yapacağı ilk şey, SWF IM istemcisinden ilk kullanıcı kullanılabilirliği durumunu almak için IMManager sınıfının `getStatus()` yöntemini çağırmaktır. Aşağıdaki gibi bu, web sayfasında, `updateStatus()` işlevinde yapılır:

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

Kod, SWF dosyasının ExternalInterface sınıfı ile yöntemlerini kaydettiğini tarayıcıya bildirip bildirmediğini izleyen `swfReady` değişkeninin değerini kontrol eder. SWF dosyası iletişimi almaya hazırsa, sonraki satır (`var currentStatus =` ...) gerçekten IMManager sınıfında `getStatus()` yöntemini çağırır. Bu kod satırında üç şey gerçekleşir:

- `getSWF()` JavaScript işlevi çağrılarak SWF dosyasını temsil eden bir JavaScript'e başvuru döndürülür. `getSWF()` öğesine iletilen parametre, bir HTML sayfasında birden çok SWF dosyası olması durumunda hangi tarayıcı nesnesinin döndürüleceğini belirler. Bu parametreye iletilen değerin, SWF dosyasını dahil etmek için kullanılan `object` etiketinin `id` niteliğiyle ve `embed` etiketinin `name` niteliğiyle eşlemesi gerekir.
- SWF dosyasına başvuru kullanılarak, `getStatus()` yöntemi, SWF nesnesinin bir yöntemiymiş gibi çağrılır. Bu durumda, "`getStatus`" işlev adı, `ExternalInterface.addCallback()` kullanılarak ActionScript işlevinin kaydedildiği ad olduğundan bu işlev adı kullanılır.
- `getStatus()` ActionScript yöntemi bir değer döndürür ve bu değer `currentStatus` değişkenine atanır, daha sonra da bu değer `status` metin alanının içeriği (`value` özelliği) olarak atanır.

Not: *Kodu takip ederseniz, `updateStatus()` işlevinin kaynak kodunda, `getSWF()` işlevini çağıran kod satırının gerçekte şu şekilde yazıldığını fark edersiniz: var currentStatus = getSWF("${application}").getStatus(); `${application}` metni, HTML sayfası şablonunda bir yer tutucudur; Adobe Flex Builder, uygulama için gerçek HTML sayfasını oluşturduğunda, bu yer tutucu metnin yerini, `object` etiketinin `id` niteliği olarak ve `embed` etiketinin `name` niteliği olarak kullanılan metnin aynısı (örnekte `IntrovertIMApp`) alır. `getSWF()` işlevi tarafından beklenen değer budur.*

`sendMessage()` JavaScript işlevi, ActionScript işlevine bir parametre iletilmesini gösterir. (`sendMessage()`, kullanıcı HTML sayfasında Gönder düğmesine bastığında çağrılan işlevdir.)

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

`newMessage()` ActionScript yöntemi yalnızca bir parametre bekler, böylece JavaScript `message` değişkeni, JavaScript kodundaki `newMessage()` yöntem çağrısında bunu bir parametre olarak kullanarak ActionScript'e iletilir.

Tarayıcı türünü algılama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tarayıcıların içeriğe erişme şekli farklı olduğundan, bu örnekteki `getSWF()` JavaScript işlevinde gösterildiği gibi, pencere veya belge nesnesini kullanarak kullanıcının hangi tarayıcıyı çalıştırdığını algılamak ve tarayıcıya özgü sözdizimine göre filme erişmek için her zaman JavaScript kullanılması önemlidir:

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

Komut dosyanız kullanıcının tarayıcı türünü algılamıyorsa, kullanıcı bir HTML kabındaki SWF dosyalarını oynatırken beklenmeyen davranışlarla karşılaşabilir.

Bölüm 48: AIR'de XML imza doğrulama

Adobe AIR 1.5 ve üstü

XML İmza Sözdizimi ve İşleme için W3C öneri alt kümesine teyit edilen dijital imzaları doğrulamak için Adobe® AIR® XMLSignatureValidator API'sindeki sınıfları kullanın (http://http://www.w3.org/TR/xmldsig-core/). XML imzaları, verilerin veya bilginin bütünlüğünü ve imzalayan kimliğini doğrulamaya yardımcı olmak üzere kullanılabilir.

XML imzaları, uygulamanız tarafından indirilen mesajları veya kaynakları doğrulamak için kullanılabilir. Örneğin, uygulamanız abonelik temelinde hizmetler sağlıyorsa, imzalanmış bir XML belgesinde abonelik şartlarını kapsülleyebilirsiniz. Birisi abonelik belgesini değiştirmeye çalıştığında, doğrulama başarısız olacaktır.

XML imzasını, bu kaynakların özetini içeren imzalı bir bildirim dahil, uygulamanız tarafından kullanılan indirilmiş kaynakları doğrulamaya yardımcı olması için de kullanabilirsiniz. Uygulamanız, imzalanan dosyadaki özeti yüklenen baytlardan hesaplanan bir özetle harmanlama yoluyla kaynakların değiştirilmediğini doğrulayabilir. Bu özellikle, indirilen kaynak bir SWF dosyası veya uygulama güvenlik sanal alanında çalıştırmak istediğiniz başka bir etkin içerik olduğunda kullanışlıdır.

XML imza doğrulama temelleri

Adobe AIR 1.5 ve üstü

XML imzalarının doğrulanmasının hızlı açıklama ve kod örnekleri için Adobe Geliştirici Bağlantısı'ndaki hızlı başlatmayla ilgili makalelere bakın.

- XML imzaları oluşturma ve doğrulama (Flex)
- XML imzaları oluşturma ve doğrulama) (Flash)

Adobe® AIR®, XML imzalarının doğrulanması için XMLSignatureValidator sınıfı ve IURIDereferencer arabirimini sunar. XMLSignatureValidator sınıfı tarafından kabul edilen XML sözdizimi, XML İmza Sözdizimi ve İşleme için W3C önerisinin bir alt kümesidir. (Önerinin yalnızca bir alt kümesi desteklendiğinden, tüm kurala uygun imzalar doğrulanamaz.) AIR, XML imzaları oluşturma için API sağlamaz.

XML imza doğrulama sınıfları

Adobe AIR 1.5 ve üstü

XML imza doğrulama API'si aşağıdaki sınıfları içerir:

Paket	Sınıflar
flash.security	• XMLSignatureValidator • IURIDereferencer arabirimi) XMLSignatureValidator dizesi sabitleri aşağıdaki sınıflarda tanımlanmıştır: • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Olay • ErrorEvent

XML imza doğrulama sınıflarını kullanma

Adobe AIR 1.5 ve üstü

XML imzasını doğrulamak amacıyla XMLSignatureValidator sınıfını kullanmak için:

- Bir XMLSignatureValidator nesnesi oluşturun
- IURIDereferencer arabiriminin uygulamasını sağlayın. XMLSignatureValidator nesnesi, imzadaki her başvuru için URI ileterek IURIDereferencer `dereference()` yöntemini çağırır. `dereference()` yöntemi, URI'yi ve başvurulan verileri (aynı belgede imza olarak veya harici bir kaynakta bulunabilecek) verileri çözmelidir.
- XMLSignatureValidator nesnesinin sertifika güvenini, iptal etme denetlemesini ve başvuru doğrulama ayarlarını uygulamanıza uygun olarak ayarlayın.
- `complete` ve `error` olayları için olay dinleyicileri ekleyin.
- İmzayı doğrulanmak üzere ileterek `verify()` yöntemini çağırın.
- `complete` ve `error` olaylarını işleyin ve sonuçları yorumlayın.

Aşağıdaki örnek, XML imzasının geçerliliğini doğrulayan `validate()` işlevini uygular. XMLSignatureValidator özellikleri, imzalayıcı sertifikasının sistem güven depolama alanında olacağı veya depolama alanında bulunan bir sertifikaya ekleneceği biçimde ayarlanır. Örnek de *XMLDereferencer* adlı uygun bir IURIDereferencer sınıfının mevcut olduğunu varsayar.

```
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);
}
```

XML imza doğrulama işlemi

Adobe AIR 1.5 ve üstü

XMLSignatureValidator `verify()` yöntemini çağırdığınızda, AIR aşağıdaki adımları gerçekleştirir:

- Çalışma zamanı, imzalayıcı sertifikasının ortak anahtarını kullanarak imzanın şifreleme bütünlüğünü doğrular.
- Çalışma zamanı, XMLSignatureValidator nesnesinin geçerli ayarlarını temel alarak sertifikanın şifreleme bütünlüğünü, kimliğini ve güvenilirliğini oluşturur.

  İmzalayıcı sertifikada konumlandırılan güven, doğrulama işleminin bütünlüğü için kilit noktadır. İmza doğrulama, iyi tanımlanmış bir şifreleme işlemi kullanılarak yürütülür, ancak imzalayıcı sertifikasının güvenilirliği, işlemsel olarak varılamayacak bir yargıdır.

  Genel olarak, bir sertifikanın güvenilir olup olmadığına karar vermenin üç yolu vardır:

  - Sertifika yetkililerine ve işletim sistemi güven depolama alanına güvenerek.

- Doğrudan imzalayıcıdan sertifikanın bir kopyasını, sertifika için güven tutturucusu görevi yapan başka bir sertifikayı veya sertifikayı güvenilir biçimde tanımlamak üzere ortak anahtar gibi yeterli bir bilgiyi alarak.
- Uygulamanızın son kullanıcısına sertifikaya güvenip güvenmediğini sorarak. Sertifikadaki tanımlayıcı bilgi esas olarak güvenilir olmadığından, böyle bir sorgu kendinden imzalı sertifikalarda geçersizdir.

- Çalışma zamanı, imzalanan verilerin şifreleme bütünlüğünü doğrular.

  İmzalanan veriler, IURIDereferencer uygulamanızın yardımıyla doğrulanır. İmza belgesindeki her başvuru için IURIDereferencer uygulamasının `dereference()` yöntemi çağrılır. `dereference()` yönteminin döndürdüğü veriler, başvuru özetini hesaplamak için kullanılır. Bu özet değeri, imza belgesinde kaydedilen özetle karşılaştırılır. Özetler eşleşirse, veriler imzalanmış olduğu için değiştirilmez.

  XML imzası doğrulama sonuçlarına güvenirken dikkate alınması gereken önemli bir nokta, yalnızca imzalı olanların güvenilir olduğudur. Örneğin, bir paketteki dosyaları listeleyen imzalı bir bildirimi düşünün. XMLSignatureValidator imzayı doğrularken yalnızca bildirimin kendisinin değiştirilip değiştirilmediğini denetler. Dosyalardaki veriler imzalı değildir, yani bildirimde başvurulan dosyalar değiştirilmiş veya silinmiş olsa bile imza doğrulanacaktır.

  Not: *Böyle bir bildirimde yer alan dosyaları doğrulamak için dosya verilerinin özetini hesaplayabilir (bildirimde kullanılan karma algoritmanın aynısını kullanarak) ve imzalı bildirimde saklanan özetin sonucuyla karşılaştırabilirsiniz. Bazı durumlarda, ek dosyaların mevcudiyetini de kontrol etmeniz gerekebilir.*

Doğrulama sonuçlarını yorumlama

Adobe AIR 1.5 ve üstü

Doğrulama sonuçları, XMLSignatureValidator nesnesinin durum özelliklerinde bildirilir. Bu özellikler, doğrulayıcı nesne *complete* olayını gönderdikten sonra okunabilir. Dört durum özelliği şunlardır: `validityStatus`, `digestStatus`, `identityStatus`, and `referencesStatus`.

validityStatus özelliği

Adobe AIR 1.5 ve üstü

`validityStatus` özelliği, imzanın genel geçerliliğini bildirir. `validityStatus`, diğer üç durum özelliğinin durumuna bağlıdır ve aşağıdaki değerlerden birine sahip olabilir:

- `valid` — `digestStatus`, `identityStatus` ve `referencesStatus` özelliklerinin tümü `valid` olduğunda.
- `invalid` — Bir veya daha çok durum özelliği `invalid` olduğunda.
- `unknown` — Bir veya daha çok durum özelliği `unknown` olduğunda ve hiçbir durum `invalid` olmadığında.

digestStatus özelliği

Adobe AIR 1.5 ve üstü

`digestStatus` özelliği, mesaj özetinin şifreleme doğrulama sonuçlarını bildirir. `digestStatus` özelliği, aşağıdaki değerlerden birine sahip olabilir:

- `valid` — İmza belgesi imzalamadan sonra değiştirilmediyse.
- `invalid` — İmza belgesi değiştirildi veya yanlış biçimlendirildiyse.
- `unknown` — `verify()` yöntemi hatasız tamamlanmadıysa.

identityStatus özelliği

Adobe AIR 1.5 ve üstü

`identityStatus` özelliği, imzalayıcı sertifikasının durumunu bildirir. Bu özelliğin değeri aşağıdaki faktörlere bağlıdır:

- sertifikanın şifreleme bütünlüğü
- sertifikanın süresi dolmuş veya iptal edilmiş olup olmadığı
- kullanılan bilgisayarda sertifikaya güvenilip güvenilmediği
- XMLSignatureValidator nesnesinin durumu (örneğin güven zinciri oluşturma amacıyla ek sertifikalar eklenip eklenmediği, bu sertifikaların güvenilir olup olmadığı ve `useSystemTrustStore` ile `revocationCheckSettings` özelliklerinin değerleri)

`identityStatus` özelliği aşağıdaki değerlere sahip olabilir:

- `valid` — İmzalayıcı sertifikası, geçerli sayılmak için aşağıdaki koşulları karşılamalıdır:
  - İmzalayıcı sertifikası değiştirilmemiş olmalıdır.
  - İmzalayıcı sertifikasının süresi dolmuş veya sertifika iptal edilmiş olmamalıdır—imzada geçerli bir zaman damgası bulunması durumu hariç. İmzada zaman damgası bulunuyorsa, belge imzalandığında geçerli olduğu süre boyunca sertifika geçerli sayılır. (Zaman damgasını imzalamak için zaman damgası servisi tarafından kullanılan sertifika, kullanıcının bilgisayarında güvenilir bir kök sertifikaya eklenmelidir.)
  - İmzalayıcı sertifikası güvenlidir. Sertifika, sistem güven depolama alanında bulunuyorsa veya sistem güven depolama alanında bulunan başka bir sertifikaya eklenmişse ve `useSystemTrustStore` özelliğini true olarak ayarladıysanız, güvenlidir. Ayrıca, XMLSignatureValidator nesnesinin `addCertificate()` yöntemini kullanarak da güvenilir bir sertifika oluşturabilirsiniz.
  - Sertifika gerçekte bir imzalayıcı sertifikasıdır.
- `invalid` — Sertifikanın süresi dolmuş veya sertifika iptal edilmiştir—ve imzalama zamanındaki geçerliliği kanıtlayan bir zaman damgası yoktur—veya sertifika değiştirilmiştir.
- `unknown` — Sertifika geçersiz değilse, ancak güvenilir de değilse. Örneğin kendinden imzalı sertifikalar `unknown` olarak bildirilir (açık olarak güvenilmediği sürece). `verify()` yöntemi hatasız olarak tamamlanmadığında veya imza özeti geçersiz olduğundan kimlik denetlenmediğinde, `identityStatus` özelliği `unknown` olarak bildirilir.

referencesStatus özelliği

Adobe AIR 1.5 ve üstü

`referencesStatus` özelliği, imzanın SignedData öğesindeki başvuruların şifreleme bütünlüğünü bildirir.

- `valid` — İmzadaki her başvurunun hesaplanan özeti, XML imzasında kendisine karşılık gelen kayıtlı özetle eşleşiyorsa. `valid` durumu, imzalanan verilerin değiştirilmediğini gösterir.
- invalid — Hesaplanan herhangi bir özet, imzada kendisine karşılık gelen özetle eşleşmiyorsa.
- `unknown` — Başvuru özetleri denetlenmediyse. Genel imza özeti `invalid` veya imzalayıcı sertifikası geçersiz olduğunda, başvurular denetlenmez. `identityStatus` özelliği `unknown` olduğunda, başvurular yalnızca `referencesValidationSettingvalidOrUnknown` ise denetlenir.

XML imzaları hakkında

Adobe AIR 1.5 ve üstü

XML imzası, XML sözdiziminde temsil edilen dijital bir imzadır. XML imzasındaki veriler, imzalı bilgilerin imzalamadan sonra değiştirilmediğini doğrulamak için kullanılabilir. Ayrıca, imzalayıcı sertifikası güvenilir bir sertifika yetkilisi tarafından kullanıldıysa, imza sahibinin kimliği ortak anahtar altyapısı aracılığıyla doğrulanabilir.

XML imzası, tüm dijital veri türlerine uygulanabilir (ikili biçim veya XML biçimindeki). XML imzaları genellikle şu amaçlar için kullanılır:

- harici veya indirilmiş kaynakların değiştirilip değiştirilmediğini kontrol etme
- bilinen bir kaynaktan gelen mesajları doğrulama
- uygulama lisansını veya abonelik ayrıcalıklarını doğrulama

Desteklenen XML imzası sözdizimi

Adobe AIR 1.5 ve üstü

AIR, XML İmza Sözdizimi ve İşleme için W3C önerisinden aşağıdaki öğeleri destekler:

- Tüm çekirdek imza sözdizim öğeleri (W3C öneri belgesinin 4. bölümü)—tam olarak desteklenmeyen KeyInfo öğesi hariç
- KeyInfo öğesi yalnızca bir X509Data öğesi içermelidir
- X509Data öğesi yalnızca bir X509Certificate öğesi içermelidir
- SHA256 özet yöntemi
- RSA-SHA1 (PKCS1) imzalama algoritması
- "Yorum içermeyen kurallı XML" kurallaştırma yöntemi ve dönüştürme
- Zarflı imza dönüştürmesi
- zaman damgaları

Aşağıdaki belge, tipik bir XML imzasını gösterir (örneği basitleştirmek için şifreleme verilerinin büyük bölümü kaldırılmıştır):

```
<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>
```

İmzanın anahtar öğeleri şunlardır:

- SignedInfo — İmzalı verilere başvurular ve imzalama sırasında hesaplanan özet değerleri içerir. İmzalı veriler, XML imzasıyla aynı belgede yer alabilir veya harici olabilir.
- SignatureValue — İmza sahibinin özel anahtarıyla şifrelenmiş SignedInfo öğesinin özetini içerir.
- KeyInfo — İmzalayıcı sertifikasının yanı sıra, güven zincirinin oluşturulması için gerekli ek sertifikaları içerir. KeyInfo öğesi teknik olarak isteğe bağlı olsa da, AIR'in bu öğe bulunmadan imzayı doğrulayamayacağını dikkate alın.

Üç genel XML imza türü mevcuttur:

- Zarflı — imza, imzalamakta olduğu XML verilerinin içine eklenir.
- Zarflanan — imzalı XML verileri, Signature öğesinin içinde yer alan bir Object öğesinde bulunur.
- Ayrılan — imzalanan veriler, XML imzasının dışında yer alır. İmzalanan veriler, harici bir dosyada bulunabilir. Alternatif olarak imzayla aynı XML belgesinde olabilir, ancak Signature öğesinin üst veya alt öğesi olamaz.

XML imzaları, imzalanan verilere başvurmak için URI'leri kullanır. İmzalama ve doğrulama uygulamaları, bu URI'leri çözmek için aynı kuralları kullanmalıdır. XMLSignatureValidator sınıfını kullanırken, bir IURIDereferencer arabirimi uygulaması sağlamalısınız. Bu uygulama, URI'nin çözülmesinden ve imzalanan verileri bir ByteArray nesnesi olarak döndürmekten sorumludur. Döndürülen ByteArray nesnesi, imzadaki özetin üretildiği algoritmanın aynısı kullanılarak özetlenir.

Sertifikalar ve güven

Adobe AIR 1.5 ve üstü

Bir sertifika ortak anahtar, tanımlayıcı bilgi ve sertifikayı yayımlayan yetkiliye ait olan bir veya daha çok sertifikadan oluşur.

Bir sertifikada güven oluşturmanın iki yolu vardır. Örneğin fiziksel medyada veya SSL işlemi gibi güvenli bir dijital iletim yoluyla doğrudan imza sahibinden sertifikanın bir kopyasını alarak güven oluşturabilirsiniz. İmzalayıcı sertifikasının güvenilir olup olmadığını belirlemek için bir sertifika yetkilisine güvenebilirsiniz.

Bir sertifika yetkilisine güvenmek için, imzalayıcı sertifikası imzanın doğrulandığı bilgisayarda güvenilen bir yetkili tarafından yayımlanmış olmalıdır. Çoğu işletim sistemi üreticisi, işletim sistemi güven depolama alanına bir dizi sertifika yetkilisine ait kök sertifikalar yerleştirir. Kullanıcılar da depolama alanına sertifika ekleyebilir veya sertifikaları depolama alanından kaldırabilir.

Sertifika güvenilir bir sertifika yetkilisi tarafından yayımlanmış olsa bile, sertifikanın güvendiğiniz birine ait olup olmadığına karar vermeniz gerekir. Birçok kullanım durumunda, bu karar son kullanıcıya bırakılır. Örneğin, bir AIR uygulamasının yüklenmesi sırasında, AIR yükleyicisi kullanıcının uygulamayı yüklemek isteyip istemediğini sorarken yayımcı sertifikasından tanımlayıcı bilgiler görüntüler. Diğer durumlarda, ortak anahtarı veya diğer sertifika bilgilerini kabul edilebilir anahtarların bir listesiyle harmanlamanız gerekebilir. (Bu listeyle oynanamaması için, liste kendi imzasıyla veya AIR şifreli yerel depoda saklanarak güvenceye alınmalıdır.)

Not: *Bağımsız doğrulama—imzanın "kendinden imzalı" olması gibi—olmadan imzalayıcı sertifikasına güvenip güvenmeyeceğinize karar verebiliyor olmanızın yanı sıra, bu yolla imzayı doğrulayarak çok fazla güvence elde edemezsiniz. İmzayı kimin oluşturduğunu bilmediğiniz sürece, imzayla oynanmamış olduğuna dair güvencenin değeri çok azdır. İmza, geçerli biçimde imzalanmış sahte bir imza olabilir.*

Sertifika süresinin dolması ve sertifika iptali

Adobe AIR 1.5 ve üstü

Tüm sertifikaların süresi dolar. Sertifikalar, sertifikayı yayımlayan yetkili tarafından, örneğin sertifikaya ilişkin özel anahtar deşifre olduğunda veya çalındığında, iptal edilebilir. İmza süresi dolmuş veya iptal edilmiş bir sertifikayla imzalandıysa, imzanın bir parçası olarak zaman damgası mevcut olmadığı sürece, imza geçersiz olarak bildirilir. Zaman damgası mevcut olduğunda, sertifika imzalama sırasında geçerli olduğu sürece, XMLSignatureValidator sınıfı imzayı doğrulayacaktır.

Zaman damgası, verilerin belirli bir tarih ve zamanda imzalandığını belgeleyen bir zaman damgası servisinden imzalanmış dijital bir mesajdır. Zaman damgaları, zaman damgası yetkilileri tarafından yayımlanır ve zaman damgası yetkilisinin kendi sertifikasıyla imzalanır. Zaman damgasının geçerli sayılması için, zaman damgasında gömülü olan zaman damgası yetkili sertifikasının kullanılan bilgisayarda güvenilen bir sertifika olması gerekir. XMLSignatureValidator, zaman damgasının doğrulanmasında kullanılacak farklı bir sertifika oluşturmak için API sağlamaz.

IURIDereferencer arabirimini uygulama

Adobe AIR 1.5 ve üstü

XML imzasını doğrulamak için, IURIDereferencer arabirimi uygulamasını sağlamalısınız. Uygulama, bir XML imza belgesinin Reference öğeleri içindeki URI'lerin çözülmesinden ve özetin hesaplanabilmesi için verilerin döndürülmesinden sorumludur. İmza oluşturulduktan sonra başvurulan verilerin değiştirilip değiştirilmediğinin belirlenmesi için, hesaplanan özet imzadaki özetle karşılaştırılır.

Not: *HTML tabanlı AIR uygulamaları, XML imzalarını doğrulamak için ActionScript uygulaması içeren bir SWF kitaplığını içe aktarmalıdır. IURIDereferencer arabirimi, JavaScript'te uygulanamaz.*

IURIDerefencer arabirimi, uygulanması gereken tek bir `dereference(uri:String)` yöntemine sahiptir. XMLSignatureValidator nesnesi, imzadaki her başvuru için bu yöntemi çağırır. Bu yöntem, verileri bir ByteArray nesnesinde döndürmelidir.

Çoğu durumda, başvuran nesnenin başvurulan verileri konumlandırmasına olanak vermeye yönelik özellikler veya yöntemleri de eklemeniz gerekecektir. Örneğin, imzalanan veriler imzayla aynı belge içinde konumlandırıldıysa, XML belgesine başvuru sağlayan bir üye değişkeni ekleyebilirsiniz. `dereference()` yöntemi bu değişkeni başvurulan verileri konumlandırmak için URI ile birlikte kullanabilir. Benzer şekilde, imzalanan veriler yerel dosya sisteminde yer alan bir dizinde konumlandırıldıysa, `dereference()` yöntemi başvurulan dosyaları çözmek için bu dizine giden yolu sağlayan bir özelliğe ihtiyaç duyabilir.

XMLSignatureValidator, URI dizelerinin yorumlanması için tamamen başvuran nesneyi temel alır. URI'lere başvurmaya ilişkin standart kurallar, XML İmza ve Sözdizimi İşleme için W3C Önerisi'nin 4.3.3 bölümünde belirtilmiştir.

Zarflı imzalarda URI'lere başvurma

Adobe AIR 1.5 ve üstü

Zarflı bir XML imzası oluşturulduğunda, imza öğeleri imzalanan verilere eklenir. Örneğin, aşağıdaki mesajı bir zarflı imza yapısı kullanarak imzaladıysanız:

```
<message>
    <data>...</data>
</message>
```

Elde edilen imzalı belge şöyle görünür:

```
<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>
```

İmzanın, URI olarak boş bir dizeye sahip tek bir Reference öğesi içerdiğine dikkat edin. Bu bağlamda boş bir dize, belgenin köküne başvurur.

Ayrıca, dönüştürme algoritmasının zarflı imza dönüştürmenin uygulandığını belirttiğine dikkat edin. Zarflı imza dönüştürme uygulandığında, XMLSignatureValidator özeti hesaplamadan önce imzayı otomatik olarak belgeden siler. Bu, başvuran nesnenin verileri döndürmesi sırasında Signature öğesini kaldırması gerekmediği anlamına gelir.

Aşağıdaki örnek, zarflı imzalar için bir başvuran nesneyi gösterir:

```
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;
            }
        }
    }
}
```

Bu dereferencer sınıfı, zarflı imza belgesini `dereference()` yöntemi tarafından kullanılabilir duruma getirmek için `signedMessage` parametresini içeren yapıcı işlevini kullanır. Zarflı imzadaki başvuru her zaman imzalı verilerin köküne başvurduğundan, `dereferencer()` yöntemi belgeyi bir bayt dizisine yazar ve döndürür.

Zarflanan ve ayrılan imzalarda URI'lere başvurma

Adobe AIR 1.5 ve üstü

İmzalı veriler imzanın kendisiyle aynı belgede yer aldığında, başvurulardaki URI'ler imzalanan öğelere yönelmek için genellikle XPath veya XPointer sözdizimini kullanır. XML İmza Sözdizimi ve İşleme için W3C Önerisi, yalnızca bu sözdizimini önerir, yani uygulamanızı karşılaşmayı beklediğiniz imzalar üzerine kurmalısınız (ve desteklenmeyen sözdiziminin üstesinden başarıyla gelebilmek için uygulamaya yeterli hata denetimini eklemelisiniz).

AIR uygulamasının imzası, bir zarflanan imza örneğidir. Uygulamadaki dosyalar, Manifest öğesinde listelenmiştir. Manifest öğesi, Reference URI'sinde Manifest öğesinin kimliğine başvuran "#PackageContents" dizesi kullanılarak belirtilir.

```
<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>
```

Bu imzanın doğrulanması için başvuran nesne Reference öğesinden `"#PackageContents"` içeren URI dizesini almalı ve bir ByteArray nesnesinde Manifest öğesini döndürmelidir. "#" simgesi, öğe kimliği niteliğinin değerini belirtir.

Aşağıdaki örnek, AIR uygulama imzalarının doğrulanması için başvuran nesneyi uygular. Uygulama, bir AIR imzasının bilinen yapısı üzerine kurularak basit tutulmuştur. Genel amaçlı bir başvuran nesne, önemli ölçüde karmaşık olabilir.

```
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;
            }

        }
    }
}
```

Bu türde bir imzayı doğruladığınızda, yalnızca Manifest öğesindeki veriler doğrulanır. Paketteki gerçek dosyalar hiç denetlenmez. Paket dosyalarıyla oynanıp oynanmadığını denetlemek için dosyaları okumalı, SHA256 özetini hesaplamalı ve sonucu bildirimde kayıtlı özetle harmanlamalısınız. XMLSignatureValidator, bunun gibi ikincil başvuruları otomatik olarak kontrol etmez.

Not: *Bu örnek, imza doğrulama işlemini göstermek için sunulmuştur. AIR uygulamasında, kendi imzasını doğrulayan az sayıda kullanım bulunur. Uygulama değiştirildiyse, değişikliği yapan aracı doğrulama kontrolünü basit şekilde kaldırabilir.*

Harici kaynaklar için özet değerlerini hesaplama

Adobe AIR 1.5 ve üstü

AIR, SHA256 özetlerinin hesaplanması için yerleşik işlevler içermez, ancak Flex SDK bir SHA256 yardımcı program sınıfı içerir. SDK ayrıca, hesaplanan özeti imzada saklanan özetle harmanlamak için kullanılabilen Base64 kodlayıcı yardımcı program sınıfını da içerir.

Aşağıdaki örnek işlev, bir AIR paket bildirimde yer alan dosyaları okur ve doğrular:

```
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;
}
```

İşlev, Manifest öğesindeki tüm başvuruları dolaşır. Her başvuru için SHA256 özeti hesaplanır, base64 biçiminde kodlanır ve bildirimdeki özetle karşılaştırılır. AIR paketindeki URI'ler uygulama diziniyle ilişkili yollara başvurur. Yollar, her zaman uygulama dizini içindeki META-INF alt dizininde yer alan imza dosyasının konumu temel alınarak çözülür. Flex SHA256 sınıfının özeti bir onaltılı karakterler dizesi olarak döndürdüğüne dikkat edin. Bu dize, onaltılı dizeyle temsil edilen baytları içeren bir ByteArray nesnesine dönüştürülmelidir.

Flash CS4'te mx.utils.SHA256 ve Base64Encoder sınıflarını kullanmak için bu sınıfları konumlandırıp uygulama geliştirme dizininize kopyalayabilir veya Flex SDK kullanan sınıfları içeren bir SWF kitaplığı derleyebilirsiniz.

Harici verilere başvuran ayrılan imzalarda URI'lere başvurma

Adobe AIR 1.5 ve üstü

URI harici bir kaynağa başvurduğunda, verilere erişilmeli ve veriler bir ByteArray nesnesine yüklenmelidir. URI mutlak bir URL içeriyorsa, bu sadece dosyanın okunması veya URL'nin istenmesi anlamına gelir. En yaygın rastlanan durum olan URI'nin göreli bir yol içermesi durumunda, IURIDereferencer uygulamanız imzalanan dosyalara giden yolları çözmek için bir yol içermelidir.

Aşağıdaki örnek, başvuran birim imzalanan dosyaların çözülmesi için temel olarak oluşturulduğunda başlatılan bir File nesnesini kullanır.

```
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;
            }
        }
    }
}
```

`dereference()` işlevi, başvuru URI'si tarafından yöneltilen dosyayı basit şekilde konumlandırır, dosya içeriğini bir bayt dizisine yükler ve ByteArray nesnesini döndürür.

Not: *Uzak harici başvuruları doğrulamadan önce, uygulamanızın kötü amaçla yapılandırılmış bir imza belgesi tarafından gerçekleştirilen "eve telefon" veya benzer türde bir saldırıya karşı ne kadar hassas olabileceğini göz önünde bulundurun.*

Bölüm 49: İstemci sistemi ortamı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bu bölüm, kullanıcının sistemiyle nasıl etkileşim kurulacağını açıklar. Hangi özelliklerin desteklendiğinin nasıl belirlendiği ve varsa, kullanıcının yüklenmiş girdi yöntemi düzenleyicisi (IME) kullanılarak nasıl çok dilli uygulama oluşturulduğu gösterilmektedir. Ayrıca uygulama etki alanlarının tipik kullanımları da gösterilmektedir.

Daha fazla Yardım konusu

flash.system.System

flash.system.Capabilities

İstemci sistemi ortamının temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Daha gelişmiş uygulamaları oluşturdukça, kullanıcılarınızın işletim sistemleri ve bunların erişim işlevleri hakkında daha ayrıntılı bilgi edinmek isteyebilirsiniz. flash.system paketi aşağıdakiler gibi sistem düzeyindeki işlevlere erişmenize izin veren bir sınıf koleksiyonu içerir.

- Kodun hangi uygulama ve güvenlik etki alanında çalıştırıldığını belirleme
- Kullanıcının Flash çalışma zamanı (Flash® Player veya Adobe® AIR™ gibi) örneğinin yeteneklerini, örn. ekran boyutunu (çözünürlüğünü) ve mp3 ses gibi belirli işlevlerin kullanılabilir olup olmadığını belirleme
- IME'yi kullanarak çok dilli siteler oluşturma
- Flash çalışma zamanının kabıyla (bir HTML sayfası veya bir kap uygulaması olabilir).
- Kullanıcının panosuna bilgi kaydetme

flash.system paketi ayrıca IMEConversionMode ve SecurityPanel sınıflarını da içerir. Bu sınıflar, sırayla IME ve Security sınıflarıyla kullandığınız statik sabitleri içerir.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi önemli terimler içerir:

İşletim sistemi İçinde diğer tüm uygulamaların çalıştığı, bilgisayarda çalışan ana program—örn. Microsoft Windows, Mac OS X veya Linux®.

Pano Kopyalanan veya kesilen metin ya da öğelerin barındırıldığı ve üzerindeki öğelerin uygulamalara yapıştırıldığı, işletim sisteminin kabı.

Uygulama etki alanı SWF dosyalarında aynı ada sahip farklı sınıflar bulunması durumunda sınıfların birbirini geçersiz kılmaması için farklı SWF dosyalarında kullanılan sınıfları ayırma mekanizması.

IME (girdi yöntemi düzenleyicisi) Standart bir klavye kullanılarak karmaşık karakter veya sembolleri girmek için kullanılan bir program (veya işletim sistemi aracı).

İstemci sistemi Programlama dilinde istemci, bir kişinin bilgisayarında çalıştırılan ve tek bir kullanıcı tarafından kullanılan bir uygulama parçası (veya uygulamanın tamamı). İstemci sistemi, kullanıcının bilgisayarında temel alınan işletim sistemidir.

System sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

System sınıfı, kullanıcının işletim sistemiyle etkileşim kurmanıza ve çalışma zamanının geçerli bellek kullanımını almanıza olanak sağlayan yöntem ve özellikleri içerir. System sınıfının yöntem ve özellikleri ayrıca `imeComposition` olaylarını dinlemenize, çalışma zamanına kullanıcının geçerli kodunu kullanarak harici metin dosyalarını yüklemesini veya bunları Unicode olarak yüklemesini bildirmenize ya da kullanıcının panosunun içeriğini ayarlamanıza olanak sağlar.

Çalışma zamanında kullanıcının sistemiyle ilgili verileri alma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`System.totalMemory` özelliğini kontrol ederek, çalışma zamanının geçerli olarak kullandığı bellek miktarını (bayt cinsinden) belirleyebilirsiniz. Bu özellik, bellek kullanımını izlemenize ve bellek düzeyindeki değişiklikleri esas alarak uygulamalarınızı en iyileştirmenize olanak sağlar. Örneğin, belirli bir görsel efekt bellek kullanımında çok büyük artışa neden oluyorsa, bu efekti değiştirmeyi veya olduğu gibi kaldırmayı düşünebilirsiniz.

`System.ime` özelliği, geçerli olarak yüklü olan Girdi Yöntemi Düzenleyicisi'nin (IME) başvurusudur. Bu özellik, `addEventListener()` yöntemini kullanarak `imeComposition` olaylarını (`flash.events.IMEEvent.IME_COMPOSITION`) dinlemenize olanak sağlar.

System sınıfındaki üçüncü özellik, `useCodePage` şeklindedir. `useCodePagetrue` değerine ayarlandığında, çalışma zamanı harici metin dosyalarını yüklemek için geleneksel kod sayfasını kullanır. Bu özelliği `false` değerine ayarlarsanız, çalışma zamanına harici dosyayı Unicode olarak yorumlamasını bildirirsiniz.

`System.useCodePage` öğesini `true` değerine ayarlarsanız, metnin görüntülenmesi için, işletim sisteminin geleneksel kod sayfasının, harici metin dosyanızda kullanılan karakterleri içermesi gerektiğini unutmayın. Örneğin, Çince karakterler içeren harici bir metin dosyasını yüklerseniz, bu karakterler, İngilizce Windows kod sayfasını kullanan bir sistemde görüntülenemez; bunun nedeni, kod sayfasında Çince karakterlerin bulunmamasıdır.

Tüm platformlardaki kullanıcıların, uygulamanızda kullanılan harici metin dosyalarını görüntüleyebildiğinden emin olmak için, tüm harici metin dosyalarını Unicode olarak kodlamanız ve `System.useCodePage` öğesini varsayılan olarak `false` değerine ayarlanmış şekilde bırakmanız gerekir. Böylece, çalışma zamanı metni Unicode olarak yorumlar.

Metni panoya kaydetme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

System sınıfı, Flash çalışma zamanının belirtilen bir dize ile kullanıcının panosunun içeriklerini ayarlamasına olanak sağlayan `setClipboard()` adında bir yöntem içerir. Güvenlik nedenlerinden dolayı, kötü amaçlı sitelerin kullanıcıların panosuna en son kopyalanan verilere erişmesine olanak sağlayabileceğinden `Security.getClipboard()` diye bir yöntem yoktur.

Aşağıdaki kod, güvenlik hatası oluştuğunda, bir hata mesajının nasıl kullanıcının panosuna kopyalanabildiğini gösterir. Kullanıcı uygulamayla ilgili olası bir hatayı bildirmek istiyorsa, hata mesajı yararlı olabilir.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 ve AIR 1.0

Clipboard sınıfını, bir kullanıcı olayına yanıt olarak pano verilerini okumak ve yazmak için kullanabilirsiniz. AIR'de, panoya erişmek için uygulama sanal alanında çalışan kod için kullanıcı olayı gerekmez.

Capabilities sınıfını kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Capabilities sınıfı, geliştiricilerin uygulamanın çalıştırılmakta olduğu ortamı belirlemesine olanak sağlar. Capabilities sınıfının çeşitli özelliklerini kullanarak, kullanıcının sisteminin çözünürlüğünü, kullanıcının sisteminin erişilebilirlik yazılımını destekleyip desteklemediğini, kullanıcının işletim sisteminin dilini ve geçerli olarak yüklü olan Flash çalışma zamanının sürümünü öğrenebilirsiniz.

Capabilities sınıfındaki özellikleri kontrol ederek, uygulamanızı belirli kullanıcı ortamıyla en iyi biçimde çalışacak şekilde özelleştirebilirsiniz. Örneğin, `Capabilities.screenResolutionX` ve `Capabilities.screenResolutionY` özelliklerini kontrol ederek, kullanıcının sisteminde kullanılan görüntü çözünürlüğünü belirleyebilir ve hangi video boyutunun en uygun olabileceğine karar verebilirsiniz. Ya da harici bir mp3 dosyasını yüklemeye çalışmadan önce, kullanıcının sisteminin mp3 çalınmasını destekleyip desteklemediğini görmek için `Capabilities.hasMP3` özelliğini kontrol edebilirsiniz.

Aşağıdaki kod, istemcinin kullandığı Flash çalışma zamanı sürümünü ayrıştırmak için normal bir ifadeyi kullanır:

```
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.");
}
```

Bilgilerin veritabanında saklanması için, kullanıcının sistem özelliklerini sunucu tarafı komut dosyasına göndermek istiyorsanız, şu ActionScript kodunu kullanabilirsiniz:

```
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);
```

Yetenekler örneği: Sistem yeteneklerini algılama

Flash Player 9 ve üstü

CapabilitiesExplorer örneği, kullanıcının Flash çalışma zamanı sürümünün hangi özellikleri desteklediğini belirlemek için, flash.system.Capabilities sınıfını nasıl kullanabileceğinizi gösterir. Bu örnekte şu teknikler öğretilmektedir:

- Capabilities sınıfını kullanarak, kullanıcının Flash çalışma zamanı sürümünün hangi yetenekleri desteklediğini belirleme
- Kullanıcının tarayıcısının hangi tarayıcı ayarlarını desteklediğini belirlemek için ExternalInterface sınıfını kullanma

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. CapabilitiesExplorer uygulama dosyalarını Samples/CapabilitiesExplorer klasöründe bulabilirsiniz. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
CapabilitiesExplorer.fla veya CapabilitiesExplorer.mxml	Flash (FLA) veya Flex (MXML) içindeki ana uygulama dosyası.
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	Sistemin Capabilities öğelerini diziye ekleme, öğeleri sıralama ve tarayıcı yeteneklerini almak için ExternalInterface sınıfını kullanma gibi ana uygulama işlevlerini sağlayan sınıf.
capabilities.html	Harici API ile iletişim kurmak için gerekli JavaScript'i içeren bir HTML konteyneri.

CapabilitiesExplorer'a genel bakış

Flash Player 9 ve üstü

CapabilitiesExplorer.mxml dosyası, CapabilitiesExplorer uygulaması için kullanıcı arabiriminin kurulmasından sorumludur. Kullanıcının Flash çalışma zamanı sürümünün yetenekleri, Sahne Alanı'nda DataGrid bileşen örneğinde görüntülenir. Ayrıca bu uygulamalar bir HTML konteynerinden çalıştırılıyorsa ve harici API kullanılabiliyorsa, bu uygulamaların tarayıcı yetenekleri de görüntülenir.

Ana uygulama dosyasının `creationComplete` olayı gönderildiğinde, `initApp()` yöntemi çağrılır. `initApp()` yöntemi, com.example.programmingas3.capabilities.CapabilitiesGrabber sınıfından `getCapabilities()` yöntemini çağırır. `initApp()` yönteminin kodu şöyledir:

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

`CapabilitiesGrabber.getCapabilities()` yöntemi, Flash çalışma zamanı uygulamasının sıralanmış bir dizisini ve tarayıcı yeteneklerini döndürür, daha sonra bu dizi, Sahne Alanı'nda `capabilitiesGrid` DataGrid bileşen örneğinin `dataProvider` özelliğine ayarlanır.

CapabilitiesGrabber sınıfına genel bakış

Flash Player 9 ve üstü

CapabilitiesGrabber sınıfının statik `getCapabilities()` yöntemi, flash.system.Capabilities sınıfındaki özelliklerin her birini bir diziye (`capDP`) ekler. Daha sonra CapabilitiesGrabber sınıfında statik `getBrowserObjects()` yöntemini çağırır. `getBrowserObjects()` yöntemi, tarayıcının yeteneklerini içeren gezgin nesnesinde döngü gerçekleştirmek için harici API'yi kullanır. `getCapabilities()` yöntemi şöyledir:

```
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;
}
```

`getBrowserObjects()` yöntemi, tarayıcının gezgin nesnesinde bulunan özelliklerin her biri için bir dizi döndürür. Bu dizi, bir veya daha fazla öğe uzunluğundaysa, tarayıcı yetenekleri dizisi (`navArr`), Flash Player yetenekleri dizisinin (`capDP`) sonuna eklenir ve dizinin tamamı alfabetik olarak sıralanır. Son olarak, sıralanan dizi ana uygulama dosyasına döndürülür ve veri ızgarasını doldurur. `getBrowserObjects()` yönteminin kodu şöyledir:

```
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;
}
```

Geçerli kullanıcı ortamında harici API varsa, Flash çalışma zamanı uygulaması, tarayıcının gezgin nesnesinde döngü gerçekleştirip ActionScript'e URL olarak kodlanmış değerlerin bir dizesini döndüren JavaScript `JS_getBrowserObjects()` yöntemini çağırır. Bu dize daha sonra bir URLVariables nesnesine (`itemVars`) dönüştürülür ve çağıran komut dosyasına döndürülen `itemArr` dizisine eklenir.

JavaScript ile iletişim kurma

Flash Player 9 ve üstü

CapabilitiesExplorer uygulamasının oluşturulmasında son kısım, tarayıcının gezgin nesnesindeki her öğe üzerinde döngü gerçekleştirip geçici diziye bir ad-değer çifti eklemek için geçerli JavaScript kodunun yazılmasıdır. container.html dosyasında JavaScript `JS_getBrowserObjects()` yönteminin kodu şöyledir:

```
<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>
```

Bu kod, gezgin nesnesindeki tüm ad-değer çiftlerini barındıracak geçici bir dizinin oluşturulmasıyla başlar. Daha sonra gezgin nesnesi üzerinde, `for..in` döngüsü kullanılarak döngü gerçekleştirilir ve istenmeyen değerlere filtre uygulamak için geçerli değerin veri türü değerlendirilir. Bu uygulamada yalnızca String veya Boolean değerleri dikkate alınır ve diğer veri türleri (örn. işlevler ya da diziler) yoksayılır. Gezgin nesnesindeki her String veya Boolean değeri, `tempArr` dizisinin sonuna eklenir. Daha sonra tarayıcının ekran nesnesi üzerinde, `for..in` döngüsü kullanılarak döngü gerçekleştirilir ve sayısal değerlerin her biri `tempArr` dizisine eklenir. Son olarak, `Array.join()` yöntemi kullanılarak geçici dizi bir dizeye dönüştürülür. Dizi, ActionScript'in URLVariables sınıfını kullanarak verileri kolayca ayrıştırmasına olanak sağlayacak şekilde, sınırlayıcı olarak ve işaretini (&) kullanır.

Bölüm 50: AIR uygulamasını başlatma ve sonlandırma

Adobe AIR 1.0 ve üstü

Bu bölüm çalışan bir uygulamayı kapatmakla ilgili konular ve seçeneklerin yanı sıra, yüklenmiş bir Adobe® AIR® uygulamasını başlatmanın yollarını açıklıyor.

Not: *NativeApplication, InvokeEvent ve BrowserInvokeEvenet nesneleri yalnızca AIR uygulama sanal alanında çalışan SWF içeriği için uygundur. Tarayıcı veya tek başına oynatıcı (projektör) dahilinde veya uygulama sanal alanının dışındaki bir AIR uygulamasının içinde olup Flash Player çalışma zamanında çalışan SWF içeriği bu sınıflara erişemez.*

AIR uygulamalarını başlatma ve sonlandırmayla ilgili hızlı açıklama ve kod örnekleri için Adobe Geliştirici Bağlantısında bulunan aşağıdaki hızlı başlangıç makalelerine bakın:

- Başlangıç Seçenekleri

Daha fazla Yardım konusu

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

Uygulama başlatma

Adobe AIR 1.0 ve üstü

AIR uygulaması, kullanıcı (veya işletim sistemi) aşağıdaki eylemlerde bulunduğunda çağrılır:

- Uygulamayı masaüstü kabuğundan başlatırsa.
- Uygulamayı komut satırı kabuğunda bir komut olarak kullanırsa.
- Uygulamanın varsayılan açılış uygulaması olduğu bir dosya türü açarsa.
- (Mac OS X) Sabitleme görev çubuğundaki uygulama simgesini tıklatırsa (uygulama o anda çalışır durumda olsun veya olmasın).
- Uygulamayı yükleyiciden başlatmayı seçerse (yeni bir yükleme işleminin sonunda veya zaten yüklenmiş olan uygulamanın AIR dosyasını çift tıklatarak).
- Yüklenen sürüm, uygulama güncellemelerini kendisinin işleyeceği uyarısını verdiğinde, bir AIR uygulaması güncellemesi başlatır (uygulama tanımlayıcı dosyasına bir `<customUpdateUI>true</customUpdateUI>` bildirimi katarak).
- (iOS) Apple Push Notification servisinden (APNs) bildirim alır.
- Uygulamayı URL aracılığıyla çalıştırır.

- AIR uygulaması için tanımlayıcı bilgiler belirten `com.adobe.air.AIR launchApplication()` yöntemini çağıran bir Flash kimlik kartı veya uygulamasını barındıran bir web sayfasını ziyaret ederse. (Uygulama tanımlayıcı ayrıca tarayıcı başlatması için bir `<allowBrowserInvocation>true</allowBrowserInvocation>` bildirimi içermelidir.)

Bir AIR uygulaması çağrıldığında, AIR, NativeApplication nesnesi yoluyla `invoke` türü bir InvokeEvent nesnesi gönderir. Bir uygulama zamanının kendini başlatmasını ve olay dinleyicisini kaydetmesini sağlamak için, `invoke` olayları atılmak yerine kuyruğa gönderilir. Bir dinleyici kaydedildiğinde sıradaki tüm olaylar iletilir.

Not: *Bir uygulama tarayıcı başlatma özelliği kullanılarak çağrıldığında, uygulama zaten çalışır halde değilse NativeApplication nesnesi yalnızca bir `invoke` olayı gönderir.*

`invoke` olaylarını almak için, NativeApplication nesnesinin (`NativeApplication.nativeApplication`)`addEventListener()` yöntemini çağırın. Bir olay dinleyicisi `invoke` olayı için kaydolduğunda, kayıttan önce gerçekleşen tüm `invoke` olaylarını da alır. Sıradaki `invoke` olayları, `addEventListener()` yöntemine yapılan çağrı geri döndükten sonra kısa aralıklarla, bir kerede bir olay olmak üzere gönderilir. Bu işlem sırasında yeni bir `invoke` olayı gerçekleşirse, bu olay sıradaki bir veya daha fazla olaydan önce gönderilebilir. Bu olay sıralaması sayesinde, başlatma kodunuz çalışmadan önce gerçekleşen tüm `invoke` olaylarını işleyebilirsiniz. Bir olay dinleyicisini çalışmaya sonradan ekleseniz de (uygulama başlatıldıktan sonra), uygulamanın başlamasından itibaren gerçekleşen tüm `invoke` olaylarını alacağını göz önünde bulundurun.

AIR uygulamasının yalnızca bir örneği başlatılır. Zaten çalışmakta olan bir uygulama tekrar çağrılırsa, AIR çalışan örneğe yeni bir `invoke` olayı gönderir. Bir `invoke` olayına yanıt vermek ve uygun eylemi gerçekleştirmek (yeni bir belge penceresi açmak gibi) AIR uygulamasının sorumluluğundadır.

`InvokeEvent` nesnesi uygulamaya iletilen tüm argümanların yanı sıra, uygulamanın çağrıldığı dizini de içerir. Uygulama bir dosya türü ilişkilendirmesi nedeniyle çağrıldıysa, dosya yolunun tamamı komut satırı argümanlarına dahil edilir. Benzer şekilde uygulama bir uygulama güncellemesi nedeniyle çağrıldıysa, güncelleme AIR dosyası yolunun tamamı sağlanır.

Bir işlemde birden çok dosya açıldığında, Mac OS X'te tek bir InvokeEvent nesnesi gönderilir. Her bir dosya `arguments` dizisinde mevcuttur. Windows ve Linux'ta her dosya için ayrı bir InvokeEvent nesnesi gönderilir.

Uygulamanız `invoke` olaylarını, bir dinleyiciyi NativeApplication nesnesiyle kaydederek işleyebilir:

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

Ve bir olay dinleyicisi tanımlama:

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

Komut satırı argümanlarını yakalama

Adobe AIR 1.0 ve üstü

Bir AIR uygulamasının başlatılmasıyla ilişkilendirilen komut satırı argümanları, NativeApplication nesnesi tarafından gönderilen InvokeEvent nesnesinde iletilir. InvokeEvent `arguments` özelliği, bir AIR uygulaması çağrıldığında işletim sistemi tarafından verilen bir dizi argümanı içerir. Argümanlar göreceli dosya yolları içeriyorsa, yolları genellikle `currentDirectory` özelliğini kullanarak çözebilirsiniz.

Bir AIR programına iletilen argümanlar, çift tırnak işareti içinde olmadığı sürece beyaz boşluk sınırlı dizeler olarak kabul edilir:

Argümanlar	Dizi
tick tock	{tick,tock}
tick "tick tock"	{tick,tick tock}
"tick" “tock”	{tick,tock}
\"tick\" \"tock\"	{"tick","tock"}

Bir InvokeEvent öğesinin `currentDirectory` özelliği, uygulamanın başlatıldığı dizini temsil eden bir File nesnesi içerir.

Bir uygulama, uygulama tarafından kaydedilen türde bir dosya açıldığı için çağrıldığında, dosyanın yerel yolu komut satırı argümanlarına dize olarak dahil edilir. (Dosyada istenilen işlemin açılmasından veya gerçekleştirilmesinden uygulamanız sorumludur.) Benzer şekilde bir uygulama kendisini güncellemek üzere programlandığında (standart AIR güncelleme kullanıcı arabirimine güvenmek yerine), kullanıcı, eşleşen uygulama kimliğine sahip bir uygulama içeren AIR dosyasını çift tıklattığında AIR dosyasının yerel yolu da dahil edilir.

Dosyaya `currentDirectory` File nesnesinin `resolve()` yöntemini kullanarak erişebilirsiniz.

```
 if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

Ayrıca argümanın gerçekten bir dosya yolu olduğunu doğrulamanız gereklidir.

Örnek: Invocation olay günlüğü

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek `invoke` olayı için dinleyicilerin nasıl kaydedileceğini ve olayın nasıl işleneceğini gösterir. Örnek, alınan tüm invocation olaylarının günlüğünü tutar ve geçerli dizini ve komut satırı argümanlarını gösterir.

ActionScript örneği

```
package
{
    import flash.display.Sprite;
    import flash.events.InvokeEvent;
    import flash.desktop.NativeApplication;
    import flash.text.TextField;

    public class InvokeEventLogExample extends Sprite
    {
        public var log:TextField;

        public function InvokeEventLogExample()
        {
            log = new TextField();
            log.x = 15;
            log.y = 15;
            log.width = 520;
            log.height = 370;
            log.background = true;

            addChild(log);

            NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvoke);
        }

        public function onInvoke(invokeEvent:InvokeEvent):void
        {
            var now:String = new Date().toTimeString();
            logEvent("Invoke event received: " + now);

            if (invokeEvent.currentDirectory != null)
            {
                logEvent("Current directory=" + invokeEvent.currentDirectory.nativePath);
            }
            else
            {
```

```
                logEvent("--no directory information available--");
            }

            if (invokeEvent.arguments.length > 0)
            {
                logEvent("Arguments: " + invokeEvent.arguments.toString());
            }
            else
            {
                logEvent("--no arguments--");
            }
        }

        public function logEvent(entry:String):void
        {
            log.appendText(entry + "\n");
            trace(entry);
        }
    }
}
```

Flex örneği

```
<?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>
```

Kullanıcının oturum açması sırasında bir AIR uygulaması başlatmak

Adobe AIR 1.0 ve üstü

Bir AIR uygulaması, geçerli kullanıcı oturum açtığında NativeApplication`startAtLogin` özelliğine `true` değeri verilerek otomatik olarak başlatılacak duruma getirilebilir. Bir kez ayarlandıktan sonra, kullanıcı her oturum açtığında uygulama otomatik olarak başlatılır. Uygulama, ayar `false` olarak değiştirilene, kullanıcı ayarı işletim sisteminden elle değiştirene veya uygulama kaldırılana kadar başlangıçta açılmaya devam eder. Oturum açılırken başlatma, bir çalışma zamanı ayarıdır. Ayar yalnızca geçerli kullanıcıya yöneliktir. Ayrıca `startAtLogin` özelliğinin `true` olarak başarıyla ayarlanması için uygulama yüklenmelidir. Uygulama yüklenmemişken özellik ayarlanırsa (ADL ile başlatıldığında olduğu gibi) bir hata verilir.

Not: *Uygulama, bilgisayar sistemi başlatıldığında başlamaz. Kullanıcı oturum açtığında başlar.*

Bir uygulamanın otomatik olarak mı yoksa bir kullanıcı eylemi sonucunda mı açıldığını belirlemek için InvokeEvent nesnesinin `reason` özelliğini inceleyebilirsiniz. Özellik `InvokeEventReason.LOGIN` öğesine eşit ise uygulama otomatik olarak başlamış demektir. Diğer çağırma yolları için `reason` özelliği aşağıdaki gibi ayarlanır:

- `InvokeEventReason.NOTIFICATION` (yalnızca iOS) - Uygulama APNs ile çağırıldı. APNs hakkında daha fazla bilgi için Push bildirimlerini kullanma bölümüne göz atın.
- `InvokeEventReason.OPEN_URL` - Uygulama, başka bir uygulama veya sistem tarafından çağırıldı.
- `InvokeEventReason.Standard` - Tüm diğer durumlar.

`Reason` özelliğine erişmek için uygulamanız AIR 1.5.1 veya üzerini hedeflemelidir (uygulama tanımlayıcı dosyasında doğru ad alanı değerini ayarlayarak).

Aşağıdaki basitleştirilmiş uygulama bir invoke olayı meydana geldiğinde nasıl davranacağına karar vermek için InvokeEvent reason özelliğini kullanıyor. Reason özelliği "oturum açma" ise, uygulama arka planda kalır. Değil ise bu özellik ana uygulamayı görünür hale getirir. Bu deseni kullanan uygulama genelde oturum açma sırasında açılır, böylece arka planda işleme veya olay izlemeye devam edebilir ve kullanıcı tarafından tetiklenen bir invoke olayına yanıt olarak bir pencere açar.

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

Not: *Davranıştaki değişikliği görmek için uygulamayı paketleyin ve yükleyin.* `startAtLogin` *özelliği yalnızca yüklü uygulamalar için ayarlanabilir.*

Tarayıcıda bir AIR uygulaması başlatma

Adobe AIR 1.0 ve üstü

Bir web sitesi, tarayıcı başlatma özelliğini kullanarak, tarayıcıdan başlatılacak şekilde ayarlı olan yüklenmiş AIR uygulamasını başlatabilir. Tarayıcı başlatmaya yalnızca uygulama tanımlayıcı dosyası `allowBrowserInvocation` ayarını `true` olarak ayarladıysa izin verilir:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Uygulama tarayıcı üzerinden çağrıldığında, uygulamanın NativeApplication nesnesi bir BrowserInvokeEvent nesnesi gönderir.

BrowserInvokeEvent olaylarını almak için, AIR uygulamasında NativeApplication nesnesinin (`NativeApplication.nativeApplication`) `addEventListener()` yöntemini çağırın. Bir olay dinleyicisi BrowserInvokeEvent olayı için kaydolduğunda, kayıttan önce gerçekleşen tüm BrowserInvokeEvent olaylarını da alır. Bu olaylar `addEventListener()` yöntemine yapılan çağrı geri döndükten sonra gönderilir, ancak bu gönderme işlemi, kayıttan sonra alınmış olma ihtimali bulunan diğer BrowserInvokeEvent olaylarından önce gerçekleşmeyebilir. Bu sayede, başlatma kodunuzun çalışmasından önce gerçekleşen (uygulama başlangıçta tarayıcıdan çağrıldığında olduğu gibi) BrowserInvokeEvent olaylarını işleyebilirsiniz. Bir olay dinleyicisini çalışmaya sonradan ekleseniz de (uygulama başlatıldıktan sonra), uygulamanın başlamasından itibaren gerçekleşen tüm BrowserInvokeEvent olaylarını alacağını göz önünde bulundurun.

BrowserInvokeEvent nesnesi aşağıdaki özelliklere sahiptir:

Özellik	**Açıklama**
arguments	Uygulamaya iletilecek bir argüman (dize) dizisi.
isHTTPS	Tarayıcıdaki içeriğin https URL şemasını kullanıp (`true`) kullanmaması (`false`).
isUserEvent	Tarayıcı başlatmanın bir user olayıyla sonuçlanıp sonuçlanmaması. AIR 1.0'da, bu ayar her zaman `true` şeklindedir; AIR'de tarayıcı başlatma özelliği için bir user olayı gereklidir.
sandboxType	Tarayıcıdaki içeriğin sanal alan türü. Geçerli değerler, `Security.sandboxType` özelliğinde kullanılabilen değerlerle aynı olarak tanımlanır ve aşağıdakilerden biri olabilir: • `Security.APPLICATION` — İçerik, uygulama güvenlik sanal alanındadır. • `Security.LOCAL_TRUSTED` — İçerik, dosya sistemiyle yerel güvenlik sanal alanındadır. • `Security.LOCAL_WITH_FILE` — İçerik, dosya sistemiyle yerel güvenlik sanal alanındadır. • `Security.LOCAL_WITH_NETWORK` — İçerik, ağ iletişimiyle yerel güvenlik sanal alanındadır. • `Security.REMOTE` — İçerik, uzak (ağ) etki alanındadır.
securityDomain	`"www.adobe.com"` veya `"www.example.org"` gibi tarayıcıdaki içeriğin güvenlik etki alanı. Bu özellik yalnızca uzak güvenlik sanal alanındaki içerik için (ağ etki alanından gelen içerik) ayarlanır. Yerel veya uygulama güvenlik sanal alanındaki içerik için ayarlanmaz.

Tarayıcı başlatma özelliğini kullanırsanız, güvenlik sonuçlarını dikkate aldığınızdan emin olun. Bir web sitesi bir AIR uygulaması açtığında, BrowserInvokeEvent nesnesinin `arguments` özelliği yoluyla veri gönderebilir. Bu verileri dosya veya kod yükleme API'leri gibi duyarlı bir işlemde kullanırken dikkatli olun. Tehlike düzeyi, uygulamanın verilerle ne yaptığına bağlı olarak değişir. Yalnızca belirli bir web sitesinin uygulamayı çağırmasını bekliyorsanız, uygulama BrowserInvokeEvent nesnesinin `securityDomain` özelliğini kontrol etmelidir. Ayrıca uygulamayı çağıran web sitesinin HTTP'leri kullanmasını gerekli kılabilirsiniz, bunu da BrowserInvokeEvent nesnesinin `isHTTPS` özelliğini kontrol ederek doğrulayabilirsiniz.

Uygulama, iletilen verileri doğrulamalıdır. Örneğin bir uygulamanın URL'leri belirli bir etki alanına iletmesi bekleniyorsa, URL'lerin gerçekten de o etki alanına işaret ettiğini doğrulamalıdır. Böylece bir saldırganın, uygulamayı yanıltarak duyarlı verileri kendisine göndermesini sağlaması engellenebilir.

Hiçbir uygulama, yerel kaynaklara işaret edebilecek BrowserInvokeEvent argümanlarını kullanmamalıdır. Örneğin bir uygulama, tarayıcı tarafından iletilen bir yola dayalı olarak File nesneleri oluşturmamalıdır. Uzak yolların tarayıcı tarafından iletilmesi bekleniyorsa, uygulama, yolların uzak bir protokol yerine `file://` protokolünü kullanmasını sağlamalıdır.

Uygulama sonlandırma

Adobe AIR 1.0 ve üstü

Bir uygulamayı sonlandırmanın en hızlı yolu NativeApplication exit () yöntemini çağırmaktır. Bu yol, uygulamanızda kaydedilecek veri veya temizlenecek harici kaynaklar olmadığında işe yarar. `exit()` öğesini çağırmak tüm pencereleri kapatır ve uygulamayı sonlandırır. Ancak uygulamanızın pencerelerinin veya diğer bileşenlerinin, örneğin gerekli verileri kaydetmek üzere sonlandırma işlemini kesmesine izin vermek için, uygun uyarı olaylarını `exit()` öğesini çağırmadan önce gönderin.

Bir uygulamayı başarıyla kapatırken düşünülecek bir başka nokta, kapatma işlemi nasıl başlarsa başlasın tek bir çalışma yolu sağlamaktır. Kullanıcı (veya işletim sistemi) uygulama sonlandırmayı aşağıdaki yollarla tetikleyebilir:

- `NativeApplication.nativeApplication.autoExittrue` olduğunda son uygulama penceresini kapatarak.
- Örneğin kullanıcı varsayılan menüden uygulamadan çık komutunu seçtiğinde işletim sisteminden uygulamadan çıkış komutunu seçerek. (Bu yalnızca Mac OS'de gerçekleşir; Windows ve Linux, sistem kromu aracılığıyla uygulama çıkış komutu sağlamaz.)
- Bilgisayarı kapatarak.

Bir çıkış komutu bu rotalardan biriyle işletim sisteminde araya girdiğinde, NativeApplication bir `exiting` olayı gönderir. Hiçbir dinleyici `exiting` olayını iptal etmezse, tüm açık pencereler kapanır. Her pencere bir `closing` ve ardından bir `close` olayı gönderir. Pencerelerden biri `closing` olayını iptal ederse, kapatma işlemi durur.

Pencerelerin kapanma sırası uygulamanız için bir sorunsa, NativeApplication öğesinden `exiting` olayını dinleyin ve pencereleri kendiniz uygun bir sırayla kapatın. Bunu yapmanız gerekebilir, örneğin araç paletleri olan bir belge pencereniz varsa. Sistemin paletleri kapatması uygunsuz, hatta daha da kötü olabilirdi, ancak kullanıcı bazı verileri kaydetmek için çıkış komutunu iptal etmeye karar verdi. Windows'ta `exiting` olayını yalnızca son pencereyi kapattıktan sonra alırsınız (NativeApplication nesnesinin `autoExit` özelliği `true` değerine ayarlıysa).

Tüm platformlarda tutarlı davranış sağlamak için, çıkış sırası ister işletim sisteminden, ister menü komutlarından veya ister uygulama mantığından başlatılmış olsun, uygulamadan çıkmak için aşağıdaki uygulamaları izleyin:

1. Her zaman uygulama kodunda `exit()` öğesini çağırmadan önce NativeApplication nesnesi üzerinden bir `exiting` olayı gönderin ve uygulamanızın başka bir bileşeninin olayı iptal etmediğinden emin olun.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2. `NativeApplication.nativeApplication` nesnesinden uygulama `exiting` olayını dinleyin ve işleyicideki tüm pencereleri kapatın (önce bir `closing` olayı göndererek). Tüm pencereler kapatıldıktan sonra, uygulama verilerini kaydetmek veya geçici dosyaları silmek gibi gerekli olan tüm temizlik görevlerini yerine getirin. Uygulamadan çıkılmadan önce bittiğinden emin olmak için temizleme sırasında yalnızca senkronize yöntemler kullanın.

   Pencerelerinizin kapanma sırası önemli değilse, `NativeApplication.nativeApplication.openedWindows` dizisinde ilerleyebilir ve sırayla tüm pencereleri kapatabilirsiniz. Sıra *önemliyse*, pencerelerin doğru sırayla kapanması için bir yöntem sağlayın.

```
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 Windows her zaman kendi `closing` olaylarını dinleyerek kendi temizleme işlemini gerçekleştirmelidir.

4 Daha önceden çağrılan işleyiciler sonraki işleyicilerin `exiting` olayını iptal edip etmeyeceğini bilemeyeceği için, uygulamanızda yalnızca bir `exiting` dinleyicisi kullanın. (Çalışma sırasına güvenmek de doğru olmaz.)

Bölüm 51: AIR çalışma zamanı ve işletim sistemi bilgileriyle çalışma

Adobe AIR 1.0 ve üstü

Bu bölüm, bir AIR uygulamasının işletim sistemi dosya ilişkilendirmelerini yönetmek, kullanıcı etkinliğini saptamak ve Adobe® AIR® çalışma zamanı hakkında bilgi almak için kullandığı yolları tartışır.

Daha fazla Yardım konusu

flash.desktop.NativeApplication

Dosya ilişkilendirmelerini yönetme

Adobe AIR 1.0 ve üstü

Uygulamanız ve bir dosya türü arasındaki ilişkilendirmeler uygulama tanımlayıcıda bildirilmelidir. Yükleme işlemi sırasında, başka bir uygulama önceden varsayılan olarak olarak belirlenmediği sürece, AIR uygulama yükleyicisi AIR uygulamasını bildirilen tüm dosya türleri için varsayılan açılış uygulaması olarak ilişkilendirir. AIR uygulaması yükleme işlemi, var olan dosya türü ilişkilendirmesini geçersiz kılmaz. İlişkilendirmeyi başka bir uygulamadan devralmak için çalışma zamanında `NativeApplication.setAsDefaultApplication()` yöntemini çağırın.

Uygulamanız başladığında beklenen dosya ilişkilendirmelerinin yerinde olduğunu doğrulamak yararlı bir uygulamadır. Bunun nedeni, AIR uygulama yükleyicisinin var olan dosya ilişkilendirmelerini geçersiz kılmaması ve kullanıcı sistemindeki dosya ilişkilendirmelerinin her zaman değişebilir olmasıdır. Başka bir uygulama güncel dosya ilişkilendirmesine sahipse, var olan bir ilişkilendirmenin devralınmasından önce kullanıcının onayını almak yerinde bir uygulamadır.

NativeApplication sınıfının aşağıdaki yöntemleri, bir uygulamanın dosya ilişkilendirmelerini yönetmesine olanak verir. Yöntemlerden her biri, dosya türü uzantısını parametre olarak alır:

Yöntem	Açıklama
isSetAsDefaultApplication()	AIR uygulaması güncel olarak belirtilen dosya türüyle ilişkilendirilmişse true değerini döndürür.
setAsDefaultApplication()	AIR uygulaması ve dosya türü açık eylemi arasında ilişkilendirme oluşturur.
removeAsDefaultApplication()	AIR uygulaması ve dosya türü arasındaki ilişkilendirmeyi kaldırır.
getDefaultApplication()	Dosya türüyle güncel olarak ilişkilendirilmiş uygulama yolunu bildirir.

AIR, yalnızca ilk olarak uygulama tanımlayıcıda bildirilen dosya türlerine ilişkin ilişkilendirmeleri yönetebilir. Kullanıcı bu dosya türü ve uygulamanız arasındaki ilişkilendirmeyi elle oluşturmuş olsa bile, bildirilmemiş bir dosya türünün ilişkilendirmeleri hakkında bilgi alamazsınız. Uygulama tanımlayıcıda bildirilmemiş bir dosya türüne ilişkin uzantı aracılığıyla herhangi bir dosya ilişkilendirme yönetimi yönteminin çağırılması, uygulamanın çalışma zamanı istisnası vermesine neden olur.

Çalıştırma zamanı sürümü ve yama düzeyini alma

Adobe AIR 1.0 ve üstü

NativeApplication nesnesi, uygulamanın yürütülmekte olduğu çalışma zamanının sürümünü belirten `runtimeVersion` özelliğini içerir (`"1.0.5"` gibi bir dize). NativeApplication nesnesi ayrıca, çalışma zamanının yama düzeyini ifade eden `runtimePatchLevel` özelliğini de içerir (2960 gibi bir sayı). Aşağıdaki kod bu özellikleri kullanır:

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

AIR özelliklerini saptama

Adobe AIR 1.0 ve üstü

Adobe AIR uygulamasıyla paketlenen bir dosya için, `Security.sandboxType` özelliği `Security.APPLICATION` sabitiyle tanımlanan değere ayarlanır. Aşağıdaki kodda gösterildiği şekilde, bir dosyanın Adobe AIR güvenlik sanal alanında bulunuyor olup olmadığını temel alarak içerik (AIR'e özgü API'ler içerebilir veya içermeyebilir) yükleyebilirsiniz.

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

AIR uygulamasıyla yüklenmeyen tüm kaynaklar, bir web tarayıcısında Adobe® Flash® Player tarafından atanacağı güvenlik sanal alanlarının aynılarına atanır. Uzak kaynaklar, kaynak etki alanlarına göre sanal alanlara yerleştirilir ve yerel kaynaklar ağ iletişimiyle yerel, dosya sistemiyle yerel veya yerel güvenilir sanal alanlara yerleştirilir.

İçeriğin çalışma zamanında yürütülüp yürütülmediğini (ve bir tarayıcıda çalışan Flash Player'da çalışıp çalışmadığını) görmek için `Capabilities.playerType` statik özelliğinin `"Desktop"` olarak ayarlanıp ayarlanmadığını kontrol edebilirsiniz.

Daha fazla bilgi için bkz. “AIR güvenliği” sayfa 1088.

Kullanıcı varlığını izleme

Adobe AIR 1.0 ve üstü

NativeApplication nesnesi, bir kullanıcının bilgisayarı ne zaman etkin olarak kullandığını saptamanıza yardımcı olan iki olay gönderir. `NativeApplication.idleThreshold` özelliğiyle belirlenen aralıkta fare veya klavye etkinliği saptanmadıysa, NativeApplication bir `userIdle` olayı gönderir. Bir sonraki klavye veya fare girdisi gerçekleştiğinde, NativeApplication nesnesi bir `userPresent` olayı gönderir. `idleThreshold` aralığı saniye cinsinden ölçülür ve 300 varsayılan değerine sahiptir (5 dakika). Ayrıca, son kullanıcı girdisinden itibaren geçen saniye sayısını da `NativeApplication.nativeApplication.lastUserInput` özelliğinden elde edebilirsiniz.

Aşağıdaki kod satırları, boşta kalma eşiğini 2 dakika olarak ayarlar ve hem `userIdle` hem de `userPresent` olayını dinler::

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

Not: *Herhangi iki `userPresent` olayı arasında yalnızca bir `userIdle` olayı gönderilir.*

Bölüm 52: AIR yerel pencereleriyle çalışma

Adobe AIR 1.0 ve üstü

Masaüstü pencerelerini oluşturmak ve yönetmek için Adobe® AIR® yerel pencere API'si tarafından sağlanan sınıfları kullanırsınız.

AIR'de yerel pencerelerin temelleri

Adobe AIR 1.0 ve üstü

AIR uygulamasında yerel pencerelerle çalışma için hızlı açıklamalar ve kod örnekleri için Adobe Geliştirici Bağlantısındaki hızlı başlamayla ilgili makalelere bakın.

- Şeffaf pencere uygulaması oluşturma) (Flex)
- Bir pencereyle etkileşim kurma
- Yerel bir pencerenin görünümünü ve verdiği hissi özelleştirme (Flex)
- Pencereleri başlatma (Flex)
- Tost stili pencereler oluşturma) (Flex)
- Pencerelerin görüntü sırasını denetleme (Flex)
- Yeniden boyutlandırılabilir, dikdörtgen olmayan pencereler oluşturma (Flex)
- Bir pencereyle etkileşim kurma (Flash)
- Yerel bir pencerenin görünümünü ve verdiği hissi özelleştirme (Flash)
- Tost stili pencereler oluşturma) (Flash)
- Pencerelerin görüntü sırasını denetleme (Flash)
- Yeniden boyutlandırılabilir, dikdörtgen olmayan pencereler oluşturma (Flash)

AIR, Flash®, Flex™ ve HTML programlama tekniklerini kullanarak yerel işletim sistemi pencereleri oluşturmak için kullanımı kolay, platformlar arası bir pencere API'si sunar.

AIR'le uygulamanızın görünümünü geliştirmede büyük bir özgürlüğe sahip olursunuz. Oluşturduğunuz pencereler Mac'te çalışırken Apple stiliyle eşleşerek, Windows'ta çalışırken Microsoft uygulamalarına uyarak ve Linux'ta pencere yöneticisiyle uyum halinde çalışarak, platforma özgü bir kod satırına sahip olmadan standart bir masaüstü uygulaması gibi görünebilir. Veya uygulamanız nerede çalışırsa çalışsın kendi stilinizi oluşturmak için Flex çerçevesinin sağladığı skin desteği olan, genişletilebilir kromu kullanabilirsiniz. Masaüstüne karşı saydamlık ve alfa karışımı için tam desteğe sahip vektör ve bitmap resmi içeren kendi pencere kromunuzu bile oluşturabilirsiniz. Dikdörtgen pencerelerden sıkıldınız mı? Yuvarlak bir tane çizin.

AIR'deki Pencereler

Adobe AIR 1.0 ve üstü

AIR, pencerelerle çalışmak için üç ayrı API destekler:

- ActionScript odaklı NativeWindow sınıfı, en düşük düzeyli pencere API'sini sağlar. NativeWindows sınıfını ActionScript ve Flash Professional uygulamalarında kullanın. Uygulamanızda kullanılan pencereleri özelleştirmek için NativeWindow sınıfını genişletmeyi düşünün.
- HTML ortamında, tarayıcı tabanlı web uygulamasında olduğu gibi JavaScript Window sınıfını kullanabilirsiniz. JavaScript Window yöntemine yapılan çağrılar, temel yerel pencere nesnelerine iletilir.
- Flex çerçevesi mx:WindowedApplication ve mx:Window sınıfları, NativeWindow sınıfı için bir Flex "sarma" sağlar. Flex ile bir AIR uygulaması oluşturduğunuzda, WindowedApplication bileşeni Application bileşeninin yerini alır ve Flex uygulamanızda her zaman başlangıç penceresi olarak kullanılmalıdır.

ActionScript pencereleri

Bir NativeWindow sınıfıyla pencereler oluşturduğunuzda, Flash Player sahnesini kullanın ve listeyi doğrudan görüntüleyin. NativeWindow sınıfına görsel bir nesne eklemek için nesneyi pencere sahnesinin görüntü listesine veya sahnedeki başka bir görüntüleme nesnesi kabına ekleyin.

HTML pencereleri

HTML pencereleri oluşturduğunuzda, içeriği görüntülemek için HTML, CSS ve JavaScript kullanırsınız. HTML penceresine görsel bir nesne eklemek için, bu içeriği HTML DOM'a eklersiniz. HTML pencereleri, NativeWindow öğesinin özel bir kategorisidir. AIR ana bilgisayarı HTML pencerelerinde temel NativeWindow örneğine erişim sağlayan bir `nativeWindow` özelliği tanımlar. Bu özelliği burada açıklanan NativeWindow özellikleri, yöntemleri ve olaylarına erişmek için kullanabilirsiniz.

Not: *JavaScript Window nesnesi de, mevcut pencere için komut dosyası oluşturmaya yönelik* `moveTo()` *ve* `close()` *gibi yöntemlere sahiptir. Örtüşen yöntemler olduğunda, en uygun yöntemi kullanabilirsiniz.*

Flex Framework pencereleri

Flex çerçevesiyle pencereler oluşturduğunuzda, pencereyi doldurmak için genellikle MXML bileşenlerini kullanırsınız. Bir Flex bileşenini bir pencereye eklemek için, bileşen öğesini pencerenin MXML tanımına eklersiniz. İçeriği dinamik olarak eklemek için ActionScript de kullanabilirsiniz. mx:WindowedApplication ve mx:Window bileşenleri Flex kapları olarak tasarlanmıştır ve NativeWindow nesneleri Flex bileşenlerini doğrudan kabul edemezken, bu bileşenler edebilir. Gerektiğinde, NativeWindow özellikleri ve yöntemlerine `nativeWindow` özelliği kullanılarak WindowedApplication ve Window nesneleri aracılığıyla erişilebilir.

İlk uygulama penceresi

Uygulamanızın ilk penceresi sizin için AIR tarafından otomatik olarak oluşturulur. AIR, uygulama tanımlayıcı dosyasının `initialWindow` öğesinde belirtilen parametreleri kullanarak pencere özelliklerini ve içeriğini ayarlar.

Kök içerik bir SWF dosyası ise, AIR bir NativeWindow örneği oluşturur, SWF dosyasını yükler, ve bunu pencere sahnesine ekler. Kök içerik bir HTML dosyası ise, AIR bir HTML penceresi oluşturur ve HTML'i yükler.

Yerel pencere sınıfları

Adobe AIR 1.0 ve üstü

Yerel pencere API'si aşağıdaki sınıfları içerir:

Paket	Sınıflar
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

Yerel pencere olay akışı

Adobe AIR 1.0 ve üstü

Yerel pencereler, ilgili bileşenleri önemli bir değişikliğin gerçekleşmek üzere olduğuna veya gerçekleştiğine dair bilgilendirmek için olaylar gönderir. Birçok pencere ilişkili olay, çiftler halinde gönderilir. İlk olay, bir değişikliğin gerçekleşmek üzere olduğuna dair uyarır. İkinci olay, değişikliğin yapıldığını bildirir. Bir uyarı olayını iptal edebilirsiniz, ancak bir bildirim olayını iptal edemezsiniz. Aşağıdaki sıra, kullanıcı pencerenin büyütme düğmesini tıklattığında gerçekleşen olay akışını gösterir:

1. NativeWindow nesnesi bir `displayStateChanging` olayı gönderir.
2. Hiçbir kayıtlı kullanıcı olayı iptal etmezse, pencere büyür.
3. NativeWindow nesnesi bir `displayStateChange` olayı gönderir.

   NativeWindow nesnesi ayrıca pencere boyutu ve konumuyla ilgili değişiklikler için de olaylar gönderir. Pencere, bu ilgili değişiklikler için uyarı olayları göndermez. İlgili değişiklikler şunlardır:

   a. Büyütme işlemi nedeniyle pencerenin sol üst köşesi hareket ettiğinde `move` olayı gönderilir.
   b. Büyütme işlemi nedeniyle pencere boyutu değiştiğinde `resize` olayı gönderilir.

   Bir NativeWindow nesnesi, pencereyi simge durumuna küçültme, yeniden yükleme, kapatma, taşıma ve yeniden boyutlandırma sırasında olayları benzer bir sırayla gönderir.

   Uyarı olayları yalnızca, pencere kromu veya başka işletim sistemin tarafından kontrol edilen bir mekanizma bir değişiklik başlattığında gönderilir. Pencere boyutu, konumu veya görüntü durumunu değiştirmek için bir window yöntemini çağırdığınızda, pencere yalnızca değişikliği bildirmek için bir olay gönderir. İsterseniz pencere `dispatchEvent()` yöntemini kullanarak bir uyarı olayı gönderebilir ve değişiklik devam etmeden önce uyarı olayınızın iptal edilip edilmediğini kontrol edebilirsiniz.

   Pencere API sınıfları, yöntemleri, özellikleri ve olaylarıyla ilgili ayrıntılı bilgi için bkz.Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Yerel pencere stili ve davranışını denetleyen özellikler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki özellikler, bir pencerenin temel görünümü ve davranışını denetler:

- `type`
- `systemChrome`
- `transparent`
- `owner`

Bir pencere oluşturduğunuzda, pencere yapıcısına iletilen NativeWindowInitOptions nesnesindeki bu özellikleri ayarlarsınız. AIR, özellikleri ilk uygulama penceresi için uygulama tanımlayıcıdan okur. (Uygulama tanımlayıcıda ayarlanamayan ve her zaman `normal` olarak ayarlanmış olan `type` özelliği hariç.) Bu özellikler pencere oluşturulduktan sonra değiştirilemez.

Bu özelliklerin bazı ayarları birbirleriyle uyumsuzdur: `transparent` özelliği `true` olarak veya `type` özelliği `lightweight` olarak ayarlandığında `systemChrome` özelliği `standard` olarak ayarlanamaz.

Pencere türleri

Adobe AIR 1.0 ve üstü

AIR pencere türleri, üç işlevsel pencere türü oluşturmak için yerel işletim sisteminin krom ve görünürlük niteliklerini birleştirir. Koddaki tür adlarına başvurmak için NativeWindowType sınıfında tanımlanan sabitleri kullanın. AIR, aşağıdaki pencere türlerini sağlar:

Tür	Açıklama
Normal	Tipik bir pencere. Normal pencereler tam boyutlu krom stilini kullanır ve Windows görev çubuğu ile Mac OS X pencere menüsünde görünürler.
Hizmet penceresi	Bir araç paleti. Hizmet pencereleri sistem kromunun daha ince bir versiyonunu kullanır ve Windows görev çubuğu ile Mac OS X pencere menüsünde görünmezler.
Hafif	Hafif pencereler krom içermez ve Windows görev çubuğunda veya Mac OS X pencere menüsünde görünmezler. Ayrıca, hafif pencereler Windows'ta Sistem (Alt+Ara çubuğu) menüsü içermezler. Hafif pencereler, açılır kutular gibi kısa süreli görüntü alanları açan bildirim kabarcıkları ve denetimler için uygundur. `type` özelliği hafif olarak ayarlandığında `systemChrome` özelliği `none` olarak ayarlanmalıdır.

Pencere kromu

Adobe AIR 1.0 ve üstü

Pencere kromu, kullanıcıların masaüstü ortamında bir pencereyi değiştirmesine olanak veren denetimlerin kümesidir. Krom öğeleri, başlık çubuğu, başlık çubuğu düğmeleri, kenarlık ve yeniden boyutlandırma kıskaçlarını içerir.

Sistem kromu

`systemChrome` özelliğini `standard` veya `none` olarak ayarlayabilirsiniz. Pencerenize kullanıcının işletim sistemi tarafından oluşturulan ve stillendirilen standart denetimler kümesini vermek için `standard` sistem kromunu seçin. Pencere için kendi kromunuzu sağlamak istiyorsanız `none` olarak ayarlayın. Koddaki sistem kromu ayarlarına başvurmak için NativeWindowSystemChrome sınıfında tanımlanan sabitleri kullanın.

Sistem kromu sistem tarafından yönetilir. Uygulamanızın denetimlerin kendisine doğrudan erişimi yoktur, ancak denetimler kullanıldığında gönderilen olaylara tepki verebilir. Pencere için standart krom kullandığınızda, `transparent` özelliği `false` olarak ve `type` özelliği `normal` veya `utility` olarak ayarlanmalıdır.

Flex kromu

Flex WindowedApplication veya Window bileşenlerini kullandığınızda pencere, sistem kromunu veya Flex çerçevesi tarafından sağlanan kromu kullanabilir. Flex kromunu kullanmak için, pencere oluşturmak üzere kullanılan `systemChrome` özelliğini `none` olarak ayarlayın. mx bileşenleri yerine Flex 4 spark bileşenlerini kullandığınızda, Flex kromu kullanabilmek için skin sınıfını belirtmelisiniz. Dahili kaplamaları veya kendi kaplamanızı kullanabilirsiniz. Aşağıdaki örnek, dahili spark WindowedApplication dış görünüm sınıfının pencere kromu sağlamak için nasıl kullanılacağını gösterir:

```
<?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>
```

Daha fazla bilgi için bkz. Flex 4'ü Kullanma: AIR pencere kapları hakkında: Pencere kromunu kontrol etme

Özel krom

Sistem kromu olmadan bir pencere oluşturduğunuzda, kullanıcı ve pencere arasındaki etkileşimleri işlemek için kendi krom denetimlerinizi eklemeniz gerekir. Dikdörtgen olmayan, şeffaf pencereler de yapabilirsiniz.

mx:WindowedApplication veya mx:Window bileşenleri ile özel krom kullanmak için, `showFlexChrome` stilini `false` değerine ayarlamalısınız. Yoksa, Flex pencerelerinize kendi kromunu ekler.

Pencere şeffaflığı

Adobe AIR 1.0 ve üstü

Bir pencerenin masaüstü veya diğer pencerelerle alfa karışımına izin vermek için `transparent` özelliğini `true` olarak ayarlayın. `transparent` özelliği pencere oluşturulmadan ayarlanmalıdır ve değiştirilemez.

Şeffaf bir pencerenin varsayılan bir arka planı yoktur. Uygulama tarafından çizilen, nesne içermeyen pencere alanları görünmezdir. Görüntülenen bir nesne birden küçük bir alfa ayarına sahipse, aynı penceredeki, diğer pencerelerdeki ve masaüstündeki görüntüleme nesneleri de dahil nesnenin altındaki her şey görünür.

Şeffaf pencereler, şekil açısından düzensiz olan, “yavaşça kaybolan” veya görünmez olan kenarlıklara sahip uygulamalar oluşturmak istediğinizde kullanışlıdır. Ancak, Alfa karışımlı büyük alanlar oluşturmak uzun sürebilir, bu nedenle efekt tutumlu kullanılmalıdır.

Önemli: *Linux'ta fare olayları tam saydam piksellerden geçmez. Büyük, tam saydam alanlar içeren pencereler oluşturmaktan kaçınmalısınız, çünkü bu şekilde kullanıcının masaüstündeki diğer pencere ve öğelere erişimini engelleyebilirsiniz. Mac OS X ve Windows'ta, fare olayları tam saydam piksellerden geçer.*

Şeffaflık, sistem kromu içeren pencerelerle kullanılamaz. Ayrıca, HTML'deki SWF ve PDF içeriği şeffaf pencerelerde görüntülenmeyebilir. Daha fazla bilgi için, bkz. "HTML sayfasına SWF ve PDF içeriği yüklenirken dikkate alınması gerekenler" sayfa 1015.

Statik `NativeWindow.supportsTransparency` özelliği, pencere şeffaflığının kullanılabilir olup olmadığını bildirir. Şeffaflık desteklenmediğinde, uygulama siyah bir arka plana karşı birleştirilir. Bu durumlarda, uygulamanın tüm saydam alanları opak siyah olarak görüntülenir. Bu özelliğin `false` değerinde olması ihtimaline karşı bir destek sağlamakta fayda vardır. Örneğin, kullanıcıya bir uyarı iletişim kutusu veya geri dönüşlü, dikdörtgen, şeffaf olmayan bir kullanıcı arabirimi görüntüleyebilirsiniz.

Şeffaflığın Mac ve Windows işletim sistemleri tarafından her zaman desteklendiğini unutmayın. Linux işletim sistemlerinde destek, bileşik pencere yöneticisini gerektirir, ancak bileşik pencere yöneticisi etkin olduğunda bile, kullanıcı görüntü seçenekleri veya donanım yapılandırması nedeniyle saydamlık kullanılamayabilir.

MXML uygulama penceresinde şeffaflık

Adobe AIR 1.0 ve üstü

Pencereyi *şeffaf* olarak oluştursanız bile, MXML penceresinin arka planı varsayılan olarak opaktır. (Pencerenin köşelerindeki şeffaflık efektine dikkat edin.) Pencereye şeffaf bir arka plan oluşturmak için, stil sayfasındaki arka plan rengi ve alfa değerini veya uygulamanızın MXML dosyasında bulunan <mx:Style> öğesini ayarlayın. Örneğin, aşağıdaki stil bildirimi arka plana biraz şeffaf, yeşil bir gölge verir:

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

HTML uygulama penceresinde şeffaflık

Adobe AIR 1.0 ve üstü

İçinde bulunduğu pencere şeffaf olsa bile, HTML pencereleri ve HTMLLoader nesnelerinde görüntülenen HTML içeriğinin arka planı varsayılan olarak opaktır. HTML içeriği için görüntülenen varsayılan arka planı kapatmak için, `paintsDefaultBackground` özelliğini `false` olarak ayarlayın. Aşağıdaki örnek bir HTMLLoader oluşturur ve varsayılan arka planı kapatır:

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

Bu örnek, bir HTML penceresinin varsayılan arka planını kapatmak için JavaScript kullanır:

```
window.htmlLoader.paintsDefaultBackground = false;
```

HTML belgesinde bulunan bir öğe, bir arka plan rengi ayarlıyorsa, bu öğenin arka planı şeffaf olmaz. Kısmi bir şeffaflık (veya opaklık) değerinin ayarlanması desteklenmez. Ancak, benzer bir görsel efekt elde etmek için PNG biçimindeki şeffaf bir grafiği bir sayfanın veya sayfa öğesinin arka planı olarak kullanabilirsiniz.

Pencere sahipliği

Bir pencere bir ya da daha fazla başka pencereye *sahip olabilir*. Bu sahip olunan pencereler her zaman ana pencerenin önünde görüntülenir, ana pencereyle birlikte simge durumuna getirilip geri yüklenir ve ana pencere kapatıldığında kapanır. Pencere sahibi olma durumu başka bir pencereye aktarılamaz veya kaldırılamaz. Bir pencere yalnızca tek bir ana pencerenin olabilir, ancak sınırsız sayıda pencereye sahip olabilir.

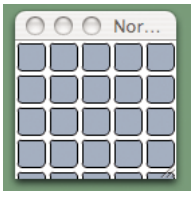
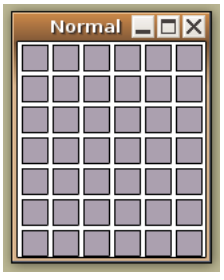
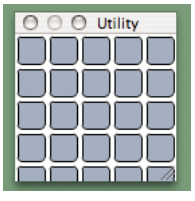
Pencere sahipliğini araç paletleri ve iletişim kutuları için kullanılan pencerelerin yönetimini kolaylaştırmak için kullanabilirsiniz. Örneğin, Kaydet iletişim kutusunu bir belge penceresiyle ilişkili bir şekilde görüntülediyseniz, belge penceresinin iletişim kutusunun sahibi olması iletişim kutusunu otomatik olarak belge penceresinin önünde tutar.

- NativeWindow.owner
- Christian Cantrell: Owned windows in AIR 2.6 (AIR 2.6'da sahibi olan pencereler)

Görsel bir pencere kataloğu

Adobe AIR 1.0 ve üstü

Aşağıdaki tablo, Mac OS X, Windows ve Linux işletim sistemlerinde pencere özellik ayarlarının farklı bileşimlerinin görsel efektlerini gösterir:

Pencere ayarları	Mac OS X	Microsoft Windows	Linux*
Type: normal SystemChrome: standart Transparent: false			
Type: hizmet SystemChrome: standart Transparent: false			

Pencere ayarları	Mac OS X	Microsoft Windows	Linux*
Type: Herhangi biri SystemChrome: yok Transparent: false			
Type: Herhangi biri SystemChrome: yok Transparent: true			
mxWindowedApplication veya mx:Window Type: Herhangi biri SystemChrome: yok Transparent: true	mx:Win...	mx:Window	mx:Window

**Compiz pencere yöneticisi ile Ubuntu*

***Not:** Aşağıdaki sistem kromu öğeleri AIR'de desteklenmez: Mac OS X Araç Çubuğu, Mac OS X Proxy Simgesi, Windows başlık çubuğu simgeleri ve diğer sistem kromu.*

Pencereler oluşturma

Adobe AIR 1.0 ve üstü

AIR, bir uygulama için ilk pencereyi otomatik olarak oluşturur, ancak ihtiyacınız olan ek pencereleri oluşturabilirsiniz. Yerel bir pencere oluşturmak için NativeWindow yapıcı yöntemini kullanın.

HTML penceresi oluşturmak için HTMLLoader `createRootWindow()` yöntemini kullanın veya bir HTML belgesinden JavaScript `window.open()` yöntemini çağırın. Oluşturulan pencere, görüntüleme listesi bir HTMLLoader nesnesi içeren bir NativeWindow nesnesidir. HTMLLoader nesnesi pencere için HTML ve JavaScript içeriği yorumlar ve görüntüler. `window.nativeWindow` özelliğini kullanarak JavaScript öğesinin altındaki NativeWindow nesnesinin özelliklerine erişebilirsiniz. (Bu özellik yalnızca AIR uygulama sanal alanında çalışan kod için erişilebilirdir.)

Bir pencereyi başlattığınızda—ilk uygulama penceresi dahil—pencereyi görünmez durumda oluşturmayı, içeriği yüklemeyi veya her türlü grafik güncellemesini yapmayı ve ardından pencereyi görünür yapmayı düşünmelisiniz. Bu sıralama, herhangi bir uyumsuz görsel değişikliğin kullanıcılarınıza görünmesini önler. Uygulamanızın ilk penceresinin görünmez durumda oluşturulacağını, uygulama açıklayıcısında `<visible>false</visible>` etiketini belirterek (veya varsayılan değer `false` olduğundan etiketi bütünüyle dışarıda bırakarak) ifade edebilirsiniz. Yeni NativeWindows varsayılan olarak görünmezdir. HTMLLoader `createRootWindow()` yöntemiyle bir HTML penceresi oluştururken, `visible` bağımsız değişkenini `false` olarak ayarlayabilirsiniz. Bir pencereyi görünür kılmak için NativeWindow `activate()` yöntemini çağırın veya `visible` özelliğini `true` olarak ayarlayın.

Pencere başlatma özelliklerini belirleme

Adobe AIR 1.0 ve üstü

Bir yerel pencerenin başlatma özellikleri, masaüstü penceresi oluşturulduktan sonra değiştirilemez. Bu sabit özellikler ve varsayılan değerleri şunlardır:

Özellik	Varsayılan değer
systemChrome	standard
type	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

Uygulama tanımlayıcı dosyasında AIR tarafından oluşturulan ilk pencereye ilişkin özellikleri ayarlayın. AIR uygulamasının ana penceresine ilişkin type özelliği her zaman *normal* olarak ayarlanır. (Tanımlayıcı dosyada `visible`, `width` ve `height` gibi ek pencere özellikleri belirtilebilir, ancak bu özellikler her zaman değiştirilebilir.)

Uygulamanız tarafından NativeWindowInitOptions sınıfı kullanılarak oluşturulan diğer yerel pencereler ve HTML pencereleri için özellikleri ayarlayın. Bir pencere oluşturduğunuzda, pencere özelliklerini belirten bir NativeWindowInitOptions nesnesini NativeWindow yapıcı işlevine veya HTMLLoader `createRootWindow()` yöntemine iletmelisiniz.

Aşağıdaki kod, bir hizmet penceresi için NativeWindowInitOptions nesnesi oluşturur:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

Transparent özelliği `true` olarak veya `type` özelliği `lightweight` *olarak ayarlandığında systemChrome özelliğinin standart olarak ayarlanması desteklenmez.*

Not: *JavaScript `window.open()` işleviyle oluşturulan bir pencere için başlatma özelliklerini ayarlayamazsınız. Ancak, kendi HTMLHost sınıfınızı uygulayarak bu pencerelerin oluşturulma biçimini geçersiz kılabilirsiniz. Daha fazla bilgi için bkz. "window.open() için JavaScript çağrılarını işleme" sayfa 1026.*

Flex mx:Window sınıfıyla bir pencere oluşturduğunuzda, pencere nesnesinin kendisine ait başlatma özelliklerini pencereye ilişkin MXML bildiriminde veya pencereyi oluşturan kodda belirtin. Alttaki NativeWindow nesnesi siz `open()` yöntemini çağırmadan oluşturulmaz. Pencere açıldığında, bu başlatma özellikleri değiştirilemez.

İlk uygulama penceresini oluşturma

Adobe AIR 1.0 ve üstü

AIR, ilk uygulama penceresini uygulama tanımlayıcıda belirtilen özellikler temelinde oluşturur ve içerik öğesinde başvurulan dosyayı yükler. İçerik öğesi bir SWF veya HTML dosyasına referans vermelidir.

Başlangıç penceresi, uygulamanızın ana penceresi olabilir veya yalnızca bir veya daha çok farklı pencerenin başlatılması için görev yapabilir. Bu pencereyi görünür yapmanız gerekmez.

İlk pencereyi ActionScript ile oluşturma

Adobe AIR 1.0 ve üstü

ActionScript kullanarak bir AIR uygulaması oluşturduğunuzda, uygulamanızın ana sınıfı Sprite sınıfını (veya Sprite öğesinin bir alt sınıfını) genişletmelidir. Bu sınıf, uygulama için ana giriş noktası olarak görev yapar.

Uygulamanız başladığında, AIR bir pencere oluşturur, ana sınıfın bir örneğini oluşturur ve örneği pencere sahnesine ekler. Pencereye erişmek için `addedToStage` olayını dinleyebilir ve daha sonra NativeWindow nesnesine başvuru almak için Stage nesnesinin `nativeWindow` özelliğini kullanabilirsiniz.

Aşağıdaki örnek, ActionScript ile oluşturulan AIR uygulamasının ana sınıfına ilişkin temel çatıyı gösterir:

```
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
        }
    }
}
```

Not: *Teknik olarak, ana sınıfın yapıcı işlevindeki `nativeWindow` özelliğine ERİŞEBİLİRSİNİZ. Ancak, bu yalnızca ilk uygulama penceresine uygulanan özel bir durumdur.*

Flash Professional'da bir uygulama oluştururken, ayrı bir ActionScript dosyasında kendi belge sınıfınızı oluşturmazsanız, ana belge sınıfı otomatik olarak oluşturulur. Sahne `nativeWindow` özelliğini kullanarak ilk pencerenin NativeWindow nesnesine erişebilirsiniz. Örneğin, aşağıdaki kod ana pencereyi büyütülmüş durumda etkinleştirir. (zaman çizgisinden):

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

İlk pencereyi Flex ile oluşturma

Adobe AIR 1.0 ve üstü

Flex çatısıyla bir AIR uygulaması oluştururken, ana MXML dosyanızın kök öğesi olarak mx:WindowedApplication öğesini kullanın. (mx:Application bileşenini kullanabilirsiniz, ancak bu, AIR'de bulunan tüm özellikleri desteklemez.) WindowedApplication bileşeni, uygulama için ilk giriş noktası olarak görev yapar.

Uygulamayı başlattığınızda, AIR yerel bir pencere oluşturur, Flex çerçevesini başlatır ve WindowedApplication nesnesini pencere sahnesine ekler. Başlatma sırası tamamlandığında, WindowedApplication bir `applicationComplete` olayı gönderir. WindowedApplication örneğinin `nativeWindow` özelliğiyle masaüstü pencere nesnesine erişin.

Aşağıdaki örnek, x ve y koordinatlarını ayarlayan basit bir WindowedApplication bileşeni oluşturur:

```
<?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>
```

Bir NativeWindow oluşturma

Adobe AIR 1.0 ve üstü

Bir NativeWindow oluşturmak için, NativeWindowInitOptions nesnesini NativeWindow yapıcısına iletin:

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

Siz `visible` özelliğini `true` olarak ayarlayana veya `activate()` yöntemini çağırana kadar pencere görüntülenmez.

Pencere oluşturulduğunda, özelliklerini başlatabilir ve sahne özelliğiyle Flash görüntü listesi tekniklerini kullanarak pencereye içerik yükleyebilirsiniz.

Neredeyse her durumda, yeni bir yerel pencerenin `scaleMode` özelliğini `noScale` olarak ayarlamalısınız. (`StageScaleMode.NO_SCALE` sabitini kullanın.) Flash ölçek modları, uygulama geliştiricisinin uygulama görüntüsüne ait en boy oranını önceden bilmediği durumlar için tasarlanmıştır. Ölçek modları, geliştiricinin en uygun dengeyi seçmesine olanak verir: içeriği kesme, uzatma veya sıkıştırma, ya da boş alanla doldurma. AIR'de görüntü alanını (pencere karesi) denetleyebildiğiniz için, pencereyi içeriğe veya içeriği pencereye göre ödün vermeden boyutlandırabilirsiniz.

Flex ve HTML için otomatik olarak `noScale` ölçek modu ayarlanır.

Not: *Geçerli işletim sisteminde izin verilen maksimum ve minimum pencere boyutlarını belirlemek için aşağıdaki statik NativeWindow özelliklerini kullanın:*

```
var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

Bir HTML penceresi oluşturma

Adobe AIR 1.0 ve üstü

Bir HTML penceresi oluşturmak için JavaScript `Window.open()` yöntemini veya AIR HTMLLoader sınıfının `createRootWindow()` yöntemini çağırabilirsiniz.

Herhangi bir güvenlik sanal alanındaki HTML özelliği JavaScript `Window.open()` yöntemini kullanabilir. İçerik uygulama sanal alanının dışında çalışıyorsa, `open()` yöntemi yalnızca fare tıklatması veya tuşa basma gibi bir kullanıcı etkileşimine yanıt olarak çağrılabilir. `open()` yöntemi çağrıldığında, belirtilen URL'deki içeriği görüntülemek için sistem kromu içeren bir pencere oluşturulur. Örneğin:

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

Not: *JavaScript `window.open()` işleviyle oluşturulan pencereyi özelleştirmek için ActionScript'teki HTMLHost sınıfını genişletebilirsiniz. Bkz. "HTMLHost sınıfının genişletilmesi hakkında" sayfa 1019.*

Uygulama güvenlik sanal alanındaki içerik, daha güçlü `HTMLLoader.createRootWindow()` pencere oluşturma yöntemine erişime sahiptir. Bu yöntemle, yeni bir pencereye ilişkin tüm oluşturma seçeneklerini belirleyebilirsiniz. Örneğin, aşağıdaki JavaScript kodu 300x400 boyutunda, sistem kromu içermeyen hafif bir pencere oluşturur:

```
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"));
```

Not: *Yeni bir pencere tarafından yüklenen içerik uygulama güvenlik sanal alanının dışındaysa, pencere nesnesi AIR özelliklerine sahip değildir: `runtime`, `nativeWindow` veya `htmlLoader`.*

Saydam bir pencere oluştururseanız, bu pencereye yüklenen HTML'deki gömülü SWF içeriği her zaman görüntülenmez. SWF dosyasına başvurmak için kullanılan gömme etiketinin veya nesnenin parametresinin `wmode` öğesini `opaque` veya `transparent` olarak ayarlamalısınız. `wmode` öğesinin varsayılan değeri `window` öğesidir, bu yüzden varsayılan olarak SWF içeriği saydam pencerelerde görüntülenmez. Hangi `wmode` ayarlanırsa ayarlansın, PDF içeriği saydam pencerelerde görüntülenemez. (AIR 1.5.2'den önce, SWF içeriği de saydam pencerelerde görüntülenemiyordu.)

`createRootWindow()` yöntemiyle oluşturulan pencereler, açılan pencereden bağımsız kalırlar. JavaScript Window nesnesinin `parent` ve `opener` özellikleri `null` değerindedir. Açılan pencere, `createRootWindow()` işlevinin döndürdüğü HTMLLoader başvurusunu kullanarak yeni pencerenin Window nesnesine erişebilir. Önceki örnek bağlamında, `newHTMLLoader.window` ifadesi oluşturulan pencerenin JavaScript Window nesnesine başvurur.

Not: *`createRootWindow()` işlevi, JavaScript ve ActionScript'ten çağrılabilir.*

mx:Window oluşturma

Adobe AIR 1.0 ve üstü

Bir mx:Window başlatmak için, kök etiketi olarak mx:Window kullanarak bir MXML dosyası oluşturabilir veya doğrudan Window sınıf yapıcısını çağırabilirsiniz.

Aşağıdaki örnek, Window yapıcısını çağırarak bir mx:Window oluşturur ve görüntüler:

```
 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);
```

Bir pencereye içerik ekleme

Adobe AIR 1.0 ve üstü

Bir AIR penceresine nasıl içerik eklediğiniz, pencerenin türüne bağlıdır. Örneğin, MXML ve HTML, pencerenin temel içeriğini tanıtıcı biçimde tanımlamanızı sağlar. Kaynakları uygulama SWF dosyalarına gömebilir veya ayrı uygulama dosyalarından yükleyebilirsiniz. Flex, Flash ve HTML içeriği anında oluşturulabilir ve bir pencereye dinamik olarak eklenebilir.

SWF içeriği veya JavaScript içeren HTML içeriği yüklerken, AIR güvenlik modelini dikkate almalısınız. Uygulama güvenlik sanal alanındaki, yani uygulamanızla yüklenen ve app ile yüklenebilen her içerik: URL şeması, tüm AIR API'lerine erişmek için tüm ayrıcalıklara sahiptir. Bu sanal alanın dışından yüklenen her içerik AIR API'lerine erişemez. Uygulama sanal alanının dışındaki JavaScript içeriği JavaScript Window nesnesinin `runtime`, `nativeWindow` veya `htmlLoader` özelliklerini kullanamaz.

Güvenli çapraz komut dosyası oluşturmayı sağlamak için, uygulama içeriği ve uygulama dışı içerik arasında sınırlı bir arabirim sağlamak üzere sanal alan köprüsünü kullanabilirsiniz. HTML içeriğinde, bu sayfadaki kodun harici içeriğe ilişkin çapraz komut dosyası oluşturmasına izin vermek için uygulamanızın sayfalarını uygulama dışı bir sanal alana eşleyebilirsiniz. Bkz. “AIR güvenliği” sayfa 1088.

SWF dosyası veya resmi yükleme

`flash.display.Loader` sınıfını kullanarak yerel bir pencerenin görüntü listesine Flash SWF dosyaları veya görüntüler yükleyebilirsiniz:

```
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);
        }
    }
}
```

Not: *ActionScript 1 veya 2 kullanılarak oluşturulan daha eski SWF dosyaları, aynı pencereye yüklendiklerinde sınıf tanımları, tek ve genel değişkenler gibi genel durumları paylaşırlar. Buna benzer bir SWF dosyası doğru çalışmak için dokunulmamış genel durumlara dayanıyorsa, aynı pencereye bir defadan fazla veya örtüşen sınıf tanımları veya değişkenleri kullanılarak başka bir SWF dosyasıyla aynı pencereye yüklenemez. Bu içerik ayrı pencerelere yüklenebilir.*

Bir NativeWindow öğesine HTML içeriği yükleme

Bir NativeWindow'a HTML içeriği yüklemek için HTMLLoader nesnesini pencere sahnesine ekleyip HTML içeriğini HTMLLoader öğesine yükleyebilir veya `HTMLLoader.createRootWindow()` yöntemini kullanarak önceden bir HTMLLoader nesnesi içeren bir pencere oluşturabilirsiniz. Aşağıdaki örnek, HTML içeriğini yerel pencere sahnesinde 300x500 piksel boyutundaki görüntü alanında görüntüler:

```
//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) );
```

Bir Flex uygulamasına HTML sayfası yüklemek için Flex HTML bileşenini kullanabilirsiniz.

SWF dosyasına başvurmak içi kullanılan gömme etiketinin veya nesnenin `wmode` parametresi `opaque` veya `transparent` olarak ayarlanmadığı sürece, pencere saydamlık kullanıyorsa (yani pencerenin `transparent` özelliği `true` ise) HTML dosyasındaki SWF içeriği görüntülenmez. Varsayılan `wmode` değeri `window` olduğundan, varsayılan olarak SWF içeriği saydam dosyada görüntülenmez. Hangi `wmode` değeri kullanılırsa kullanılsın, PDF içeriği saydam pencerede görüntülenmez.

Ayrıca, HTMLLoader denetimi ölçeklenirse, döndürülürse veya HTMLLoader `alpha`özelliği 1.0.'dan farklı bir değere ayarlanırsa SWF içeriği de PDF içeriği de görüntülenmez.

SWF içeriğini HTML penceresine bir kaplama olarak ekleme

HTML pencereleri bir NativeWindow örneğinin içinde bulunduğundan, Flash görüntü nesnelerini görüntü listesindeki HTML katmanının üstüne ve altına ekleyebilirsiniz.

Bir görüntü nesnesini HTML katmanının üstüne eklemek için `window.nativeWindow.stage` özelliğinin `addChild()` yöntemini kullanın. `addChild()` yöntemi, katmanlı içeriği pencerede bulunan tüm içeriğin üzerine ekler.

HTML katmanının altına bir görüntü nesnesi eklemek için, `index` parametresine ilişkin sıfır değerini ileterek, `window.nativeWindow.stage` özelliğinin `addChildAt()` yöntemini kullanın. Bir nesnenin sıfır dizinine yerleştirilmesi, HTML görüntüsü de dahil varolan içeriği bir katman yukarı taşır ve yeni içeriği bunun altına ekler. HTML sayfasının altındaki katmanlı içeriğin görünür olması için `HTMLlLoader` nesnesinin `paintsDefaultBackground` özelliğini `false` olarak ayarlamalısınız. Ayrıca, bir arka plan rengi ayarlayan sayfanın her öğesi şeffaf olmayacaktır. Örneğin sayfanın gövde öğesi için bir arka plan rengi ayarladığınızda, sayfanın hiçbir yeri şeffaf olmaz.

Aşağıdaki örnek, Flash görüntü nesnelerinin bir HTML sayfasına kaplama ve altlık olarak nasıl ekleneceğini gösterir. Örnek, iki basit şekil nesnesi oluşturarak bunlardan birini HTML içeriğinin altına, diğerini üstüne ekler. Örnek, `enterFrame` olayını temel alan şekil konumunu da günceller.

```
 <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>
```

Bu örnek, AIR'de ActionScript ve JavaScript arasındaki sınırları aşan bazı gelişmiş tekniklere temel bir giriş sağlar. ActionScript görüntüleme nesnelerini iyi bilmiyorsanız, *ActionScript 3.0 Geliştirici Rehberi* içindeki "Görüntü programlama" sayfa 153 bölümüne bakın.

Örnek: Yerel bir pencere oluşturma

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, yerel bir pencerenin nasıl oluşturulacağını gösterir:

```
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();
}
```

Pencereleri yönetme

Adobe AIR 1.0 ve üstü

Masaüstü pencerelerinin görünümünü, davranışını ve yaşam döngüsünü yönetmek için NativeWindow sınıfının özelliklerini ve yöntemlerini kullanırsınız.

Not: *Flex çerçevesini kullanırken, pencere davranışını çerçeve sınıflarını kullanarak yönetmek genellikle daha iyidir. NativeWindow özelliklerinin ve yöntemlerinin çoğuna mx:WindowedApplication ve mx:Window sınıfları üzerinden erişilebilir.*

Bir NativeWindow örneği alma

Adobe AIR 1.0 ve üstü

Bir pencereyi değiştirmek için önce pencere örneğini almalısınız. Pencere örneğini aşağıdaki konumlardan birinden alabilirsiniz:

- Pencere oluşturmak için kullanılan yerel pencere yapıcısı:

  ```
  var win:NativeWindow = new NativeWindow(initOptions);
  ```

- Pencere sahnesinin `nativeWindow` özelliği:

  ```
  var win:NativeWindow = stage.nativeWindow;
  ```

- Penceredeki görüntüleme nesnesinin `stage` özelliği:

  ```
  var win:NativeWindow = displayObject.stage.nativeWindow;
  ```

- Pencere tarafından gönderilen yerel pencere olayının `target` özelliği:

  ```
  private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
  {
      var win:NativeWindow = event.target as NativeWindow;
  }
  ```

- Pencerede görüntülenen HTML sayfasının `nativeWindow` özelliği:

  ```
  var win:NativeWindow = htmlLoader.window.nativeWindow;
  ```

- NativeApplication nesnesinin `activeWindow` ve `openedWindows` özellikleri:

  ```
  var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
  var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
  ```

  `NativeApplication.nativeApplication.activeWindow` uygulamanın etkin penceresine başvurur. (Ancak, etkin pencere bu AIR uygulamasına ait bir pencere değilse `null` değerini döndürür.)
  `NativeApplication.nativeApplication.openedWindows` dizisi, AIR uygulamasındaki kapatılmamış tüm pencereleri içerir.

Flex mx:WindowedApplication ve mx:Window nesneleri görüntüleme nesnesi olduğundan, `stage` özelliğini aşağıdaki şekilde kullanıp bir MXML dosyasında uygulama penceresine kolayca başvuru yapabilirsiniz:

```
 <?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
```

Not: *WindowedApplication veya Window bileşeni Flex çerçevesi tarafından pencere sahnesine eklenene kadar,* `stage` *özelliği* `null` *değerindedir. Bu davranış Flex Application bileşeninin davranışıyla tutarlıdır, ancak WindowedApplication ve Window bileşenlerinin başlatma döngüsünde önceden gerçekleşen* `creationComplete` *gibi olaylara ilişkin dinleyicilerde sahneye veya NativeWindow örneğine erişmenin olanaksız olduğu anlamına gelir.* `applicationComplete` *olayı gönderildiğinde sahneye ve NativeWindow örneğine erişmek güvenlidir.*

Pencereleri etkinleştirme, gösterme ve gizleme

Adobe AIR 1.0 ve üstü

Bir pencereyi etkinleştirmek için NativeWindow `activate()` yöntemini çağırın. Bir pencerenin etkinleştirilmesi pencereyi öne çıkarır, ona klavye ve fare odağı kazandırır ve gerektiğinde pencereyi yeniden yükleyerek veya `visible` özelliğini `true` olarak ayarlayarak pencereyi görünür duruma getirir. Bir pencereni etkinleştirilmesi uygulamadaki diğer pencerelerin sırasını değiştirmez. `Activate()` yönteminin çağrılması, pencerenin `activate` olayı göndermesine neden olur.

Bir pencereyi etkinleştirmeden göstermek için `visible` özelliğini `true` olarak ayarlayın. Bu, pencereyi öne çıkarır, ancak odağı pencereye atamaz.

Bir pencereyi görünümden gizlemek için, `visible` özelliğini `false` olarak ayarlayın. Pencerenin gizlenmesi pencerenin ve tüm ilişkili görev çubuğu simgelerinin ve Mac OS X'te Pencereler menüsündeki girişin görüntülenmesini engeller.

Bir pencerenin görünürlüğünü değiştirdiğinizde, bu pencerenin sahip olduğu diğer pencerelerin görünürlüğü de değişir. Örneğin, bir pencereyi gizlerseniz, sahip olduğu tüm pencereler de gizlenir.

Not: *Mac OS X'te dock pencere bölümünde bir simgeye sahip olan simge durumuna küçültülmüş bir pencereyi tam olarak gizlemek olanaksızdır. Simge durumuna küçültülen bir pencerede* `visible` *özelliği* `false` *olarak ayarlandıysa, pencereye ilişkin dock simgesi görüntülenmeye devam eder. Kullanıcı simgeyi tıklattığında, pencere görünür bir duruma yeniden yüklenir ve görüntülenir.*

Pencere görüntü sırasını değiştirme

Adobe AIR 1.0 ve üstü

AIR, pencerelerin görüntü sırasını doğrudan değiştirmek için birkaç yöntem sağlar. Bir pencereyi görüntü sırasının önüne veya arkasına, bir pencereyi başka bir pencerenin üzerine veya ardına taşıyabilirsiniz. Aynı zamanda, kullanıcı pencereleri etkinleştirerek yeniden sıralayabilir.

Bir pencerenin `alwaysInFront` özelliğini `true` olarak ayarlayarak, bu pencereyi diğer pencerelerin önünde tutabilirsiniz. Birden çok pencere bu ayara sahip olduğunda, bu pencerelerin görüntü sırası kendi aralarında belirlenir, ancak bu pencereler her zaman `alwaysInFront` özelliği false olarak ayarlanmış pencerelerin önünde sıralanır.

AIR uygulaması etkin olmadığında bile, en üst grupta yer alan pencereler diğer uygulamalardaki pencerelerin üzerinde görüntülenir. Bu davranış kullanıcı için zorlayıcı olabileceğinden, `alwaysInFront` özelliği yalnızca gerekli ve uygun olduğunda `true` olarak ayarlanmalıdır. Uygun kullanım örnekleri şunları içerir:

- Araç ipuçları, açılır listeler, özel menüler ve açılır kutular gibi denetimler için geçici açılır pencereler. Odağı kaybettiklerinde bu tür pencerelerin kapatılmaları gerektiğinden, kullanıcının başka bir pencereyi görüntülemesinin engellenmesi can sıkıcı olmayabilir.
- Çok acil hata mesajları ve uyarılar. Kullanıcının zamanında yanıt vermemesi durumunda kurtarılamayacak bir değişikliğin gerçekleşme olasılığı, bir uyarı penceresini ekranın önüne taşımayı gerekli kılabilir. Ancak çoğu hata ve uyarı, normal pencere görüntü sırasında işlenebilir.
- Kısa süreli tost stili pencereler.

Not: *AIR,* `alwaysInFront` *özelliğinin uygun kullanımını zorlamaz. Ancak, uygulamanız bir kullanıcının iş akışını aksatıyorsa, uygulamanın kullanıcının çöp kutusuna gönderilme ihtimali de yüksektir.*

Bir pencere başka pencerelere sahipse, sahip olduğu pencereler onun önünde sıralanır. `orderToFront()` öğesini çağırır veya başka pencereleri olan bir pencerede `alwaysInFront` öğesini `true` olarak ayarlarsanız, sahip olunan pencereler sahip pencereyle birlikte diğer pencerelerin önünde tekrar sıralanır ancak sahip olunan pencereler hala sahip pencerenin önünde görüntülenir.

Sahip olunan pencerelerde sıralama yöntemlerini çağırmak aynı pencere tarafından sahip olunan pencereler arasında normal bir şekilde çalışır, ancak grubun dışında pencerelerle karşılaştırıldığında sahip olunan pencere grubunun tümünün sırasını değiştirebilir. Örneğin, sahip olunan bir pencerede `orderToFront()` öğesini çağırırsanız hem o pencere, hem sahibi hem de aynı sahibe ait olan diğer pencereler pencere görüntüleme sırasının önüne getirilir.

NativeWindow sınıfı, bir pencerenin diğer pencerelere göre görüntü sırasını ayarlamak için aşağıdaki özellikleri ve yöntemleri sağlar:

Üye	Açıklama
alwaysInFront özelliği	Pencerenin en üst pencere grubunda görüntülenip görüntülenmediğini belirtir. Neredeyse her durumda, `false` en iyi ayardır. `false` değerinin `true` olarak değiştirilmesi pencereyi diğer tüm pencerelerin önüne getirir. (Ancak pencereyi etkinleştirmez.) `true` değerinin `false` olarak değiştirilmesi pencereyi en üst grupta kalan pencerelerin arkasında, ancak hala diğer pencerelerin önünde sıralar. Özelliğin bir pencere için geçerli değerine ayarlanması, pencere görüntü sırasını değiştirmez. `alwaysInFront` ayarının başka bir pencerenin sahip olduğu pencereler üzerinde bir etkisi yoktur.
orderToFront()	Pencereyi öne getirir.
orderInFrontOf()	Pencereyi belirli bir pencerenin önüne getirir.
orderToBack()	Pencereyi diğer pencerelerin arkasına gönderir.
orderBehind()	Pencereyi doğrudan belirli bir pencerenin arkasına gönderir.
activate()	Pencereyi öne geçirir (pencereyi görünür duruma getirerek ve odağı pencereye atayarak).

Not: *Bir pencere gizli (`visible` özelliği `false`) veya simge durumundaysa, görüntü sırası yöntemlerinin etkisi olmaz.*

Linux işletim sisteminde, farklı pencere yöneticileri pencere görüntü sırasına ilişkin farklı kurallar uygular:

- Bazı pencere yöneticilerinde, yardımcı program pencereleri her zaman normal pencerelerin önünde görüntülenir.
- Bazı pencere yöneticilerinde, `alwaysInFront` özelliği `true` olarak ayarlanmış tam ekranlı bir pencere, her zaman `alwaysInFront` özelliği `true` olan diğer pencerelerin önünde görüntülenir.

Bir pencereyi kapatma

Adobe AIR 1.0 ve üstü

Bir pencereyi kapatmak için `NativeWindow.close()` yöntemini kullanın.

Bir pencerenin kapatılması pencere içeriğini boşaltır, ancak diğer nesneler bu içeriğe başvuruyorsa, içerik nesneleri ortadan kaldırılmaz. `NativeWindow.close()` yöntemi senkronize olmayan biçimde yürütülür, pencerede bulunan uygulama kapatma işlemi sırasında çalışmaya devam eder. Close yöntemi, kapatma işlemi başarılı olduğunda bile bir close olayı gönderir. NativeWindow nesnesi teknik olarak hala geçerlidir, ancak çoğu özellik ve yönteme kapatılmış bir pencerede erişmek, bir IllegalOperationError oluşturur. Kapatılan bir pencereyi yeniden açamazsınız. Pencerenin kapanıp kapanmadığını test etmek için pencerenin `closed` özelliğini kontrol edin. Bir pencereyi görünümden gizlemek için `NativeWindow.visible` özelliğini `false` olarak ayarlayın.

`Nativeapplication.autoExit` özelliği varsayılan `true` değerinde olduğunda, son pencere kapandığında uygulama kapanır.

Bir sahibi bulunan tüm pencereler, sahip pencere kapatıldığında kapatılır. Sahip olunan pencereler bir kapanma olayı göndermezler bu nedenle kapatılma olayını engelleyemezler. Bir close olayı gönderilir.

Pencere işlemlerinin iptaline izin verme

Adobe AIR 1.0 ve üstü

Bir pencere sistem kromu kullandığında, uygun olayların varsayılan davranışı dinlenerek veya iptal edilerek kullanıcının pencereyle etkileşimi iptal edilebilir. Örneğin, bir kullanıcı sistem kromu kapatma düğmesini tıklattığında, `closing` olayı gönderilir. Bir kayıtlı dinleyici olayın `preventDefault()` yöntemini çağırdığında, pencere kapanmaz.

Pencere sistem kromu kullanmıyorsa, hedeflenen değişikliklerin bildirim olayları değişiklik yapılmadan önce otomatik olarak gönderilmez. Bu nedenle, bir pencereyi kapatmaya, pencere durumunu değiştirmeye veya pencere sınırı özelliklerinden birini ayarlamaya ilişkin yöntemleri çağırırsanız, değişiklik iptal edilemez. Pencere değişikliği yapılmadan uygulamanızdaki bileşenleri bilgilendirmek için, uygulama mantığınız pencerenin `dispatchEvent()` yöntemini kullanarak ilgili bildirim olayını gönderebilir.

Örneğin, aşağıdaki mantık bir pencere kapatma düğmesi için iptal edilebilir bir olay işleyicisi uygular:

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

Olay `preventDefault()` yöntemi bir dinleyici tarafından çağrılırsa, `dispatchEvent()` yöntemi `false` değerini döndürür. Ancak, bu yöntem başka nedenlerle de `false` değerini döndürebilir, bu nedenle değişikliğin iptal edilmesi gerekip gerekmediğini test etmek için açıkça `isDefaultPrevented()` yöntemini kullanmak daha uygundur.

Bir pencereyi büyütme, simge durumuna küçültme ve geri yükleme

Adobe AIR 1.0 ve üstü

Pencereyi büyütmek için NativeWindow `maximize()` yöntemini kullanın.

```
myWindow.maximize();
```

Pencereyi simge durumuna küçültmek için NativeWindow `minimize()` yöntemini kullanın.

```
myWindow.minimize();
```

Pencereyi geri yüklemek için (yani, simge durumuna küçültülmeden veya büyütülmeden önceki boyutuna geri döndürmek için), NativeWindow `restore()` yöntemini kullanın.

```
myWindow.restore();
```

Sahibi olan bir pencere, sahip olan pencere simge durumuna küçültüldüğünde veya geri yüklendiğinde simge durumuna küçültülür ya da geri yüklenir. Sahip olunan pencere simge durumuna küçültüldüğünde sahibi de simge durumuna küçültüldüğünden herhangi bir olay gönderilmez.

Not: *Bir AIR penceresini büyütme sonucunda ortaya çıkan davranış, Mac OS X standart davranışından farklıdır. Uygulama tarafından tanımlanmış "standart" bir boyut ve kullanıcı tarafından ayarlanan son boyut arasında geçiş yapmak yerine, AIR pencereleri uygulama veya kullanıcı tarafından ayarlanan son boyut ve ekranın tam kullanılabilir alanı arasında geçiş yapar.*

Linux işletim sisteminde, farklı pencere yöneticileri pencere görüntü durumunun ayarlanmasına ilişkin farklı kurallar uygular:

- Bazı pencere yöneticilerinde, yardımcı program pencereleri ekranı kaplayamaz.
- Pencere için maksimum boyut ayarlandıysa, bazı pencereler pencerenin ekranı kaplamasına izin vermez. Diğer bazı pencere yöneticileri, ekranı kaplaması gereken görüntü durumunu ayarlar, ancak pencereyi yeniden boyutlandırmaz. Her iki durumda da, görüntü durumu değişiklik olayı gönderilmez.
- Bazı pencere yöneticileri pencerenin `maximizable` veya `minimizable` ayarlarına önem vermez.

Not: *Linux'ta pencere özellikleri senkronize olmayan şekilde değişir. Programınızın bir satırındaki görüntüleme durumunu değiştirir ve sonraki satırdaki değeri okursanız, okunan değer hala eski ayarı yansıtacaktır. Tüm platformlarda, NativeWindow nesnesi görüntüleme durumu değiştiğinde* `displayStateChange` *olayını gönderir. Pencerenin yeni durumunu temel alarak bir eylemde bulunmanız gerekirse, bunu her zaman bir* `displayStateChange` *olay işleyicisinde yapın. Bkz. "Pencere olaylarını dinleme" sayfa 925.*

Örnek: Bir pencereyi simge durumuna küçültme, büyütme, geri yükleme ve kapatma

Adobe AIR 1.0 ve üstü

Aşağıdaki kısa MXML uygulaması Window `maximize()`, `minimize()`, `restore()` ve `close()` yöntemlerini gösterir:

```
<?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>
```

Flash için aşağıdaki ActionScript örneği, NativeWindow `minimize()`, `maximize()`, `restore()` ve `close()` yöntemlerini tetikleyen dört tıklatılabilir metin alanı oluşturur:

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

Bir pencereyi yeniden boyutlandırma ve taşıma

Adobe AIR 1.0 ve üstü

Bir pencere sistem kromu kullandığında, krom, pencerenin yeniden boyutlandırılması ve masaüstünde taşınması için sürükleme denetimleri sağlar. Pencere sistem kromu kullanmıyorsa, kullanıcının pencereyi yeniden boyutlandırması ve taşımasını olanaklı kılmak için kendi denetimlerinizi eklemelisiniz.

Not: *Bir pencereyi yeniden boyutlandırmak veya taşımak için, önce NativeWindow örneğine bir başvuru almalısınız. Bir pencere başvurusunun nasıl alınacağı hakkında bilgi için* *"Bir NativeWindow örneği alma" sayfa 916.*

Bir pencereyi yeniden boyutlandırma

Kullanıcının pencereyi etkileşimli olarak yeniden boyutlandırmasına izin vermek için NativeWindow `startResize()` yöntemini kullanın. Bu yöntem bir `mouseDown` olayından çağrıldığında, yeniden boyutlandırma işlemi fareyle başlatılır ve işletim sistemi bir `mouseUp` olayı aldığında tamamlanır. `startResize()` yöntemini çağırırken, pencerenin yeniden boyutlandırılacağı ilk kenar ve köşeyi belirten bir argüman iletirsiniz.

Pencere boyutunu programlama yoluyla ayarlamak için pencerenin `width`, `height` ve `bounds` özelliklerini istenen boyutlara göre ayarlayın. Sınırları belirledikten sonra, pencere boyutu ve konum aynı anda değiştirilebilir. Ancak, değişikliklerin oluştuğu sıra garanti edilmez. Bazı Linux pencere yöneticileri, pencerelerin masaüstü ekranının sınırları dışına genişlemesine izin vermez. Bu durumlarda, aksi halde değişikliklerin net etkisi kurala uygun bir pencereyle sonuçlanacak olsa da, son pencere boyutu özelliklerin ayarlandığı sıra nedeniyle sınırlanabilir. Örneğin, ekranın altında yer alan pencerenin hem yüksekliğini hem de y konumunu değiştirdiğinizde, yükseklik değişikliği y konumu değişmeden önce uygulandığından tam yükseklik değişikliği gerçekleşmeyebilir.

Not: *Linux'ta pencere özellikleri senkronize olmayan şekilde değişir. Bir pencereyi programınızın bir satırında yeniden boyutlandırırsanız ve bir sonrakinde boyutları okursanız, eski ayarları yansıtmaya devam edeceklerdir. Tüm platformlarda, NativeWindow nesnesi pencere yeniden boyutlandırıldığında* `resize` *olayını gönderir. Penceredeki kontrolleri pencerenin yeni boyutu veya durumuna dayalı olarak düzenlemek gibi bir eylemde bulunmanız gerekirse, bunu her zaman* `resize` *olay işleyicisinde yapın. Bkz.* *"Pencere olaylarını dinleme" sayfa 925.*

Sahnenin ölçekleme modu, pencere sahnesi ve içeriklerinin pencere yeniden boyutlandırıldığında nasıl davranacağını belirler. Sahne ölçekleme modlarının, web tarayıcısında olduğu gibi, uygulamanın görüntü alanının boyutu veya en boy oranının denetiminde olmadığı durumlar için tasarlanmış olduğunu unutmayın. Genellikle `scaleMode` özelliğini `StageScaleMode.NO_SCALE` olarak ayarladığınızda en iyi sonuçları alırsınız. Yine de, pencere içeriğinin ölçeklenmesini istiyorsanız, değişen pencere sınırlarına yanıt olarak içeriğin `scaleX` ve `scaleY` parametrelerini ayarlayabilirsiniz.

Bir pencereyi taşıma

`Bir pencereyi yeniden boyutlandırmadan ta ımak için NativeWindow startMove()` yöntemini kullanın. `startResize()` yöntemi gibi, `startMove()` yöntemi bir `mouseDown` olayından çağrıldığında, taşıma işlemi fareyle başlatılmıştır ve işletim sistemi bir `mouseUp` olayı aldığında tamamlanır.

`startResize()` ve `startMove()` yöntemleriyle ilgili ayrıntılı bilgi için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Bir pencereyi programlanabilir bir şekilde taşımak için, pencerenin `x`, `y`, veya `bounds` özelliklerini istenilen konuma ayarlayın. Sınırları belirledikten sonra, pencere boyutu ve konum aynı anda değiştirilebilir.

Not: *Linux'ta pencere özellikleri senkronize olmayan şekilde değişir. Programınızın bir satırında bir pencere taşırsanız ve sonraki satırdaki değeri okursanız, okunan değer hala eski ayarı yansıtacaktır. Tüm platformlarda, NativeWindow nesnesi konum değiştiğinde`move` olayını gönderir. Pencerenin yeni konumunu temel alarak bir eylemde bulunmanız gerekirse, bunu her zaman bir `move` olay işleyicisinde yapın. Bkz. "Pencere olaylarını dinleme" sayfa 925.*

Örnek: Pencereleri yeniden boyutlandırma ve taşıma

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek bir pencerede yeniden boyutlandırma ve taşıma işlemlerinin nasıl başlatılacağını gösterir:

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

Pencere olaylarını dinleme

Adobe AIR 1.0 ve üstü

Bir pencerenin gönderdiği olayları dinlemek için, pencere örneğiyle bir dinleyiciyi kaydedin. Örneğin, closing olayını dinlemek için, bir dinleyiciyi pencereyle aşağıdaki gibi kaydedin:

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

Bir olay gönderildiğinde `target` özelliği olayı gönderen pencereye başvurur.

Çoğu pencere olayının iki ilişkili mesajı vardır. İlk mesaj, bir pencere değişikliği yapılacağına (ve bunun iptal edilebileceğine) işaret ederken, ikinci mesaj değişikliğin gerçekleştiğini bildirir. Örneğin, kullanıcı pencerenin kapatma düğmesini tıklattığında, closing olayı mesajı gönderilir. Hiçbir dinleyici olayı iptal etmezse, pencere kapatılır ve close olayı tüm dinleyicilere gönderilir.

Genellikle `closing` gibi uyarı olayları, yalnızca bir olayı tetiklemek için sistem kromu kullanıldıysa gönderilir. Örneğin, pencere `close()` yönteminin çağrılması otomatik olarak `closing` olayını göndermez—yalnızca `close` olayı gönderilir. Ancak, bir closing olayı nesnesi oluşturabilir ve bunu pencerenin `dispatchEvent()` yöntemini kullanarak gönderebilirsiniz.

Bir Event nesnesi gönderen pencere olayları:

Olay	Açıklama
activate	Pencere odak aldığında gönderilir.
deactivate	Pencere odağı kaybettiğinde gönderilir.
closing	Pencere kapatılmak üzere olduğunda gönderilir. Bu otomatik olarak yalnızca sistem kromu kapatma düğmesine basıldığında veya Mac OS X'te Çıkış komutu çağrıldığında gerçekleşir.
close	Pencere kapatıldığında gönderilir.

Bir NativeWindowBoundsEvent nesnesi gönderen pencere olayları:

Olay	Açıklama
moving	Pencerenin sol üst köşesi taşıma, yeniden boyutlandırma veya pencere görüntü durumu değişikliği sonucunda konumunu değiştirmeden hemen önce gönderilir.
move	Sol üst köşe konumunu değiştirdikten sonra gönderilir.
resizing	Pencere genişliği veya yüksekliği, yeniden boyutlandırma veya görüntü durumu değişikliği sonucunda değişmeden hemen önce gönderilir.
resize	Pencere boyut değiştirdikten sonra gönderilir.

NativeWindowBoundsEvent olayları için, bekleyen veya tamamlanan bir değişiklikten önce ve sonraki pencere sınırlarını belirlemek üzere `beforeBounds` ve `afterBounds` özelliklerini kullanabilirsiniz.

Bir NativeWindowDisplayStateEvent nesnesi gönderen pencere olayları:

Olay	Açıklama
displayStateChanging	Pencere görüntü durumu değişmeden hemen önce gönderilir.
displayStateChange	Pencere görüntü durumu değiştikten sonra gönderilir.

NativeWindowDisplayStateEvent olayları için, bekleyen veya tamamlanan bir değişiklikten önce ve sonraki pencere görüntü durumunu belirlemek üzere `beforeDisplayState` ve `afterDisplayState` özelliklerini kullanabilirsiniz.

Bazı Linux pencere yöneticilerinde, maksimum boyuta ayarına sahip bir pencere büyütüldüğünde, görüntü durum değişikliği olayı gönderilmez. (Pencere maksimum görüntü durumuna ayarlanır, ancak yeniden boyutlandırılmaz.)

Tam ekran pencereleri görüntüleme

Adobe AIR 1.0 ve üstü

Sahne'nin `displayState` özelliğini `StageDisplayState.FULL_SCREEN_INTERACTIVE` olarak ayarlamak, pencereyi tam ekran moduna getirir, ancak bu modda klavye girişine *izin verilir*. (Bir tarayıcıda çalışan SWF içeriğinde, klavye girişine izin verilmez.) Kullanıcı, tam ekran modundan çıkmak için Escape tuşuna basar.

Not: *Bazı Linux pencere yöneticileri, pencere için maksimum boyut ayarlı olduğunda, pencere boyutlarını ekranı kaplamak üzere değiştirmez. (Ancak pencere sistem kromunu kaldırır.)*

Örneğin, aşağıdaki Flex kodu, basit bir tam ekran uçbirimi kuran basit bir AIR uygulaması tanımlar:

```
 <?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>
```

Flash için aşağıdaki ActionScript örneği, basit bir tam ekranlı metin uçbirimi simüle eder:

```
    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);
        }
    }
```

Bölüm 53: AIR'de görüntü ekranları

Adobe AIR 1.0 ve üstü

Bilgisayara veya cihaza bağlı görüntü ekranlarıyla ilgili bilgiye erişmek için Adobe® AIR® Screen sınıfını kullanın.

Daha fazla Yardım konusu

flash.display.Screen

AIR'de görüntüleme ekranlarının temelleri

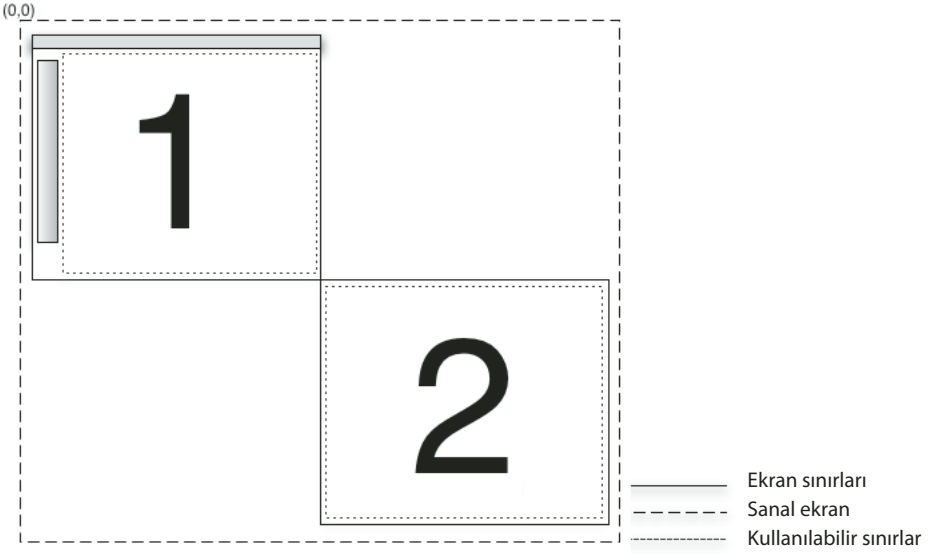
Adobe AIR 1.0 ve üstü

- Sanal masaüstünü ölçme (Flex)
- Sanal masaüstünü ölçme) (Flash)

Ekran API'si, sistem ekran bilgisinin elde edilmesi için statik üyeler ve belirli bir ekranın açıklanması için örnek üyeler sağlayan Screen adlı tek bir sınıf içerir.

Bilgisayar sistemi, sanal alana yerleştirilmiş çeşitli masaüstü ekranlarına karşılık gelen, kendisine eklenmiş birkaç monitör ve görüntü içerebilir. AIR Screen sınıfı ekranlar, ekranların göreceli olarak yerleştirilmesi ve kullanılabilir alanları hakkında bilgi sağlar. Aynı ekrana birden çok monitör eşleniyorsa, yalnızca bir ekran mevcut olur. Ekran boyutu monitörün görüntü alanından büyükse, ekranın hangi bölümünün geçerli olarak görünür olduğunu belirlemenin bir yolu yoktur.

Ekran, bağımsız bir masaüstü görüntü alanını temsil eder. Ekranlar, görsel masaüstü içinde dikdörtgenler olarak tanımlanır. Birincil görüntü olarak atanan ekranın sol üst köşesi, görsel masaüstünün koordinat sisteminin başlangıç noktasıdır. Ekranı tanımlamak için kullanılan tüm deperler piksel cinsinden verilir.



Bu ekran yerleştirmesinde, görsel masaüstünde iki ekran bulunuyor. Ana ekranın üst sol köşesinin koordinatları (1 No'lu) her zaman (0,0)'dır. Ekran yerleştirmesi ekran #2'yi ana ekran olarak belirleyecek biçimde değiştirilirse, ekran #1'in koordinatları negatif olur. Ekrana ilişkin kullanılabilir sınırlar bildirilirken menü çubukları, görev çubukları ve dock'lar hariç tutulur.

Ekran API sınıfı, yöntemleri, özellikleri ve olaylarıyla ilgili ayrıntılı bilgi için bkz. Adobe Flash Platformu için ActionScript 3.0 Başvurusu.

Ekranları numaralandırma

Adobe AIR 1.0 ve üstü

Sanal masaüstü ekranlarını aşağıdaki ekran yöntemleri ve özellikleriyle numaralandırabilirsiniz:

Yöntem veya Özellik	Açıklama
Screen.screens	Kullanılabilir ekranları açıklayan bir dizi Screen nesnesi sağlar. Dizinin sırası önemli değildir.
Screen.mainScreen	Ana ekran için bir Screen nesnesi sağlar. Mac OS X'te ana ekran her zaman menü çubuğunu görüntüleyen ekrandır. Windows'ta ana ekran, sistem tarafından belirlenen birincil ekrandır.
Screen.getScreensForRectangle()	Belirtilen dikdörtgenle kesişen ekranları tanımlayan bir dizi Screen nesnesi sağlar. Bu yönteme aktarılan dikdörtgen, sanal masaüstündeki piksel koordinatlarındadır. Hiçbir ekran dikdörtgenle kesişmiyorsa, dizi boştur. Bu yöntemi, bir pencerenin hangi ekranlarda görüntülendiğini bulmak için kullanabilirsiniz.

Screen class yöntemleri ve özelliklerinden dönen değerleri kaydetmeyin. Kullanıcı veya işletim sistemi var olan ekranları ve bu ekranların yerleştirilmesini her zaman değiştirebilir.

Aşağıdaki örnek, ok tuşlarına basılmasına yanıt olarak bir pencereyi birden çok ekran arasında taşımak için ekran API'sini kullanır. Pencereyi sonraki ekrana taşımak için örnek `screens` dizisini getirir ve bu diziyi dikey veya yatay olarak sıralar (basılan ok tuşuna göre). Daha sonra kod sıralanan dizide her ekranı geçerli ekranın koordinatlarıyla karşılaştırarak ilerler. Örnek, pencerenin geçerli ekranını tanımlamak için pencere sınırlarında iletilen `Screen.getScreensForRectangle()` yöntemini çağırır.

```
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;
    }
  }
}
```

Bölüm 54: Yazdırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash çalışma zamanları (Adobe® Flash® Player ve Adobe® AIR™ gibi), sayfaları yazıcı kuyruğuna iletebilmeniz için bir işletim sisteminin yazdırma arabirimiyle iletişim kurabilir. Kuyruğa gönderilen her sayfa, veritabanı değerleri ve dinamik metinler de dahil olmak üzere kullanıcı için görülebilir, dinamik veya ekran dışı olan içerikleri barındırabilir. Ayrıca, flash.printing.PrintJob sınıfının özellikleri, kullanıcının yazıcı ayarlarını esas alan değerler içerir, böylece sayfaları uygun şekilde sayfaları biçimlendirebilirsiniz.

Sonraki içerik, bir yazdırma işi oluşturmak, kullanıcının yazdırma ayarlarını okumak ve Flash çalışma zamanından ve kullanıcının işletim sisteminden gelen geribildirimler doğrultusunda yazdırma işi üzerinde ayarlamalar yapmak için flash.printing.PrintJob sınıf yöntem ve özelliklerinin kullanılmasına yönelik stratejiler sağlar.

Yazdırma temelleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0'da, çıktıda mürekkep ve kağıt temsiline dönüştürmek üzere görüntüleme içeriğinin anlık görüntülerini oluşturmak için PrintJob sınıfını kullanırsınız. İçeriğin yazdırılmak üzere ayarlanması ile ekranda görüntülenmek üzere ayarlanması bazı yönlerden birbirine benzer—her iki durumda da istediğiniz mizanpajı oluşturmak için öğeleri konumlandırıp boyutlandırırsınız. Ancak yazdırma, ekran mizanpajından farklı olmasını sağlayan bazı özelliklere sahiptir. Örneğin, yazıcılar bilgisayar monitörlerinden farklı bir çözünürlük kullanır; bilgisayar ekranının içerikleri dinamik olup değişebilirken yazdırılan içerik doğası gereği statiktir ve yazdırma planlanırken, sabit sayfa boyutu sınırlamaları ve çoklu sayfa yazdırma olasılığı da göz önünde bulundurulmalıdır.

Bu farklar netmiş gibi görünse de, ActionScript ile yazdırma ayarlaması yapılırken bunların dikkate alınması gerekir. Doğru bir yazdırma işleminin temelinde, sizin ve kullanıcı yazıcısının özellikleri tarafından belirtilen değerlerden oluşan bir kombinasyon bulunur. PrintJob sınıfının, kullanıcı yazıcısının önemli özelliklerini belirlemenizi sağlayan özellikleri vardır.

Önemli kavramlar ve terimler

Aşağıdaki başvuru listesi yazdırmaya ilişkin önemli terimler içerir:

Kuyruk İşletim sisteminin veya yazıcı sürücüsü yazılımının, yazdırılmayı bekleyen sayfaların takibini yapan ve uygun olduğunda bunları yazıcıya gönderen bir bölümü.

Sayfa yönlendirme Yazdırılan içeriğin yatay veya dikey olacak şekilde kağıda göre döndürülmesi.

Yazdırma işi Tek bir çıktıyı oluşturan sayfa veya sayfalar kümesi.

Sayfa yazdırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yazdırmayı işlemek için PrintJob sınıfının bir örneğini kullanırsınız. Flash Player veya AIR uygulaması üzerinden temel bir sayfa yazdırmak için sırayla şu dört deyimi kullanırsınız:

- `new PrintJob()`: Belirttiğiniz adda yeni bir yazdırma işi örneği oluşturur.
- `PrintJob.start()`: İşletim sistemi için yazdırma işlemini başlatarak kullanıcı için yazdırma iletişim kutusunu çağırır ve yazdırma işinin salt okunur özelliklerini doldurur.
- `PrintJob.addPage()`: Sprite nesnesi (ve içerdiği alt öğeler), yazdırma alanının boyutu ve yazıcının görüntüyü vektör olarak mı yoksa bitmap olarak mı yazdırması gerektiği gibi, yazdırma işi içerikleriyle ilgili ayrıntıları içerir. Birçok sayfa üzerine çoklu hareketli grafik yazdırmak için `addPage()` öğesine art arda çağrılar yapabilirsiniz.
- `PrintJob.send()`: Sayfaları işletim sisteminin yazıcısına gönderir.

Bu nedenle, örneğin basit bir yazdırma görevi komut dosyası (bir araya getirme için `package`, `import` ve `class` ifadelerini içeren):

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

Not: *Bu örnek, yazdırma işi komut dosyasının temel öğelerini göstermek üzere tasarlanmış olup herhangi bir hata işleme içermez. Yazdırma işini iptal eden bir kullanıcıya düzgün şekilde yanıt veren bir komut dosyası oluşturmak için, bkz.* *"İstisnalarla ve döndürmelerle çalışma" sayfa 935.*

Herhangi bir nedenden dolayı PrintJob nesnesinin özelliklerini temizlemek için PrintJob değişkenini `null` değerine ayarlayın (örn. `myPrintJob = null`).

Flash çalışma zamanı görevleri ve sistem yazdırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash çalışma zamanları, işletim sisteminin yazdırma arabirimine sayfalar gönderdiğinden, Flash çalışma zamanları tarafından yönetilen görevlerin ve işletim sisteminin kendi yazdırma arabirimi tarafından yönetilen görevlerin kapsamına dikkat edin. Flash çalışma zamanları bir yazdırma işini başlatabilir, yazıcının sayfa ayarlarından bazılarını okuyabilir, bir yazdırma işinin içeriğini işletim sistemine iletebilir ve kullanıcının veya sistemin bir yazdırma işini iptal edip etmediğini doğrulayabilir. Yazıcıya özgü iletişim kutularını görüntüleme, kuyruğa alınan bir yazdırma işini iptal etme veya yazıcının durumu hakkında bildirim gönderme gibi diğer işlemler, işletim sistemi tarafından işlenir. Flash çalışma zamanları, bir yazdırma işinin başlatılmasıyla veya biçimlendirilmesiyle ilgili bir sorun olduğunda yanıt verebilir ancak işletim sisteminin yazdırma arabiriminden yalnızca belirli özellikler ya da koşullar hakkında bildirim gönderebilir. Bir geliştirici olarak kodunuzun bu özellikleri veya koşulları yanıtlaması gerekir.

İstisnalarla ve döndürmelerle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcının yazdırma işini iptal etmiş olma ihtimaline karşı, `addPage()` ve `send()` çağrılarını çalıştırmadan önce, `PrintJob.start()` yönteminin `true` değerini döndürüp döndürmediğini kontrol edin. Devam etmeden önce bu yöntemlerin iptal edilmiş olup olmadığını kontrol ermenin basit bir yolu, aşağıdaki gibi, bu yöntemlerin `if` ifadesinde sarılmasıdır:

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

`PrintJob.start()` öğesi `true` ise, kullanıcı Print'i seçer (veya Flash Player ya da AIR gibi bir Flash çalışma zamanı, Yazdır komutunu başlatabilir). Bu yüzden, `addPage()` ve`send()` yöntemleri çağrılır.

Ayrıca Flash çalışma zamanları yazdırma işleminin yönetilmesine yardımcı olmak üzere `PrintJob.addPage()` yöntemi için istisnalar atar, böylece hataları yakalayıp kullanıcıya bilgi ve seçenekler sağlayabilirsiniz. `PrintJob.addPage()` yöntemi başarısız olursa, başka bir işlevi çağırabilir veya geçerli yazdırma işini durdurabilirsiniz. Bu istisnaları yakalamak için, aşağıdaki örnekte görüldüğü gibi, `addPage()` çağrılarını bir `try..catch` deyimine gömersiniz. Örnekteki `[params]` öğesi, yazdırmak istediğiniz gerçek içeriği belirten parametrelerin yer tutucusudur:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

Yazdırma işi başlatıldıktan sonra, `PrintJob.addPage()` öğesini kullanarak içerik ekleyebilir ve bunun bir istisna oluşturup oluşturmadığını (örneğin, kullanıcının yazdırma işini iptal edip etmediğini) görebilirsiniz. İstisna oluşuyorsa, kullanıcıya (veya Flash çalışma zamanına) bilgi ve seçenekler sağlamak için `catch` deyimine mantık ekleyebilir veya geçerli yazdırma işini durdurabilirsiniz. Sayfayı başarıyla eklerseniz `PrintJob.send()` öğesini kullanarak sayfaları yazıcıya gönderme işlemine devam edebilirsiniz.

Flash Player veya AIR uygulaması, yazdırma işinin yazıcıya gönderilmesiyle ilgili bir sorunla karşılaşırsa (örneğin, yazıcı çevrimdışı olduğunda), bu istisnayı da yakalayabilir ve bilgi veya daha fazla seçenek sağlayabilirsiniz (örneğin bir mesaj metni görüntüleyerek veya animasyonun içine bir uyarı yerleştirerek). Örneğin, aşağıdaki kodun gösterdiği gibi, bir `if..else` deyimindeki metin alanına yeni metin atayabilirsiniz:

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

Çalışan bir örnek için bkz. “Yazdırma örneği: Ölçekleme, kırpma ve yanıtlama” sayfa 944.

Sayfa özellikleriyle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcı Yazdır iletişim kutusunda Tamam'ı tıklatıp `PrintJob.start()` öğesi `true` değerini döndürdükten sonra, yazıcının ayarları tarafından tanımlanan özelliklere erişebilirsiniz. Bu ayarlar arasında kağıt genişliği, kağıt yüksekliği (`pageHeight` ve `pageWidth`) ve kağıtta içeriğin yönlendirmesi yer alır. Bunlar Flash çalışma zamanı tarafından denetlenmeyen yazıcı ayarları olduğundan, bu ayarları değiştiremezsiniz ancak yazıcıya gönderdiğiniz içeriği geçerli ayarlarla eşleşecek şekilde hizalamak için bunları kullanabilirsiniz. Daha fazla bilgi için, bkz. “Boyutu, ölçeği ve yönlendirmeyi ayarlama” sayfa 937.

Vektör veya bitmap oluşturmayı ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Her sayfayı vektör grafiği veya bitmap görüntüsü olarak kuyruğa almak için yazdırma işini elle ayarlayabilirsiniz. Bazı durumlarda vektör yazdırma, bitmap yazdırmaya göre daha küçük bir kuyruk dosyası ve daha iyi bir görüntü oluşturacaktır. Ancak içeriğiniz bir bitmap görüntüsü barındırıyorsa ve siz de alfa saydamlığı veya renk efektlerini korumak istiyorsanız, sayfayı bitmap görüntüsü olarak yazdırın. Ayrıca, PostScript olmayan bir yazıcı herhangi bir vektör grafiğini bitmap görüntüsüne dönüştürür.

PrintJobOptions nesnesini `PrintJob.addPage()` öğesinin üçüncü parametresi olarak geçirerek bitmap yazdırmayı belirtirsiniz.

Flash Player ve AIR 2'den önce AIR için, PrintJobOptions nesnesinin `printAsBitmap` parametre setini aşağıdaki gibi `true` olarak ayarlayın:

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

Üçüncü parametre için bir değer belirtmezseniz, yazdırma işi varsayılan olarak vektör yazdırmayı kullanacaktır.

AIR 2 ve üstü için, PrintJobOptions nesnesinin `printMethod` özelliğini kullanarak yazdırma yöntemini belirtin. Bu özellik, PrintMethod sınıfında sabit olarak tanımlanan üç değer kabul eder:

- `PrintMethod.AUTO`: Yazdırılmakta olan içeriğe göre en iyi yazdırma yöntemini otomatik olarak seçer. Örneğin, bir sayfada metin bulunuyorsa, vektör yazdırma yöntemi seçilir. Ancak, metnin üzerine alfa saydamlığına sahip bir filigran görüntüsü bindirilirse, saydamlığı korumak için bitmap yazdırma seçilir.
- `PrintMethod.BITMAP`: İçeriğe bakılmaksızın bitmap yazdırmaya zorlar
- `PrintMethod.BITMAP`: İçeriğe bakılmaksızın vektör yazdırmaya zorlar

Yazdırma işi deyimlerini zamanlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0, bir PrintJob nesnesini tek bir kareyle kısıtlamaz. (Önceki ActionScript sürümlerinde bu kısıtlama vardı.) Ancak, kullanıcı Yazdır iletişim kutusunda Tamam düğmesini tıklattıktan sonra işletim sistemi yazdırma durumu bilgilerini kullanıcıya görüntülediğinden, sayfaları kuyruğa göndermek için olabildiğince çabuk şekilde `PrintJob.addPage()` ve `PrintJob.send()` öğelerini çağırın. `PrintJob.send()` çağrısını içeren kareye ulaşan bir gecikme yazdırma işlemini geciktirir.

ActionScript 3.0'da, 15 saniyelik komut dosyası zaman aşımı sınırı vardır. Bu nedenle, bir yazdırma işi sırasındaki her önemli deyim arasındaki süre 15 saniyeden uzun olamaz. Başka bir deyişle, 15 saniyelik zaman aşımı sınıfı şu aralıklar için geçerlidir:

- `PrintJob.start()` ile birinci `PrintJob.addPage()` arasında
- `PrintJob.addPage()` ile bir sonraki `PrintJob.addPage()` arasında
- Son `PrintJob.addPage()` ile `PrintJob.send()` arasında

Bu aralıklardan herhangi biri 15 saniyeden uzun sürerse, PrintJob örneğinde `PrintJob.start()` öğesine yapılan bir sonraki çağrı `false` değerini döndürür ve PrintJob örneğindeki bir sonraki `PrintJob.addPage()` öğesi, Flash Player veya AIR uygulamasının bir çalışma zamanı istisnası atmasına neden olur.

Boyutu, ölçeği ve yönlendirmeyi ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

"Sayfa yazdırma" sayfa 934 bölümünde, temel yazdırma işinin adımları ayrıntılı şekilde ele alınmaktadır ve burada çıktı sözcüğü, belirtilen hareketli grafiğin ekran boyutunun ve konumunun yazdırılmış eşdeğerini yansıtır. Ne var ki yazıcılar yazdırma için farklı çözünürlükler kullanır ve yazdırılan hareketli grafiğin görünümünü olumsuz yönde etkileyen ayarlara sahip olabilir.

Flash çalışma zamanları bir işletim sisteminin yazdırma ayarlarını okuyabilir ancak bu özelliklerin salt okunur olduğunu ve özelliklerin değerlerini yanıtlayabilseniz de ayarlayamadığınızı unutmayın. Bu nedenle de, örneğin, yazıcının sayfa boyutu ayarlarını öğrenip içeriğinizi bu boyuta sığacak şekilde ayarlayabilirsiniz. Ayrıca yazıcının kenar boşluğu ayarlarını ve sayfa yönlendirmesini de belirleyebilirsiniz. Yazıcı ayarlarını yanıtlamak için, bir yazdırma alanı belirtin, ekran çözünürlüğü ile yazıcının nokta ölçüleri arasındaki farkı ayarlayın veya kullanıcının yazıcısının boyut ve yönlendirme ayarlarına uyacak şekilde içeriğinizi dönüştürün.

Yazdırma alanı için dikdörtgenler kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`PrintJob.addPage()` yöntemi, hareketli grafiğin yazdırılmasını istediğini bölgesini belirtmenize olanak sağlar. İkinci parametre olan `printArea`, bir Rectangle nesnesi şeklindedir. Bu parametre için bir değer sağlamaya yönelik üç seçeneğiniz vardır:

- Aşağıdaki örnekte olduğu gibi, belirli özelliklere sahip bir Rectangle nesnesi oluşturun ve sonra `addPage()` çağrısında bu dikdörtgeni kullanın:

  ```
  private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
  myPrintJob.addPage(sheet, rect1);
  ```

- Önceden bir Rectangle nesnesi belirtmediyseniz, aşağıdaki örnekte olduğu gibi, bunu çağrının içinde yapabilirsiniz:

  ```
  myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
  ```

- `addPage()` çağrısında üçüncü parametre için değer sağlamayı planlıyorsanız ancak bir dikdörtgen belirtmek istemiyorsanız, aşağıdaki gibi, ikinci parametre için `null` değerini kullanabilirsiniz;

  ```
  myPrintJob.addPage(sheet, null, options);
  ```

Nokta ve pikselleri karşılaştırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir dikdörtgenin genişliği ve yüksekliği piksel değerlerindedir. Bir yazıcı, yazdırma ölçüm birimi olarak puntoları kullanır. Puntolar sabit bir fiziksel boyuttadır (1/72 inç) ancak ekrandaki pikselin boyutu o ekranın çözünürlüğüne bağlıdır. Pikseller ile puntolar arasındaki dönüştürme oranı, yazıcı ayarlarına ve hareketli grafiğin ölçeklenmiş olup olmamasına bağlıdır. 72 piksel genişliğindeki ölçeklenmemiş bir hareketli grafik, ekran çözünürlüğünden bağımsız olarak, bir nokta bir piksele eşit olacak şekilde bir inç genişliğinde yazdırılır.

İnç ve santimetre birimlerini twip'lere veya puntolara (bir twip, bir puntonun 1/20'sidir) dönüştürmek için aşağıdaki eşitlikleri kullanabilirsiniz:

- 1 punto = 1/72 inç = 20 twip
- 1 inç = 72 punto = 1440 twip
- 1 santimetre = 567 twip

`printArea` parametresini çıkarırsanız veya bu parametre yanlış şekilde iletilirse, hareketli grafiğin tüm alanı yazdırılır.

Ölçekleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir Sprite nesnesini yazdırmadan önce ölçeklemek istiyorsanız, `PrintJob.addPage()` yöntemini çağırmadan önce ölçek özelliklerini ayarlayın (bkz. "Boyut ve ölçekleme nesnelerini işleme" sayfa 182) ve yazdırma işleminden sonra bunları geri orijinal değerlerine ayarlayın. Sprite nesnesinin ölçeği ile `printArea` özelliği arasında herhangi bir ilişki yoktur. Başka bir deyişle, 50 piksel x 50 piksel olan bir yazdırma alanı belirtirseniz, 2500 piksel yazdırılır. Sprite nesnesini ölçeklerseniz, aynı 2500 piksellik alan yazdırılır, ancak Sprite nesnesi ölçeklenen boyutta yazdırılır.

Bir örnek için bkz. "Yazdırma örneği: Ölçekleme, kırpma ve yanıtlama" sayfa 944.

Yatay ve dikey yönlendirme için yazdırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve AIR uygulaması yönlendirme ayarlarını algılayabildiğinden, aşağıdaki örnekte gösterildiği gibi, yazıcı ayarlarına yanıt olarak içerik boyutunu veya dönüşü ayarlamak için ActionScript uygulamanızda mantık oluşturabilirsiniz:

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

Not: *Kağıtta içerik yönlendirmesine yönelik sistem ayarını okumayı planlıyorsanız, PrintJobOrientation sınıfını içe aktarmayı unutmayın. PrintJobOrientation sınıfı, sayfadaki içerik yönlendirmesini tanımlayan sabit değerler sağlar. Şu deyimi kullanarak sınıfı içe aktarırsınız:*

```
import flash.printing.PrintJobOrientation;
```

Sayfa yüksekliğini ve genişliğini yanıtlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yazıcı yönlendirme ayarlarına benzer bir strateji kullanarak, sayfa yüksekliği ve genişliği ayarlarını okuyabilir ve `if` deyimine bir mantık gömerek bunları yanıtlayabilirsiniz. Aşağıdaki kod bir örneği gösterir:

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

Üstelik, aşağıdaki örnekte gösterdiği gibi, sayfa ve kağıt boyutları karşılaştırılarak sayfanın kenar boşluğu ayarları belirlenebilir:

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

Gelişmiş yazdırma teknikleri

Adobe AIR 2 ve üstü

Adobe AIR 2'den itibaren, PrintJob sınıfı ek özellik ve yöntemlere sahiptir ve üç ilave sınıf desteklenir: PrintUIOptions, PaperSize ve PrintMethod. Bu değişiklikler, ilave yazıcı iş akışlarına izin verir ve yazarların yazdırma işlemi üzerinde daha fazla kontrol sahibi olmasını sağlar. Değişiklikler şunları içerir:

- Sayfa yapısı iletişim kutuları: Hem standart hem de özel sayfa yapısı iletişim kutuları görüntülenebilir. Kullanıcı, yazdırma işlemine başlamadan önce sayfa aralıklarını, kağıt boyutunu, yönlendirmeyi ve ölçeklemeyi ayarlayabilir.
- Yazdırma görünümü: Kağıt boyutunu, kenar boşluklarını ve içeriğin sayfadaki konumunu tam olarak gösteren bir görünüm modu oluşturulabilir.
- Sınırlı yazdırma: Yazarlar, yazdırılabilir sayfaların aralığı gibi yazdırma seçeneklerini sınırlandırabilir.
- Kalite seçenekleri: Yazarlar bir belgenin yazdırma kalitesini ayarlayabilir ve kullanıcının çözünürlüğü ve renk seçeneklerini belirlemesine izin verebilir.
- Birden fazla yazdırma oturumu: Tek bir PrintJob örneği artık birden fazla yazdırma oturumu için kullanılabilir. Uygulamalar sayfa yapısı ve yazdırma iletişim kutuları her görüntülendiğinde tutarlı ayarlar sağlayabilir.

Yazdırma iş akışı değişiklikleri

Yeni yazdırma iş akışı şu adımlardan oluşur:

- `new PrintJob()`: Bir PrintJob örneği oluşturur (veya varolan bir örneği yeniden kullanır). `selectPaperSize()` gibi birçok yeni PrintJob özelliği ve yöntemi, yazdırma işi başlamadan veya yazdırma sırasında kullanılabilir durumdadır.
- `PrintJob.showPageSetupDialog()`: (isteğe bağlı) bir yazdırma işlemi başlatmadan sayfa yapısı iletişim kutusunu görüntüler.
- `PrintJob.start()` veya `PrintJob.start2()`: `start()` yöntemine ek olarak, `start2()` yöntemi yazdırma kuyruğu sürecini başlatmak için kullanılır. `start2()` yöntemi, Yazdır iletişim kutusunu gösterip göstermemeyi ve gösterilirse iletişim kutusunu özelleştirip özelleştirmemeyi seçmenize imkan verir.
- `PrintJob.addPage()`: Yazdırma işine içerik ekler. Önceki işleme göre değişmemiştir.
- `PrintJob.send()` veya `PrintJob.terminate()`: Sayfaları seçilen yazıcıya gönderir veya yazdırma işini göndermeden sonlandırır. Yazdırma işleri bir hataya yanıt olarak sonlandırılır. Bir PrintJob örneği sonlandırılırsa, hala yeniden kullanılabilir durumdadır. Yazdırma işinin yazıcıya gönderilmesine veya sonlandırılmasına bakılmaksızın, PrintJob örneğini yeniden kullandığınızda geçerli yazdırma ayarları korunur.

Sayfa yapısı iletişim kutusu

Geçerli ortam destekliyorsa, `showPageSetupDialog()` yöntemi işletim sisteminin Sayfa Yapısı iletişim kutusunu görüntüler. Bu yöntemi çağırmadan önce her zaman `supportsPageSetupDialog` özelliğini kontrol edin. Aşağıda basit bir örnek bulunmaktadır:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

Sayfa Kurulumu iletişim kutusunda hangi seçeneklerin görüntülendiğini kontrol etmek için bu yöntem isteğe bağlı olarak bir PrintUIOptions sınıfı özelliği ile çağrılabilir. En az ve en fazla sayfa sayısı ayarlanabilir. Aşağıdaki örnek yazdırmayı ilk üç sayfaya sınırlar:

```
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);
}
```

Yazdırma ayarlarını değiştirme

PrintJob örneğinin ayarları, oluşturulduktan sonra herhangi bir zamanda değiştirilebilir. Buna bir yazma işi gönderildikten veya durdurulduktan sonra `addPage()` aramaları arasındaki ayarları değiştirme de dahildir. `printer` özelliği gibi bazı ayarlar, tek tek sayfalar için değil, tüm yazdırma işi için geçerli olur. Bu ayarlar `start()` veya `start2()` çağrılmadan önce yapılmalıdır.

`selectPaperSize()` yöntemi Sayfa Yapısı ve Yazdırma iletişim kutularında varsayılan kağıt boyutunu ayarlamak için çağrılabilir. Ayrıca bir yazdırma işi sırasında sayfa aralığına yönelik kağıt boyutunu ayarlamak için de kullanılabilir. 10 numara zarf boyutunun seçildiği örnekteki gibi, `PaperSize` sınıfında tanımlanan sabitleri kullanarak çağrılır:

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

Geçerli yazıcı işinin yazıcı adını almak veya ayarlamak için `printer` özelliğini kullanın. Varsayılan olarak, varsayılan yazıcının adına ayarlanır. Kullanılabilir yazıcı yoksa veya sistem yazdırmayı desteklemiyorsa `printer` özelliği `null` olur. Yazıcıyı değiştirmek için ilk olarak `printers` özelliğini kullanarak kullanılabilir yazıcıların listesini alın. Bu özellik, Dize öğeleri kullanılabilir yazıcı adları olan bir Vektördür. Yazıcıyı aktif hale getirmek için, `printer` özelliğini bu Dize değerlerinden birine ayarlayın. Etkin bir yazıcı işinin `printer` özelliği değiştirilemez. `start()` veya `start2()` öğesine yapılan başarılı bir çağrıdan sonra veya görev gönderilmeden veya başarısızlıkla sonlanmadan önce bunu değiştirmeye çalışır. Bu özelliği ayarlamaya ilişkin bir örnek:

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

`copies` özelliği, işletim sisteminin Yazdırma iletişim kutusunda ayarlanan kopya sayısının değerini alır. `firstPage` ve `lastPage` özellikleri sayfa aralığını alır. `orientation` özelliği kağıt yönlendirme ayarını alır. Bu özellikler Yazdır iletişim kutusundaki değerleri geçersiz kılacak şekilde ayarlanabilir. Aşağıdaki örnek bu özellikleri ayarlar:

```
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;
```

`PrintJob` öğesiyle ilişkili aşağıdaki salt okunur ayarlar geçerli yazıcı kurulumuyla ilgili faydalı bilgiler sağlar:

- `paperArea`: Yazıcı aracının nokta cinsinden dikdörtgen sınırları.
- `printableArea`: Yazdırılabilir alanın nokta cinsinden dikdörtgen sınırları.
- `maxPixelsPerInch`: Geçerli yazıcının inç başına piksel cinsinden fiziksel çözünürlüğü.
- `isColor`: Geçerli yazıcının renk yazdırma özelliği (geçerli yazıcı renk yazdırabiliyorsa `true` değerini döndürür).

Bkz. "Yazdırma örneği: Sayfa düzeni ve yazdırma seçenekleri" sayfa 946.

Yazdırma örneği: Çoklu sayfa yazdırma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Birden çok sayfa içeriği yazdırırken, her içerik sayfasını farklı bir hareketli grafikle ilişkilendirebilir (bu durumda, `sheet1` ve `sheet2`) ve daha sonra her hareketli grafik için `PrintJob.addPage()` öğesini kullanabilirsiniz. Aşağıdaki kod bu tekniği gösterir:

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

Yazdırma örneği: Ölçekleme, kırpma ve yanıtlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bazı durumlarda, görüntüleme nesnesinin ekranda görüntülenme şekli ile kağıda yazdırıldığında görüntülenme şekli arasındaki farkları birbirine uygun hale getirmek için, görüntüleme nesnesini yazdırırken nesnenin boyutunu (veya diğer özelliklerini) ayarlamak isteyebilirsiniz. Yazdırma işleminden önce görüntüleme nesnesinin özelliklerini ayarlarken (örneğin, `scaleX` ve `scaleY` özelliklerini kullanarak), nesne, yazdırma alanı için tanımlanmış dikdörtgenden daha büyük olacak şekilde ölçeklenirse, nesnenin kırpılacağını unutmayın. Sayfalar yazdırıldıktan sonra özellikleri sıfırlamak isteyebilirsiniz.

Aşağıdaki kod, `txt` görüntüleme nesnesinin (yeşil arka plan kutusunun değil) boyutlarını ölçekler ve metin alanı, belirtilen dikdörtgenin boyutlarına göre kırpılır. Yazdırma işleminden sonra, metin alanı ekranda görüntülendiği orijinal boyutuna geri döner. Kullanıcı, işletim sisteminin Yazdır iletişim kutusundan yazdırma işini iptal ederse, Flash çalışma zamanındaki içerik, işin iptal edildiğini kullanıcıya bildirmek için değişir.

```
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;
        }
    }
}
```

Yazdırma örneği: Sayfa düzeni ve yazdırma seçenekleri

Adobe AIR 2 ve üstü

Aşağıdaki örnek; kopya sayısı, kağıt boyutu (legal) ve sayfa yönü (yatay) için PrintJob ayarlarını başlatır. Sayfa Yapısı iletişim kutusunu görüntülenecek ilk öğe olmaya zorlar ve sonrasında Yazdırma iletişim kutusunu görüntüleyerek yazdırma işini başlatır.

```
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;
        }
    }
}
```

Bölüm 55: Geolocation

Bir aygıt coğrafi konumu destekliyorsa, coğrafi konum API'sini aygıtın geçerli coğrafi konumunu almak için kullanabilirsiniz. Aygıt bu özelliği destekliyorsa, coğrafi konum bilgilerini alabilirsiniz. Bu bilgilere yükseklik, doğruluk, başlık, hız ve o konumdaki son değişikliğin zaman damgası dahildir.

Coğrafi konum sınıfı aygıtın konum alıcısına yanıt olarak `update` olayları gönderir. `update` olayı bir GeolocationEvent nesnesidir.

Daha fazla Yardım konusu

flash.sensors.Geolocation

flash.events.GeolocationEvent

Coğrafi konum değişikliklerini saptama

Bir coğrafi konum sensörü kullanmak için, Coğrafi konum nesnesini somutlaştırın ve gönderdiği `update` olaylarına kaydolun. `update` olayı bir Coğrafi Konum olayı nesnesidir. Olayın sekiz özelliği bulunur:

- `altitude`—Metre cinsinden yükseklik.
- `heading`—Tamsayı olarak derece cinsinden hareketin yönü (kuzeye göre).
- `horizontalAccuracy`—Metre cinsinden yatay doğruluk.
- `latitude`—Derece cinsinden enlem.
- `longitude`—Derece cinsinden boylam.
- `speed`—Saniye başına metre cinsinden hız.
- `timestamp`—Çalışma zamanının başlatılmasından olay zamanına kadar geçen milisaniye sayısı.
- `verticalAccuracy`—Metre cinsinden dikey doğruluk.

`timestamp` özelliği dahili bir nesnedir. Diğerleri ise Sayı nesneleridir.

Aşağıda bir metin alanında coğrafi konum verilerini gösteren basit bir örnek verilmektedir:

```
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()
}
```

Bu örneği kullanmak için, `geoTextField` metin alanını oluşturup bu kodu kullanmadan önce ekran listesine eklemeyi unutmayın.

Coğrafi konum nesnesinin `setRequestedUpdateInterval()` yöntemini çağırarak coğrafi konum için istenilen zaman aralığını ayarlayabilirsiniz. Bu yöntemin çağırdığı tek parametre, milisaniye cinsinden talep edilen güncelleme aralığı olan `interval` parametresidir:

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

Coğrafi konum güncellemeleri arasındaki asıl süre bu değerden daha büyük veya daha az olabilir. Güncelleme aralığındaki herhangi bir değişiklik tüm kayıtlı dinleyicileri etkiler. `setRequestedUpdateInterval()` yöntemini çağırmazsanız, uygulama aygıtın varsayılan aralığına göre güncelleme alır.

Kullanıcı, bir uygulamanın coğrafi konum verilerine erişmesini önleyebilir. Örneğin, bir uygulama coğrafi konum verilerini almaya çalıştığında, iPhone kullanıcıyı uyarır. Uyarıya yanıt olarak, uygulamanın coğrafi konum verilerine erişimi kullanıcı tarafından engellenebilir. Coğrafi konum nesnesi, kullanıcı mevcut olmayan coğrafi konum verilerine eriştiği zaman bir `status` olayı gönderir. Ayrıca, Coğrafi Konum nesnesi, coğrafi konum sensörü mevcut olmadığında `true` olarak ayarlanan bir `muted` özelliğine sahiptir. `muted` özelliği değiştiğinde, Coğrafi Konum nesnesi bir `status` olayı gönderir. Aşağıdaki kod, coğrafi konum verilerinin mevcut olmadığı durumların nasıl tespit edileceğini göstermektedir:

```
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 = "";
        }
    }
}
```

Not: *GPS birimi içermeyen birinci nesil iPhone'lar, yalnızca zaman zaman* `update` *olayları gönderir. Bu cihazlarda ilk olarak bir Geolocation nesnesi bir veya iki* `update` *olayı dağıtır. Daha sonra bilgi önemli ölçüde değiştiğinde,* `update` *olaylarını dağıtır.*

Coğrafi konum desteğini kontrol etme

Çalışma zamanı ortamının bu özelliği kullanabilme becerisini sınamak için `Geolocation.isSupported` özelliğini kullanın:

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

Şu anda, coğrafi konum yalnızca iPhone için ActionScript tabanlı uygulamalarda ve Flash Lite 4'te desteklenmektedir. `Geolocation.isSupported` özelliği çalışma zamanında `true` ise, coğrafi konum desteği mevcuttur.

Bazı iPhone modellerinde bir GPS ünitesi bulunmaz. Bu modeller coğrafi konum bilgilerini almak için başka yöntemler kullanır (cep telefonu triangülasyonu gibi). Bu modellerde veya GPS özelliği devre dışı olan herhangi bir iPhone cihazında, bir Coğrafi Konum nesnesi başlangıçta yalnızca bir veya iki `update` olayı gönderir.

Bölüm 56: Uygulamaları uluslararasılaştırma

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

flash.globalization paketi farklı dillerin ve bölgelerin düzenine uyarlanan uluslararası yazılımların oluşturulmasını kolaylaştırır.

Daha fazla Yardım konusu

flash.globalization paketi

"Uygulamaları yerelleştirme" sayfa 969

Charles Bihis: Want to Localize your Flex/AIR Apps? (Flex/AIR Uygulamalarınızı Yerelleştirmek İster Misiniz?)

Uygulamaları uluslararasılaştırma temelleri

Küreselleştirme ve uluslararasılaştırma terimleri bazen birbirinin yerine kullanılabilir. Ancak bu terimlerin en yaygın bilinen tanımları, küreselleştirmenin iş ve mühendislik süreçlerinin bir birleşimini ifade ettiğini ve uluslararasılaştırmanın ise yalnızca mühendislikle ilgili olduğunu belirtir.

Aşağıda bazı önemli terimlerin tanımları vardır:

Küreselleştirme Ürünleri ve şirket etkinliklerini küresel olarak hazırlamak ve başlatmak için gereken geniş mühendislik ve iş süreçleri kapsamı. Küreselleştirme, uluslararasılaştırma, yerelleştirme ve kültüre uyarlama gibi mühendislik etkinliklerinden ve ürün yönetimi, mali planlama, pazarlama ve yasal çalışma gibi iş etkinliklerinden oluşur. Küreselleşme zaman zaman *G11n* (G harfi, ardından 11 harf daha ve ardından n harfi) olarak kısaltılır. *"Küreselleştirme işletmelerin işidir."*

Uluslararasılaştırma Yeniden tasarlamak ve yeniden derlemek zorunda kalmadan bir ürünün birden çok dili, komut dosyasını ve kültürel düzeni (para birimlerini, sıralama kurallarını, tarih biçimlerini ve daha fazlasını içerir) işleyebilmesini sağlayan bir mühendislik süreci. Bu süreç iki etkinlik kümesine bölünebilir: etkinleştirme ve yerelleştirme. Uluslararasılaştırma bazen *dünya çapında uygunluk* olarak ifade edilir ve bazen *I18n* olarak kısaltılır. *"Uluslararasılaştırma mühendislerin işidir."*

Yerelleştirme Bir ürünü veya hizmeti belirli bir dile, kültüre ve istenen yerel "görünüm ve deneyim" anlayışına uyarlama süreci. Yerelleştirme bazen *L10n* olarak kısaltılır. *"Yerelleştirme çevirmenlerin işidir."*

Kültüre uygun hale getirme Bir kültürün kendine özgü ihtiyaçları için belirli özellikler geliştiren veya uyarlayan bir mühendislik süreci. Örnekler Adobe InDesign uygulamasındaki Japonca yayınlama özelliklerini ve Adobe Acrobat uygulamasındaki Hanko desteğini içerir.

Bazı diğer uluslararasılaştırma terimleri aşağıdaki gibi tanımlanabilir:

Karakter Kümesi Bir dilde veya bir dil grubunda kullanılan karakterler. Bir karakter kümesi ulusal karakterleri, özel karakterleri (noktalama işaretleri ve matematik sembolleri gibi), sayısal rakamları ve bilgisayar denetim karakterlerini içerir.

Harmanlama Belirli bir yerel ayar için metni uygun bir sırada sıralama.

Yerel ayar Coğrafi, politik veya kültürel bir bölgede (bir çok durumda tek bir ülkeyi belirtir) kullanılan dili ve kültürel gelenekleri yansıtan bir değer. Benzersiz bir yerel ayar tanımlayıcısı (yerel ayar kimliği) bu değeri temsil eder. Yerel ayar kimliği yerel ayara özgü destek sağlayan yerel ayar verisi kümesinin aranması için kullanılır. Bu destek ölçme birimleri, numaraların ve tarihlerin ayrıştırılması ve biçimlendirilmesi vb. için geçerlidir.

Kaynak Paketi Bir uygulamanın içinde kullanılacağı yerel ayar için oluşturulmuş, depolanmış yerel ayara özgü öğeler. Kaynak paketi genellikle tüm metin öğelerini uygulamanın kullanıcı arabiriminde barındırır. Paket içinde, bu öğeler verilen yerel ayar için uygun dile çevrilir. Ayrıca belirli bir yerel ayara yönelik mizanpajı veya kullanıcı arabiriminin davranışını değiştiren diğer ayarları da içerebilir. Bir kaynak paketi, yerel ayara özgü olan diğer medya türlerini veya diğer medya türlerine yapılan başvuruları içerebilir.

flash.globalization paketine genel bakış

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

flash.globalization paketi temeldeki işletim sisteminin kültürel destek özelliklerini kullanır. Ayrı kullanıcıların kültürel düzenine uyan uygulamaları yazmayı kolaylaştırır.

Paketteki ana sınıflar şunları içerir:

- Dizelerin sıralanmasını ve eşleşmesini yöneten Collator sınıfı
- Sayıları para birimi miktarı dizeleri olarak biçimlendiren ve para birimi miktarlarını ve sembollerini girdi dizelerinden ayrıştıran CurrencyFormatter sınıfı
- Tarih değerlerini biçimlendiren DateTimeFormatter sınıfı
- Belli bir yerel hakkında bilgi almak için LocaleID sınıfı
- Sayısal değerleri biçimlendiren ve ayrıştıran NumberFormatter sınıfı
- Dizelerin yerel ayar duyarlı küçük büyük harf dönüşümünü işleyen StringTools sınıfı

flash.globalization paketi ve kaynak yerelleştirmesi

flash.globalization paketi kaynak yerelleştirmesi yapmaz. Ancak, flash.globalization yerel kimliğini başka teknikler kullanarak yerelleştirilmiş kaynakları almak için anahtar değer olarak kullanabilirsiniz. Örneğin, Flex ile oluşturulmuş uygulama kaynaklarını ResourceManager ve ResourceBundle sınıflarını kullanarak yerelleştirebilirsiniz. Daha fazla bilgi edinmek için bkz. Flex Uygulamalarını Yerelleştirme.

"AIR uygulamalarını yerelleştirme" sayfa 970 bölümünde ele alındığı gibi, Adobe AIR 1.1, AIR uygulamalarını yerelleştirmeye yardımcı olacak bazı özelliklere de sahiptir.

Bir uygulamayı uluslararasılaştırmaya genel yaklaşım

Aşağıdaki adımlarda bir uygulamayı flash.globalization paketi kullanarak uluslararasılaştırmaya yüksek düzeyde genel olan bir yaklaşım sunulmaktadır:

1. Yerel ayarı belirleyin veya yapın.
2. Bir hizmet sınıfı örneği oluşturun (Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter veya StringTools).
3. lastOperationStatus özelliklerini kullanarak hata ve çözümleri kontrol edin.
4. Yerel ayara özel ayarları kullanarak bilgileri biçimlendirip gösterin.

Bir sonraki adım ise yerel ayara özel dizeleri ve kullanıcı arabirimi kaynaklarını yükleyip görüntülemektir. Bu adımda şu görevler bulunabilir:

- Kullanıcı arabirimini dize uzunluklarına uygun olacak şekilde yeniden boyutlandırmak için otomatik düzen özelliklerini kullanma
- Doğru fontları seçme ve fontları destekleme
- Diğer yazma sistemlerini desteklemek için FTE metin motorunu kullanma
- Giriş yöntemi düzenleyicilerinin doğru kullanılmasını sağlama

Hata ve çözümleri kontrol etme

flash.globalization hizmeti sınıflarının tümü hata tanımlamak için benzer bir desen kullanır. Ayrıca istenilen ancak kullanılabilir durumda olmayan bir yerel ayardan, kullanıcının işletim sisteminin desteklediği bir yerel ayara geçiş desenleri de aynıdır.

Aşağıdaki örnek, hizmet sınıfları somutlaştırılırken hata ve çözümlerin nasıl kontrol edileceğini göstermektedir. Her hizmet sınıfının, en son çağırılan yöntemin hataları mı yoksa uyarıları mı tetiklediğini belirten bir lastOperationStatus özelliği vardır.

```
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);
    }
}
```

Bu örnek, bir çözüm yerel kimliğinin kullanıldığını veya bir hatanın olduğunu belirten mesaja dayanır. Uygulamanız, gerekirse ilave hata işleme mantığı gerçekleştirebilir. Örneğin, kullanıcıya bir mesaj gösterebilir veya uygulamayı özel, desteklenen bir yerel kullanmaya zorlayabilirsiniz.

Yerel ayarı belirleme

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Yerel ayar, bir ülke veya bölgeye özel dil ve kültür gelenekleri kombinasyonunu belirler.

Yerel ayar tanımlayıcısı bir dize olarak güvenli bir şekilde yönetilebilir. Bununla birlikte, LocaleID sınıfını bir yerel ayarla ilgili ek bilgiler elde etmek için kullanabilirsiniz.

Bir LocaleID aşağıdaki şekilde oluşturulur:

```
var locale:LocaleID = new LocaleID("es-MX");
```

LocaleID nesnesi oluşturulduktan sonra, yerel ayar kimliğiyle ilgili veri alabilirsiniz. `getKeysAndValues()`, `getLanguage()`, `getRegion()`, `getScript()`, `getVariant()` ve `isRightToLeft()` yöntemlerini ve `name` özelliğini kullanın.

Bu yöntem ve özelliklerden alınan değerler, yerel ayar hakkında doğrudan yerel ayar tanımlayıcısından elde edilemeyecek başka bilgileri yansıtabilir.

Bir uygulama, tarih biçimlendiricisi gibi yerele-duyarlı bir hizmet oluşturduğunda, kullanılması amaçlanan yerel ayarı belirtmek zorundadır. Desteklenen yerel ayarların listesi bir işletim sisteminden diğer işletim sistemine göre değişir; bu nedenle, istenen yerel ayar mevcut olmayabilir.

Flash Player ilk olarak talep ettiğiniz yerel ayarın dil koduyla eşleşmeye çalışır. Ardından, eşleşen bir yazma sistemi (komut dosyası) ve bölge bularak yerel ayarı geliştirir. Örneğin:

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

Bu örnekte `LocaleID()` yapıcısı o kullanıcının “es” dil koduyla en iyi eşleşen yerel ayar hakkında veri almıştır.

Yerel ayar kimliğini ayarlama

Bir uygulamanın geçerli yerel ayarını yapmanın birkaç yolu vardır:

- Uygulamaya doğrudan tek bir yerel ayar kimliği gömme. Bu yaygın bir yaklaşım olsa da, uygulamanın uluslararasılaştırılmasını desteklemez.
- Kullanıcının işletim sisteminde veya tarayıcısında bulunan yerel ayar kimliği tercihlerini ya da başka kullanıcı tercihlerini kullanın. Bu teknik genellikle kullanıcı için en iyi yerel ayarların alınmasını sağlasa da, her zaman doğru değildir. İşletim sistemi ayarlarının, kullanıcının gerçek tercihlerini yansıtmaması riski mevcuttur. Örneğin, kullanıcı ortak bir bilgisayar kullanıyor ve işletim sisteminin tercih edilen yerel ayarlarını değiştiremiyor olabilir.
- Yerel ayar kimliğini kullanıcının tercihlerine göre ayarladıktan sonra, kullanıcıya desteklenen yerel ayarlar listesinden seçim yapma imkanı verin. Uygulamanız birden fazla yerel ayarı destekleyebiliyorsa, normalde en iyi strateji budur.

Bu üçüncü seçeneği aşağıdaki gibi uygulayabilirsiniz:

1. Kullanıcı profilinden, tarayıcı ayarlarından, işletim sistemi ayarlarından veya bir çerezden kullanıcının yerel ayarlarının veya dillerinin bir listesini alın. (Uygulamanız bu mantığı kendisi uygulamalıdır. flash.globalization kütüphanesi bu tip tercihleri doğrudan okumayı desteklemez.)
2. Uygulamanızın bu yerel ayarlardan hangilerini desteklendiğini belirleyin ve varsayılan olarak en iyisini seçin. Bir kullanıcının tercih edilen yerel ayarlarına ve işletim sistemi tarafından desteklenen yerel ayarlara göre en iyi yerel ayarları bulmak için LocaleID.determinePreferredLocales() yöntemini kullanın.
3. Yerel ayar tatmin edici değilse, kullanıcıya varsayılan yerel ayarı değiştirme imkanı verin.

Diğer yerel ayar ve dil sınıflarının sınırlamaları

`fl.lang.Locale` sınıfı, metin dizelerini dize değerleri içeren kaynak yığınları kullanarak yerel ayara göre değiştirme imkanı verir. Ancak, bu sınıf sayı, para birimi veya tarih biçimlendirmesi, sıralama ve eşleştirme gibi diğer uluslararasılaştırma özelliklerini desteklemez. Bunun yanında, bu sınıf yalnızca Flash Professional ile birlikte kullanılabilir.

`flash.system.Capabilities.language` özelliğini kullanarak işletim sisteminin geçerli dil kodu ayarını da alabilirsiniz. Ancak, bu özellik yalnızca iki karakterli ISO 639-1 dil kodunu alır—tüm yerel ayar kimliğini değil—ve yalnızca belirli bir yerel ayar kümesini destekler.

AIR 1.5 ile `flash.system.Capabilities.languages` özelliğini kullanabilirsiniz. Bu özellik kullanıcının tercih edilen kullanıcı arabirimi dillerinin bir dizisini sağlar. Bu nedenle `Capabilities.language` sınırlamaları yoktur.

Rakamları biçimlendirme

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Sayısal değerlerin gösterilme biçimi, bölgeden bölgeye büyük farklılıklar gösterir. Örneğin, bazı yerel ayarlarda 123456.78 sayısı şu şekilde gösterilir:

Yerel ayar	Sayı Biçimi
en-US (İngilizce, ABD)	-123,456.78
de-DE (Almanca, Almanya)	-123.456,78
fr-FR (Fransa, Fransızca)	-123 456,78
de-CH (Almanca, İsviçre)	-123'456.78
en-IN (İngilizce, Hindistan)	-1,23,456.78
Çoğu Arapça yerel ayarları	123,456.78-

Sayı biçimlerini etkileyen çok sayıda faktör vardır:

- Ayırıcılar. Tamsayı ile sayının kesirli kısımları arasına ondalık ayırıcı konulur. Bu bir nokta, virgül veya başka bir karakter olabilir. Grup ayırıcısı veya binlik ayırıcısı bir nokta, virgül ya da bölünemez boşluk ya da başka bir karakter olabilir.
- Gruplandırma yöntemleri. Ondalık noktasının sol tarafındaki her bir grup ayırıcısı arasındaki basamak sayısı iki veya üç ya da başka bir değer olabilir.
- Negatif sayı göstergeleri Negatif sayılar, sayının solunda veya sağında bulunan bir eksi işaretiyle ya da mali uygulamalarda parantez içinde gösterilebilir. Örneğin, negatif 19, -19, 19- veya (19) olarak gösterilebilir.
- Virgülden önceki ve sonraki sıfırlar. Bazı kültürel geleneklerde gösterilen sayıların önüne veya ardına sıfır eklenir. Örneğin, 0,17 değeri diğer seçeneklerle birlikte ,17, 0,17 veya 0,170 olarak gösterilebilir.
- Sayı karakteri grupları. Hintçe, Arapça ve Japonca dahil birçok dilde farklı sayı karakteri grupları kullanılır. flash.globalization paketi 0-9 rakamlarıyla eşleşen tüm sayı karakteri gruplarını destekler.

NumberFormatter sınıfı sayısal değerleri biçimlendirirken tüm bu faktörleri göz önünde bulundurur.

NumberFormatter sınıfını kullanma

NumberFormatter sınıfı, sayısal değerleri (int, unit veya Number tipi) belli bir yerel ayarın kurallarına göre biçimlendirir.

Aşağıdaki örnekte, bir sayıyı kullanıcının işletim sistemi tarafından sağlanan varsayılan biçimlendirme özelliklerini kullanarak biçimlendirmenin en kolay yolu gösterilmektedir:

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

Sonuç, kullanıcının yerel ayarlarına ve kullanıcı tercihlerine göre farklılık gösterir. Örneğin, kullanıcının yerel ayarı fr-FR ise, biçimlendirilmiş değer şöyle olur:

-123.456,789

Bir sayıyı kullanıcının ayarlarına bağlı olmaksızın yalnızca belli bir yerel ayar için biçimlendirmek istiyorsanız, yerel ayar adını özel olarak ayarlayın. Örneğin:

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

Bu durumda sonuç şu şekildedir:

-123'456.789

formatNumber() yöntemi parametre olarak Number parametresini alır. NumberFormatter sınıfı girdi olarak int alan bir formatInt() yöntemine ve bir unit alan formatUint() yöntemine sahiptir.

NumberFormatter sınıfının özelliklerini bu örnekte gösterilen şekilde ayarlayarak, biçimlendirme mantığını açıkça kontrol edebilirsiniz:

```
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)
```

Bu örnek ilk olarak bir NumberFormatter nesnesi oluşturur ve ardından:

- negatif sayı biçimini parantez içinde (mali uygulamalardaki gibi) kullanmak üzere ayarlar;
- ondalık ayırıcıdan sonraki basamak sayısını 5 olarak ayarlar;
- beş ondalık yer elde etmek üzere virgülden sonra sıfır kullanılması gerektiğini belirtir;
- ondalık ayırıcıyı virgül olarak ayarlar;
- biçimlendiriciye herhangi bir grup ayırıcısı kullanmamasını söyler.

Not: *Bu özelliklerden bazıları değiştirildiğinde, elde edilen sayı biçimi artık belirtilen yerel ayarın tercih edilen biçimine uygun olmaz. Bu özelliklerden bazılarını yalnızca yerel ayarlar konusunda bilgi sahibi olmanız gerekmediğinde; takip eden sıfırların sayısı gibi bir biçimin tek bir açısı üzerinde ayrıntılı kontrole sahip olmanız gerektiğinde; veya kullanıcı değişimi doğrudan, örneğin Windows Denetim Masası üzerinden istediğinde kullanın.*

Sayısal değer içeren dizeleri ayrıştırma

NumberFormatter sınıfı, yerele özel biçimlendirme gereksinimlerine uyan dizelerden sayısal değerler de alabilir. NumberFormatter.parseNumber() yöntemi bir dizeden tek bir sayısal değer alır. Örneğin:

```
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
```

parseNumber() yöntemi yalnızca sayılar ve işaret ve ayırıcılar gibi sayı biçimlendirme karakterleri içeren dizeleri işler. Dize diğer karakterleri içeriyorsa, bir hata kodu ayarlanır:

```
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
```

Başka alfabetik karakterler içeren dizelerden sayı almak için, NumberFormatter.parse() yöntemini kullanın:

```
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
```

parse() yöntemi, value özelliğinde ayrıştırılmış sayısal değer bulunan bir NumberParseResult nesnesi gönderir. startIndex özelliği, bulunan ilk sayısal karakterin dizinini gösterir. Dizenin basamaktan önce ve sonra gelen kısımlarını almak için startIndex ve endIndex özelliklerini kullanabilirsiniz.

Para birimi değerlerini biçimlendirme

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Para birimi değerlerini gösterme biçimleri de sayı biçimleri kadar farklılık gösterir. Örneğin, bazı yerel ayarlarda $123456.78 değeri şu şekilde gösterilir:

Yerel ayar	Sayı Biçimi
en-US (İngilizce, ABD)	$123,456.78
de-DE (Almanca, Almanya)	123.456,78 $
en-IN (İngilizce, Hindistan)	$ 1,23,456.78

Para birimi biçimlendirme, sayı biçimlendirmedeki faktörlerin aynısını ve aşağıdaki diğer faktörleri içerir:

- Para birimi ISO kodu. Kullanılmakta olan yerel ayarın USD veya EUR gibi üç harfli ISO 4217 para birimi kodu.
- Para birimi sembolü. Kullanılan geçerli yerel ayarın $ veya € gibi para birimi sembolü veya dizesi.
- Negatif para birimi biçimi. Para birimi miktarının sayısal kısmına göre para birimi sembolünün ve negatif sembolünün veya parantezlerin konumunu tanımlar.
- Pozitif para birimi biçimi. Para birimi miktarının sayısal kısmına göre para birimi sembolünün konumunu tanımlar.

CurrencyFormatter sınıfını kullanma

CurrencyFormatter sınıfı, sayısal değerleri belli bir yerel ayarın kurallarına göre para birimi dizeleri ve biçimlendirilmiş sayılar içeren dizelere biçimlendirir.

Yeni bir CurrencyFormatter nesnesi somutlaştırdığınızda, para birimini geçerli yerel ayarın varsayılan para birimine ayarlar.

Aşağıdaki örnekte, Almanca yerel ayarları kullanılarak oluşturulmuş bir CurrencyFormatter nesnesinin, para birimi tutarlarını Euro cinsinden kabul ettiği gösterilmektedir:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Çoğu durumda, bir yerel ayarın varsayılan para birimine bağlı kalmamalısınız. Kullanıcının varsayılan yerel ayarı desteklenmiyorsa, CurrencyFormatter sınıfı yedek bir yerel ayar atar. Yedek yerel ayarın varsayılan para birimi farklı olabilir. Bunun yanında, tutarlar kullanıcının yerel para biriminde olmasa bile, normal olarak para birimi biçimlerinin kullanıcınıza doğru görünmesini istersiniz. Örneğin, Kanadalı bir kullanıcı bir Alman şirketinin fiyatlarını Euro cinsinden, ancak Kanadalı tarzda biçimlendirilmiş halde görmek isteyebilir.

CurrencyFormatter.setCurrency() yöntemi kullanılacak tam para birimi dizesini ve para birimini belirtir.

Aşağıdaki örnekte para birimi tutarları Kanada'nın Fransız kısmındaki kullanıcılara Euro cinsinden gösterilmektedir:

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

Kesin para birimi sembolleri belirleyerek kafa karışıklığını önlemek için setCurrency() yöntemi de kullanılabilir. Örneğin:

```
cf.setCurrency("USD","US$");
```

format() yöntemi para birimi sembolü yerine varsayılan olarak üç karakterli ISO 4217 para birimi kodunu gösterir. ISO 4217 kodları açıktır ve yerelleştirilmeleri gerekmez. Ancak, çoğu kullanıcı ISO kodları yerine para birimi sembollerini görmeyi tercih etmektedir.

CurrencyFormatter sınıfı, biçimlendirilmiş bir para birimi dizesinin hangi sembolü kullanacağına karar vermenize yardımcı olabilir: dolar işareti veya Euro işareti gibi bir para birimi sembolü veya USD veya EUR gibi üç karakterli ISO para birimi. Örneğin, Kanada doları miktarı Kanada'daki kullanıcılar için $200 olarak görüntülenebilir. Ancak A.B.D'deki bir kullanıcı için CAD 200 olarak görüntülenebilir. Tutarın para birimi sembolünün, kullanıcının yerel ayarlarına göre belirsiz veya yanlış olup olmadığını belirlemek için formattingWithCurrencySymbolIsSafe() yöntemini kullanın.

Aşağıdaki örnekte, Euro cinsinden bir değer en-US yerel ayarlarına dönüştürülmektedir. Kullanıcının yerel ayarına bağlı olarak, çıktı dizesi ya ISO para birimi kodu ya da para birimi sembolü kullanır.

```
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
}
```

Para birimi değerleri içeren dizeleri ayrıştırma

CurrencyFormatter sınıfı yerel ayara özel biçimlendirme gereksinimlerine uyan bir girdi dizesinden de para birimi tutarını ve para birimi dizesini alabilir. CurrencyFormatter.parse() yöntemi, ayrıştırılmış tutarı ve para birimi dizesini burada gösterildiği gibi bir CurrencyParseResult nesnesine depolar:

```
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
```

Girdi dizesinin para birimi dizesi kısmı bir para birimi sembolü, bir para birimi ISO kodu ve başka metin karakterleri içerebilir. Para birimi dizesinin, negatif sayı göstergesinin ve sayısal değerin konumları negativeCurrencyFormat ve positiveCurrencyFormat özellikleriyle belirtilen biçimlerle aynı olmalıdır. Örneğin:

```
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 $
```

Bu örnekte, girdi dizesinde eksi işareti ve sayının ardından bir para birimi dizesi bulunmaktadır. Ancak, en-US yerel ayarının varsayılan negativeCurrencyFormat değeri ilk önce negatif göstergenin gelmesi gerektiğini belirtmektedir. Sonuç olarak, parse() yöntemi bir hata oluşturur ve ayrıştırılmış değer NaN olur.

negativeCurrencyFormat özelliği para birimi dizesinin önce geleceğini belirtecek şekilde 2'ye ayarlandıktan sonra, parse() yöntemi başarılı olur.

Tarihleri ve saatleri biçimlendirme

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Tarih ve saat değerlerinin gösterilme biçimi de bölgeden bölgeye büyük farklılıklar gösterir. Örneğin, burada 1962 yılının ikinci günü öğleden sonra saat 1:01'in bazı yerel ayarlarda kısa biçimde nasıl gösterildiği verilmektedir:

Yerel ayar	Tarih ve Saat Biçimi
en-US (İngilizce, ABD)	1/2/62 1:01pm
fr-FR (Fransa, Fransızca)	2/1/62 13:01
ja-JP (Japonya, Japonca)	1962/2/1 13:01

DateTimeFormatter sınıfını kullanma

DateTimeFormatter sınıfı Tarih değerlerini belli bir yerel ayarın kurallarına göre tarih ve saat dizelerine biçimlendirir.

Biçimlendirme, tarih veya saat değerleriyle değiştirilen harf dizisini içeren desen dizisini takip eder. Örneğin, "yyyy/MM" deseninde, "yyyy" karakterlerinin yerine dört basamaklı yıl gelir, ardından "/" karakteri ve iki basamaklı bir ay koyulur.

Desen dizesi setDateTimePattern() yöntemi kullanılarak açıkça ayarlanabilir. Ancak, desenin kullanıcının yerel ayarına ve işletim sistemi tercihlerine göre otomatik olarak ayarlanmasına izin vermek en iyisidir. Bu uygulama sonucun kültürel açıdan uygun olmasını garantiler.

DateTimeFormatter, tarihleri ve saatleri üç standart stilde (UZUN, ORTA ve KISA) ifade edebilir ve bir ÖZEL desen de kullanabilir. Bu stillerden biri tarih, bir ikincisi ise saat için kullanılabilir. Her stil için kullanılan geçerli desenler, işletim sistemine göre biraz farklılık gösterebilir.

DateTimeFormatter nesnesini oluşturduğunuzda stilleri belirtebilirsiniz. Stil parametreleri belirtilmemişse, varsayılan olarak DateTimeStyle.LONG'a ayarlanır. Stilleri daha sonra aşağıdaki örnekte gösterildiği gibi setDateTimeStyles() yöntemiyle değiştirebilirsiniz:

```
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
```

Ay ve gün adlarını yerelleştirme

Çoğu uygulama, ayların ve haftanın günlerinin adlarını takvim görüntüleri veya açılır liste halinde kullanır.

Ay adlarının yerelleştirilmiş bir listesini DateTimeFormatter.getMonthNames() yöntemini kullanarak alabilirsiniz. İşletim sistemine bağlı olarak, tam ve kısa biçimler mevcut olabilir. Ay adlarını tam uzunluğunda almak için DateTimeNameStyle.FULL değerini geçin. Daha kısa sürümlerini almak içinse DateTimeNameStyle.LONG_ABBREVIATION veya DateTimeNameStyle.SHORT_ABBREVIATION değerlerini geçin.

Bazı dillerde, bir ay adı tarih biçiminde gün değerinin yanına yerleştirildiğinde değişir (-in haline geçer). Ay adlarını tek başına kullanmayı planlıyorsanız, DateTimeNameContext.STANDALONE değerini getMonthNames() yöntemine geçirmelisiniz. Ancak ay adlarını biçimlendirilmiş tarihlerde kullanmak istiyorsanız DateTimeNameContext.FORMAT değerini geçin.

```
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.
```

DateTimeFormatter.getWeekdayNames() yöntemi, hafta günlerinin adlarının yerelleştirilmiş bir listesini sunar. getWeekdayNames() yöntemi, getMonthNames() yöntemiyle aynı nameStyle ve bağlam parametrelerini kabul eder.

```
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.
```

Bunun yanında, getFirstWeekday() yöntemi günün seçili yerel ayarda geleneksel olarak hafta başını işaret eden indeks değerini verir.

Dizeleri sıralama ve harmanlama

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Harmanlama, nesneleri doğru sıraya dizme işlemidir. Harmanlama kuralları yerel ayarlara göre önemli ölçüde farklılık gösterir. Kurallar ayrıca bir listeyi sıralıyorsanız veya bir metin arama algoritmasındaki gibi benzer öğeleri eşleştiriyorsanız farklılık gösterir.

Sıralama yaparken, büyük ve küçük harfler veya aksan gibi harfçikler vb. küçük farklılıklar çoğunlukla önemlidir. Örneğin, ö harfi (üzerinde iki nokta olan o) genellikle Fransızca veya İngilizce'deki düz o harfi ile eş değerde görülür. Ancak, aynı harf İsveççe'de z harfini takip eder. Ayrıca, Fransızca'da ve diğer dillerin bazılarında, bir kelimedeki son aksan farkı bu kelimenin sıralı bir listedeki yerini belirler.

Arama yaparken, ilgili eşleşme bulma şansını arttırmak için çoğu zaman büyük/küçük harf veya aksan farklarını göz ardı etmek istersiniz. Örneğin, Fransızca bir belgede "cote" karakterleriyle yapılacak bir arama, büyük ihtimalle "cote", "côte" ve "coté" eşleşmelerini verir.

Collator sınıfını kullanma

Collator sınıfının başlıca yöntemleri ağırlıklı olarak sıralama için kullanılan compare() yöntemi ve eşleşen değerler için kullanılan equals() yöntemidir.

Aşağıdaki örnekte compare() ve equals() yöntemlerinin farklı davranışları gösterilmektedir.

```
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
}
```

Aynı örnekte ilk olarak Fransız-Fransa yerel ayarları için SIRALAMA modunda bir Collator nesnesi oluşturulur. Ardından, yalnızca aksan işaretleri farklı olan iki kelime sıralanır. Buradan SIRALAMA harmanlamasının aksanlı ve aksansız karakterleri birbirinden ayırdığı görülmektedir.

Sıralama, bir referansı Array.sort() yönteminin parametresi olarak Collator nesnesinin sort() yöntemine geçirerek gerçekleştirilir. Sıralama sırasını kontrol etmek için Collator nesnesini kullanmanın en etkili yollarından biri bu tekniktir.

Örnek bundan sonra EŞLEŞTİRME modunda bir Collator nesnesi oluşturur. Collator nesnesi iki kelimeyi karşılaştırdığında, bunlara eşitmiş gibi davranır. Bu da EŞLEŞTİRME harmanlamasının aksanlı ve aksansız karakterlere aynı değeri verdiğini göstermektedir.

Collator sınıfının davranışını özelleştirme

Collator sınıfı varsayılan olarak yerel ayara ve kullanıcının sistem tercihlerine göre işletim sisteminden elde edilen dize harmanlama kurallarını kullanır. Çeşitli özellikleri açıkça ayarlayarak compare() ve equals() yöntemlerinin davranışını özelleştirebilirsiniz. Aşağıdaki tabloda özellikler ve bu özelliklerin harmanlamalar üzerindeki etkisi listelenmektedir:

Collator Özelliği	Efekt
numericComparison	Sayı karakterlerine sayı veya metin olarak işlem yapılmasını kontrol eder.
ignoreCase	Büyük harf ve küçük harf farklılıklarının göz ardı edilip edilmediğini denetler.
ignoreCharacterWidth	Bazı Çince ve Japonca karakterlerin tam-geniş ve yarı-geniş hallerinin eşit olarak değerlendirilip değerlendirilmeyeceğini kontrol eder.
ignoreDiacritics	Aynı taban karakterleri kullanıp farklı aksanları veya aksan işaretleri bulunan dizelerin eşit olarak değerlendirilip değerlendirilmeyeceğini kontrol eder.
ignoreKanaType	Yalnızca kullanılmakta olan kana karakter tipi farklı olan dizelerin eşit olarak değerlendirilip değerlendirilmeyeceğini kontrol eder.
ignoreSymbols	Boşluk, para birimi sembolü, matematik sembolü ve başka sembol karakterlerinin göz ardı edilip edilmeyeceğini kontrol eder.

Aşağıdaki kod, ignoreDiacritics özelliğini true olarak ayarlamanın Fransızca kelimeler listesindeki sırayı değiştireceğini göstermektedir:

```
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
```

Büyük/Küçük harf dönüştürme

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Dillerin büyük harfli halleri (büyük) ile küçük harfli halleri (küçük) arasında harf dönüştürme kuralları da farklıdır.

Örneğin, Latin alfabesini kullanan çoğu dilde “I” baş harfinin küçük hali “i”dir. Ancak, bazı dillerde (Türkçe ve Azerice gibi) noktasız bir “ı” harfi de bulunur. Sonuç olarak bu dillerde küçük harfli noktasız “ı” büyük harfte ”I” olur. Küçük “i” ise büyük harfte noktalı “İ” olur.

StringTools sınıfı, bu tip dönüşümleri gerçekleştirmek için dile özel kuralları kullanan yöntemlere sahiptir.

StringTools sınıfını kullanma

StringTools sınıfı büyük/küçük harf dönüşümlerini gerçekleştirmek için iki yöntem kullanır: toLowerCase() ve toUpperCase(). Bir yerel ayar kimliğiyle yapıcıyı çağırarak StringTools nesnesi oluşturursunuz. StringTools sınıfı o yerel ayara (veya yedek yerel ayara) ait büyük/küçük harf dönüştürme kurallarını işletim sisteminden alır. Büyük/küçük harf dönüştürme algoritmasını daha da özelleştirmek mümkün değildir.

Aşağıdaki örnekte “ß” (keskin S) harfi içeren bir Almanca ifadeyi dönüştürmek için toUpperCase() ve toLowerCase() yöntemleri kullanılmaktadır.

```
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
```

toUpperCase() yöntemi küçük “ß” harfini büyük harf olarak “SS” öğesine dönüştürür. Bu dönüştürme yalnızca tek taraflıdır. “SS” harfleri yeniden küçük harfe dönüştürüldüğünde, sonuç “ß” değil “ss” olur.

Örnek: Bir stok onaylayıcı uygulamasını uluslararasılaştırma

Flash Player 10.1 ve üstü, Adobe AIR 2.0 ve üstü

Global Stok Onaylayıcı uygulaması, üç farklı borsadaki (Amerika Birleşik Devletleri, Japonya ve Avrupa) stoklar hakkında hayali veriler alır ve gösterir. Verileri farklı yerel ayarların kurallarına göre biçimlendirir.

Bu örnek, flash.globalization paketinin aşağıdaki özelliklerini göstermektedir:

- Yerele duyarlı sayı biçimlendirme
- Yerele duyarlı para birimi biçimlendirme
- Para birimi ISO kodunu ve para birimi sembollerini ayarlama
- Yerele duyarlı tarih biçimlendirme
- Uygun ay adı kısaltmalarını alma ve gösterme

Bu örneğin uygulama dosyalarını edinmek için bkz. www.adobe.com/go/learn_programmingAS3samples_flash_tr. Global Stok Onaylayıcı uygulama dosyaları, Örnekler/GlobalStockTicker klasöründe bulunabilir. Uygulama aşağıdaki dosyaları içerir:

File	Açıklama
GlobalStockTicker.mxml veya GlobalStockTicker.fla	Flex (MXML) veya Flash (FLA) uygulaması için kullanıcı arabirimi.
styles.css	Uygulama kullanıcı arabirimi için stiller (yalnızca Flex).
com/example/programmingas3/stockticker/flex/FinGraph.mxml	Temsili stok verilerinin bir çizelgesini gösteren MXML bileşeni (yalnızca Flex).
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	Uygulama için kullanıcı arabirimi mantığını içeren belge sınıfı (yalnızca Flash).
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Flash DataGrid bileşeni için özel bir hücre işleyici (yalnızca Flash).

File	Açıklama
com/example/programmingas3/stockticker/FinancialGraph.as	Temsili stok verilerinin bir çizelgesini çizen ActionScript sınıfı.
com/example/programmingas3/stockticker/Localizer.as	Yerel ayarı ve para birimini yönetip sayı, para birimi tutarı ve tarihlerin yerelleştirilmiş biçimlendirmesini yapan ActionScript sınıfı.
com/example/programmingas3/stockticker/StockDataModel.as	Global Stok Onaylayıcı örneğinin tüm örnek verilerini tutan ActionScript sınıfı.

Kullanıcı arabirimini ve örnek verileri anlama

Uygulamanın başlıca kullanıcı arabirimi öğeleri şunlardır:

- bir Yerel Ayar seçmek için aşağı açılır kutu
- bir Pazar seçmek için aşağı açılır kutu
- Her pazardaki altı şirketin verilerini gösteren bir DataGrid
- seçili şirketin stoğuna ait temsili geçmiş verilerini gösteren bir çizelge

Uygulama yerel ayarlar, pazarlar ve şirket stokları hakkındaki tüm örnek verilerini StockDataModel sınıfına depolar. Bunun gibi gerçek bir uygulama, verileri bir sunucudan alıp StockDataModel gibi bir sınıfa depolar. Bu örnekte, tüm veri StockDataModel sınıfına gömülmüştür.

Not: *Mali çizelgede gösterilen verilerin DataGrid kontrolünde gösterilen verilerle eşleşmesi şart değildir. Ne zaman farklı bir şirket seçilse, çizelge rastgele yeniden çizilir. Bu çizelge yalnızca örneklendirme amaçlıdır.*

Yerel ayarı belirleme

Başlangıçtaki birkaç kurulum işinin ardından, uygulama varsayılan yerel ayarın biçimlendirici nesnelerini oluşturmak üzere Localizer.setLocale() yöntemini çağırır. setLocale() yöntemi, kullanıcın aşağı açılır Yerel Ayar kutusundan her yeni değer alışıyla birlikte de çağrılır.

```
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);
    }
}
```

İlk olarak setLocale() yöntemi bir LocaleID nesnesi oluşturur. Bu nesne, daha sonra gerekirse geçerli yerel ayar hakkındaki ayrıntıları almayı kolaylaştırır.

Sonra yerel ayarın yeni NumberFormatter, CurrencyFormatter ve DateTimeFormatter nesnelerini oluşturur. Formatter nesnesi her oluşturulduğunda traceError() yöntemini çağırır. Bu yöntem, istenilen yerel ayarda sorun varsa konsolda hata ve uyarı mesajları görüntüler. (Gerçek bir uygulama, bu tip hataları sadece izlemek yerine bunlara göre harekete geçmelidir).

CurrencyFormatter nesnesi oluşturulduktan sonra, setLocale() yöntemi biçimlendiricinin para birimi ISO kodunu, para birimi sembolünü ve fractionalDigits özelliklerini önceden belirlenen değerlere ayarlar. (Bu değerler, kullanıcının aşağı açılır Pazarlar kutusundan her yeni pazar seçişiyle birlikte ayarlanır).

DateTimeFormatter nesnesi oluşturulduktan sonra, setLocale() yöntemi yerelleştirilmiş ay adı kısaltmalarının bir dizisini de alır.

Verileri biçimlendirme

Biçimlendirilmiş stok verileri bir DataGrid kontrolünde sunulur. DataGrid sütunlarının her biri, sütun değerini uygun biçimlendirici nesneyi kullanarak biçimlendiren bir etiket işlevi çağırır.

Örneğin, Flash sürümünde DataGrid sütunları aşağıdaki kodla kurulur:

```
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;
```

Örneğin Flex sürümü DataGrid'ini MXML'de ilan eder. Her sütunun etiket işlevlerini de tanımlar.

labelFunction özellikleri, Localizer sınıfının biçimlendirme yöntemlerini çağıran aşağıdaki işlevleri ifade eder:

```
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);
}
```

Localizer yöntemleri bundan sonra uygun biçimlendiricileri kurar ve çağırır:

```
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);
}
|
```

Bölüm 57: Uygulamaları yerelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yerelleştirme, birden çok yerel ayarı desteklemek için varlıkları ekleme işlemidir. Yerel ayar, dil ve ülke kodunun kombinasyonudur. Örneğin en_US Birleşik Devletler'de konuşulan İngilizceyi, fr_FR Fransa'da konuşulan Fransızcayı belirtir. Bu yerel ayarlar için bir uygulamayı yerelleştirmek üzere iki varlık kümesi sağlarsınız: biri en_US yerel ayarı, diğeri fr_FR yerel ayarı için.

Yerel ayarlar dilleri paylaşabilir. Örneğin en_US ve en_GB (İngiltere) farklı yerel ayarlardır. Bu durumda her iki yerel ayar da İngilizce dilini kullanır, ancak ülke kodu onların farklı yerel ayarlar olduğunu ve bu nedenle farklı varlıklar kullanabileceklerini gösterir. Örneğin en_US yerel ayarındaki bir uygulama "color" kelimesini yazabilir, ancak aynı kelime en_GB yerel ayarında "colour" şeklindedir. Ayrıca para birimleri yerel ayara bağlı olarak dolar veya pound biçimindedir ve tarih ve saat formatı da farklı olabilir.

Ayrıca ülke kodunu belirtmeden de bir dil için bir varlık kümesi sağlayabilirsiniz. Örneğin, İngilizce dili için en varlıkları ve en_US yerel ayarı için de ABD İngilizcesi'ne özgü ek varlıklar sağlayabilirsiniz.

Yerelleştirme, uygulamanızda kullanılan dizeleri çevirmenin ötesinde bir işlemdir. Ses dosyaları, görüntüler ve videolar gibi her tür varlığı içerebilir.

Bir yerel ayar seçme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamanızın hangi yerel ayarı kullandığını belirlemek için, aşağıdaki yöntemlerden birini kullanabilirsiniz:

- flash.globalization paketi — kullanıcının işletim sistemi ve kullanıcı tercihlerine göre varsayılan yerel ayarı almak için flash.globalization paketindeki yerele duyarlı sınıfları kullanın. Flash Player 10.1 veya üstü ya da AIR 2.0 veya üstü çalışma zamanlarında çalışacak uygulamalar için tercih edilen yaklaşım budur. Daha fazla bilgi edinmek için bkz. “Yerel ayarı belirleme” sayfa 954.
- Kullanıcı istemi — Uygulamayı varsayılan bir yerel ayarda başlatıp, ardından kullanıcıdan tercih ettiği yerel ayarı seçmesini isteyebilirsiniz.
- `(Yalnızca AIR) Capabilities.languages` — `Capabilities.languages` özelliği, işletim sistemi üzerinden ayarlandığı şekilde kullanıcının tercih edilen dillerinde bulunan dillerin bir dizisini listeler. Dizeler RFC4646 (http://www.ietf.org/rfc/rfc4646.txt) tarafından tanımlanan dil etiketleri içerir (varsa komut ve bölge bilgileri). Dizeler sınırlayıcı olarak tire kullanır (örneğin `"en-US"` veya `"ja-JP"`). Döndürülen dizideki ilk giriş, language özelliğiyle aynı birincil dil kimliğine sahipti. Örneğin `languages[0]"en-US"` olarak ayarlıysa, `language` özelliği şu şekilde ayarlıdır: `"en"`. Ancak language özelliği `"xu"` (bilinmeyen bir dili belirtir) olarak ayarlıysa, `languages` dizisindeki ilk öğe farklıdır.
- `Capabilities.language` — `Capabilities.language` özelliği, işletim sisteminin kullanıcı arabirimi dil kodunu sağlar. Ancak bu özellik 20 bilinen dille sınırlıdır. İngilizce sistemlerde bu özellik yalnızca dil kodunu döndürür, ülke kodunu döndürmez. Bu nedenlerden ötürü, `Capabilities.languages` dizisindeki ilk öğeyi kullanmak daha yararlıdır.

Flex içeriğini yerelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe Flex, Flex içeriğini yerelleştirmeye yönelik bir çerçeve içerir. Bu çerçeve Locale, ResourceBundle ve ResourceManagerImpl sınıflarını ve bunların yanı sıra IResourceBundle, IResourceManagerImpl arabirimlerini içerir.

Uygulama yerel ayarlarını sıralama için hizmet sınıflarını içeren bir Flex yerelleştirme kütüphanesi Google Code (http://code.google.com/p/as3localelib/) sitesinde mevcuttur.

Daha fazla Yardım konusu

http://code.google.com/p/as3localelib/

Flaş içeriğini yerelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe Flash Professional, ActionScript 3.0 bileşenlerinde bir Locale sınıfı içerir. Locale sınıfı sayesinde, bir SWF dosyasının çok dilli bir metni nasıl görüntülediğini kontrol edebilirsiniz. Flash Dizeler paneli sayesinde, dinamik metin alanlarında dize sabit değerleri yerine dize kimliklerini kullanabilirsiniz. Bu hizmet, dile özgü bir XML dosyasından yüklenen metni görüntüleyen bir SWF dosyası oluşturmanıza olanak verir. Locale sınıfını kullanmayla ilgili bilgi için bkz. Adobe Flash Platform için ActionScript 3.0 Başvurusu.

AIR uygulamalarını yerelleştirme

Adobe AIR 1.0 ve üstü

AIR SDK bir HTML Yerelleştirme Çerçevesi (AIRLocalizer.js dosyasında bulunan) sağlar. Bu çerçeve, bir HTML tabanlı uygulamada birden fazla yerel ayarla çalışmaya destek olan API'leri içerir. Yerel ayarları sıralamak için kullanabileceğiniz bir ActionScript kütüphanesi http://code.google.com/p/as3localelib/ adresinde mevcuttur.

Daha fazla Yardım konusu

http://code.google.com/p/as3localelib/

Tarihleri, saatleri ve para birimlerini yerelleştirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulamaların tarihi, saati ve para birimini sunma şekli her yerel ayar için birbirinden oldukça farklıdır. Örneğin tarihler için ABD standardı ay/gün/yıl, Avrupa için tarihleri sunma şekliyse gün/ay/yıl şeklindedir.

Tarihleri, saatleri ve para birimlerini formatlamak için kod yazabilirsiniz. Örneğin aşağıdaki kod bir Date nesnesini ay/gün/yıl veya gün/ay/yıl formatına dönüştürür. `locale` değişkeni (yerel ayarı temsil eden) `"en_US"` olarak ayarlıysa, işlev ay/gün/yıl formatını döndürür. Örnek bir Date nesnesini tüm diğer yerel ayarlar için gün/ay/yıl formatına dönüştürür:

```
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

Flex çerçevesi tarihleri, saatleri ve para birimlerini biçimlendirmek için kontroller içerir. Bu kontrollere DateFormatter ve CurrencyFormatter kontrolleri dahildir.

- mx:DateFormatter
- mx:CurrencyFormatter

Bölüm 58: HTML ortamı hakkında

Adobe AIR 1.0 ve üstü

Adobe® AIR®, Safari web tarayıcısı tarafından da HTML ve JavaScript içeriğini ayrıştırmak, oluşturmak ve mizanpajını yapmak için kullanılan WebKit *(www.webkit.org)* öğesini kullanır. AIR API'lerinin HTML içeriğinde kullanımı isteğe bağlıdır. Yalnızca HTML ve JavaScript ile, HTMLLoader nesnesinin veya HTML penceresinin içeriğinde programlama yapabilirsiniz. Varolan birçok HTML sayfası ve uygulaması birkaç değişiklikle çalışmalarını sürdürmelidir (WebKit ile uyumlu HTML, CSS, DOM ve JavaScript özellikleri kullandıklarını varsayarak).

Önemli: Adobe AIR çalışma zamanının yeni sürümleri güncellenmiş WebKit sürümleri içerebilir. Yeni bir AIR sürümündeki bir WebKit güncellemesi konuşlandırılmış bir AIR uygulamasında beklenmeyen değişikliklere yol açabilir. Bu değişiklikler bir uygulamadaki HTML içeriğinin davranış ve görünümünü etkileyebilir. Örneğin, WebKit oluşturmasındaki iyileştirmeler veya düzeltmeler bir uygulamanın kullanıcı arabirimindeki öğelerin mizanpajını değiştirebilir. Bu nedenden dolayı, uygulamanızda bir güncelleme mekanizması sağlamanız önerilir. Uygulamanızı AIR uygulamasında bulunan Webkit sürümündeki bir değişiklikten dolayı güncellemeniz gerekirse, AIR güncelleme mekanizması kullanıcıdan uygulamanın yeni bir sürümünü yüklemesini isteyebilir.

Aşağıdaki tablo, AIR'de kullanılanla eşdeğer WebKit sürümünü kullanan Safari web tarayıcısı sürümlerini listeler:

AIR sürümü	Safari sürümü
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

Bir HTMLLoader nesnesi tarafından döndürülen varsayılan kullanıcı aracısı dizesini inceleyerek yüklü WebKit sürümünü her zaman belirleyebilirsiniz:

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

AIR'de kullanılan WebKit sürümünün açık kaynak sürümüyle aynı olmadığını unutmayın. Bazı özellikler AIR'de desteklenmezken, ilgili WebKit sürümünde henüz mevcut olmayan güvenlik ve hata düzeltmeleri AIR sürümünde bulunabilir. Bkz. "AIR'de desteklenmeyen WebKit özellikleri" sayfa 987.

AIR uygulamaları dosya sistemine tam erişimle doğrudan masaüstünde çalıştığı için, HTML içeriğinin güvenlik modeli, tipik bir web tarayıcısının güvenlik modelinden daha katıdır. AIR'de, yalnızca uygulama yükleme dizininden yüklenen veriler *uygulama sanal alanına* yerleştirilir. Uygulama sanal alanı en yüksek düzeyde ayrıcalığa sahiptir ve AIR API'lerine erişim sağlar. AIR, nereden geldiğini temel alarak diğer içeriği ayrı sanal alanlara yerleştirir. Dosya sisteminden yüklenen dosyalar yerel bir sanal alana gider. Ağdan http: veya https: protokolleri kullanılarak yüklenen dosyalar, uzak sunucunun etki alanını temel alarak bir sanal alana gider. Bu uygulama dışı sanal alanlardaki içeriğin herhangi bir AIR API'sine erişmesi yasaktır ve tipik bir web tarayıcısında olduğu gibi çalışır.

Alfa, ölçekleme veya saydamlık ayarları uygulanırsa, AIR'deki HTML içeriği SWF veya PDF içeriğini görüntülemez. Daha fazla bilgi için bkz. "HTML sayfasına SWF ve PDF içeriği yüklenirken dikkate alınması gerekenler" sayfa 1015 ve "Pencere şeffaflığı" sayfa 904.

Daha fazla Yardım konusu

WebKit DOM Başvurusu

Safari HTML Başvurusu

Safari CSS Başvurusu

www.webkit.org

HTML ortamına genel bakış

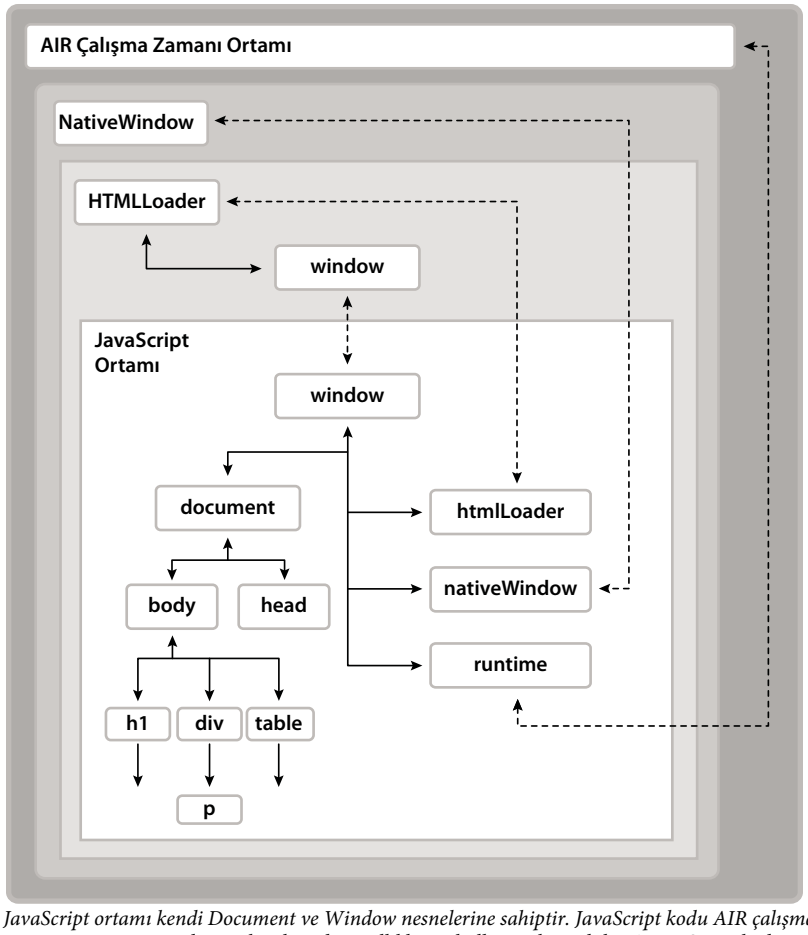
Adobe AIR 1.0 ve üstü

Adobe AIR, HTML oluşturucu, belge nesnesi modeli ve JavaScript yorumlayıcısıyla beraber eksiksiz bir tarayıcı benzeri JavaScript ortamı sağlar. JavaScript ortamı, AIR HTMLLoader sınıfı tarafından temsil edilir. HTML pencerelerinde HTMLLoader nesnesi tüm HTML içeriğini kapsar ve bunun karşılığında NativeWindow nesnesinde de bulunur. SWF içeriğinde Sprite sınıfını genişleten HTMLLoader sınıfı, diğer görüntüleme nesneleri gibi, bir sahnenin görüntüleme listesine eklenebilir. Sınıfın Adobe® ActionScript® 3.0 özellikleri "AIR HTML Kabı için Komut Dosyası Oluşturma" sayfa 1013 içinde ve ayrıca *Adobe Flash Platform için ActionScript 3.0 Başvurusu* içinde açıklanır. Flex çerçevesinde, AIR HTMLLoader sınıfı mx:HTML bileşeninde sarılır. mx:HTML bileşeni UIComponent sınıfını genişletir, böylece bu sınıf doğrudan diğer Flex kaplarıyla birlikte kullanılabilir. Aksi takdirde mx:HTML bileşeni içindeki JavaScript ortamı aynıdır.

JavaScript ortamı ve AIR ana bilgisayarı ile ilişkisi hakkında

Adobe AIR 1.0 ve üstü

Aşağıdaki diyagram JavaScript ortamı ve AIR çalışma zamanı ortamı arasındaki ilişkiyi anlatmaktadır. Yalnızca tek bir yerel pencere gösterilmesine rağmen, AIR uygulaması birden çok pencereyi içerebilir. (Ve tek bir pencere birden çok HTMLLoader nesnesi içerebilir.)



JavaScript ortamı kendi Document ve Window nesnelerine sahiptir. JavaScript kodu AIR çalışma zamanı ortamıyla etkileşime çalışma zamanını, nativeWindow ve htmlLoader özelliklerini kullanarak geçebilir. ActionScript kodu JavaScript ortamıyla etkileşime, JavaScript Window nesnesine bir başvuru olan HTMLLoader nesnesinin pencere özelliğini kullanarak geçebilir. Ayrıca hem ActionScript hem de JavaScript nesneleri, AIR ve JavaScript nesnelerinin her ikisi tarafından da gönderilen olayları dinleyebilir.

`runtime` özelliği, AIR API sınıflarına erişim sağlar ve access sınıfı (statik de denir) üyelerinin yanı sıra, yeni AIR nesneleri oluşturmanızı da sağlar. Bir AIR API'sine erişmek için, sınıfın adını paketle beraber `runtime` özelliğine eklersiniz. Örneğin bir File nesnesi oluşturmak için, şu ifadeyi kullanırsınız:

```
var file = new window.runtime.filesystem.File();
```

Not: *AIR SDK bir JavaScript dosyası sağlar, `AIRAliases.js`, bu dosya en yaygın kullanılan AIR sınıfları için daha uygun başka adlar tanımlar. Bu dosyayı içe aktardıktan sonra, window.runtime.package.Class yerine daha kısa bir biçim olan air.Class'ı kullanabilirsiniz. Örneğin File nesnesini `new air.File()` ile oluşturabilirsiniz.*

NativeWindow nesnesi, masaüstü penceresini kontrol etmeye yönelik özellikler sağlar. Bir HTML sayfasından, `window.nativeWindow` özelliğine sahip NativeWindow nesnesine erişebilirsiniz.

HTMLLoader nesnesi içeriğin nasıl yüklendiğine ve oluşturulduğuna dair özellikler, yöntemler ve olaylar sağlar. Bir HTML sayfasından, `window.htmlLoader` özelliğine sahip üst HTMLLoader nesnesine erişebilirsiniz.

Önemli: *Yalnızca bir uygulamanın parçası olan sayfalar üst düzey belge olarak yüklendiklerinde, `htmlLoader`, `nativeWindow` veya `runtime` özelliklerine sahip olur. Bu özellikler belge bir frame veya iframe'e yüklendiğinde eklenmez. (Alt belge, aynı güvenlik sanal alanında olduğu sürece üst belgedeki bu özelliklere erişebilir. Örneğin bir kareye yüklenmiş olan belge, üst öğesinin `runtime` özelliğine `parent.runtime` ile erişebilir.)*

Güvenlik hakkında

Adobe AIR 1.0 ve üstü

AIR, kaynağın etki alanına dayalı olarak bir güvenlik sanal alanındaki tüm kodu çalıştırır. Uygulama yükleme dizininden yüklenen içerikle sınırlı olan uygulama içeriği, *uygulama* sanal alanına yerleştirilir. Çalışma zamanı ortamına ve AIR API'lerine erişim yalnızca bu sanal alanın içinden HTML ve JavaScript'in çalıştırılmasıyla mümkündür. Aynı zamanda, `load` olayı sayfası için tüm işleyiciler geri döndükten sonra, en dinamik JavaScript değerlendirmesi ve çalışması uygulama sanal alanında engellenir.

Bir uygulama sayfasını, sayfayı bir frame veya iframe'e yükleyerek ve karenin AIR'e özgü `sandboxRoot` ve `documentRoot` niteliklerini ayarlayarak uygulama olmayan sanal alana eşleyebilirsiniz. `sandboxRoot` değerini gerçek bir uzak etki alanına ayarlayarak, bu alandaki içeriği çapraz komut dosyasına yazmak üzere sanal alana sahip içeriği etkinleştirebilirsiniz. Sayfaları bu şekilde eşlemek, *içerik birleştirme* uygulamasında olduğu gibi uzak içeriği yüklerken ve komut dosyasına yazarken yararlı olabilir.

Uygulama içeriğinin ve uygulama olmayan içeriğin birbirlerinin çapraz dosyasını oluşturmasını sağlamanın bir başka yolu ve aynı zamanda AIR API'lerine uygulama olmayan içerik erişimi vermenin tek yolu, *sanal alan köprüsü* oluşturmaktır. *Üst-alt* köprüsü bir alt karedeki, iframe'deki veya penceredeki içeriğin, uygulama sanal alanında tanımlanan belirli yöntemlere ve özelliklere erişmesini sağlar. Tersine, *alt-üst* köprüsü, uygulama içeriğinin alt öğenin sanal alanında tanımlanan belirli yöntemlere ve özelliklere erişmesini sağlar. Sanal alan köprüleri, window nesnesinin `parentSandboxBridge` ve `childSandboxBridge` özellikleri ayarlanarak kurulur. Daha fazla bilgi için, bkz. "Adobe AIR'de HTML güvenliği" sayfa 1092 ve "HTML frame ve iframe öğeleri" sayfa 983.

Eklentiler ve gömülü nesneler hakkında

Adobe AIR 1.0 ve üstü

AIR, Adobe® Acrobat® eklentisini destekler. Kullanıcıların PDF içeriğini görüntülemek için Acrobat veya Adobe® Reader® 8.1 (veya daha üst) sürümüne sahip olmaları gerekmektedir. HTMLLoader nesnesi, kullanıcı sisteminin PDF'leri görüntüleyip görüntülemediğini kontrol etmeye yarayan bir özellik sağlar. SWF dosya içeriği HTML ortamında da görüntülenebilir, ancak bu özellik AIR'de yerleşiktir ve harici bir eklenti kullanmaz.

AIR'de başka Webkit eklentileri desteklenmez.

Daha fazla Yardım konusu

"Adobe AIR'de HTML güvenliği" sayfa 1092

"HTML Sanal Alanları" sayfa 976

"HTML frame ve iframe öğeleri" sayfa 983

"JavaScript Window nesnesi" sayfa 981

"XMLHttpRequest nesnesi" sayfa 977

"AIR'ye PDF içeriği ekleme" sayfa 557

AIR ve Webkit

Adobe AIR 1.0 ve üstü

Adobe AIR, Safari web tarayıcısında da kullanılan açık kaynak Webkit motorunu kullanır. AIR, güvenlik nedenlerinin yanı sıra, çalışma zamanı sınıflarına ve nesnelerine erişim sağlamak için birçok uzantı ekler. Ayrıca Webkit de HTML, CSS ve JavaScript için W3C standartlarında bulunmayan özellikler ekler.

Burada yalnızca AIR eklemeleri ve en önemli Webkit uzantıları ele alınmaktadır; standart olmayan HTML, CSS ve JavaScript ile ilgili ek belgeler için, bkz. www.webkit.org ve developer.apple.com. Standart bilgileri için, W3C sitesine bakın. Mozilla ayrıca HTML, CSS ve DOM konularında önemli bir genel başvuru sağlar (Webkit ve Mozilla motorları birbirine benzemez).

AIR'de JavaScript

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

AIR, ortak JavaScript nesnelerinin tipik davranışında birçok değişiklik yapar. Bu değişikliklerin birçoğu AIR'de güvenli uygulamaların yazılmasını kolaylaştırmak için yapılır. Aynı zamanda, davranıştaki bu farklılıklar, bazı ortak JavaScript kodlama desenlerinin ve bu desenleri kullanan mevcut web uygulamalarının her zaman AIR'de beklendiği gibi çalıştırılamayabileceği anlamına gelir. Bu tür sorunların düzeltilmesiyle ilgili bilgi için bkz. “Güvenlikle ilişkili JavaScript hatalarını önleme” sayfa 993.

HTML Sanal Alanları

Adobe AIR 1.0 ve üstü

AIR içeriği kaynağına bağlı olarak, ayrı sanal alanlara yerleştirir. Sanal alan kuralları, Adobe Flash Player tarafından uygulanan sanal alan kurallarının yanı sıra, çoğu web tarayıcısı tarafından uygulanan aynı kaynak ilkesiyle tutarlıdır. Ayrıca AIR, uygulama içeriğini dahil etmek ve korumak için yeni bir *uygulama* sanal alan türü sağlar. AIR uygulamalarını geliştirirken karşılaşabileceğiniz sanal alan türleriyle ilgili daha fazla bilgi için, bkz. “Güvenlik sanal alanları” sayfa 1055.

Çalışma zamanı ortamına ve AIR API'lerine erişim yalnızca uygulama sanal alanının içinden HTML ve JavaScript'in çalıştırılmasıyla mümkündür. Ancak aynı zamanda çeşitli biçimlerdeki dinamik JavaScript değerlendirmesi ve çalışması, güvenlik nedenleriyle büyük ölçüde uygulama sanal alanıyla sınırlıdır. Bu kısıtlamalar, uygulamanız doğrudan bir sunucudan bilgi yüklese de yüklemese de geçerlidir. (Hatta dosya içeriği, yapıştırılan dizeler ve doğrudan kullanıcı girişi güvenilmez olabilir.)

Bir sayfadaki içeriğin kaynağı, gönderildiği sanal alanı belirler. Yalnızca uygulama dizininden yüklenen içerik (`app:` URL şeması tarafından başvurulan yükleme dizini) uygulama sanal alanına yerleştirilir. Dosya sisteminden yüklenen içerik *local-with-filesystem* veya *local-trusted* sanal alanına yerleştirilir, böylece yerel dosya sistemindeki içeriğe erişimine ve bu içerikle etkileşimine izin verilir, ancak uzak içeriğe erişimine izin verilmez. Ağdan yüklenen içerik, kaynak etki alanına karşılık gelen uzak bir sanal alana yerleştirilir.

Bir uygulama sayfasının uzak sanal alandaki içerikle serbestçe etkileşmesini sağlamak için, sayfa, uzak içerikle aynı etki alanına eşlenebilir. Örneğin bir Internet hizmetinden eşleme verileri görüntüleyen bir uygulama yazarsanız, uygulamanızın hizmetten içerik yükleyen ve görüntüleyen sayfası hizmet etki alanına eşlenebilir. Sayfaları uzak sanal alana ve etki alanına eşleme nitelikleri, kare ve iframe HTML öğelerine eklenen yeni niteliklerdir.

Uygulama dışı bir sanal alandaki içeriğin AIR özelliklerini güvenli bir şekilde kullanmasını sağlamak için, üst sanal alan köprüsü kurabilirsiniz. Uygulama içeriğinin yöntemleri güvenli bir şekilde çağırmasını ve diğer sanal alanlardaki içeriğin özelliklerine erişmesini sağlamak için, alt sanal alan köprüsü kurabilirsiniz. Güvenli burada uzak içeriğin açık olmayan nesnelere, özelliklere veya yöntemlere yanlışlıkla başvuru alamaması anlamına gelir. Yalnızca basit veri türleri, işlevler ve adsız nesneler köprüden geçirilebilir. Ancak yine de tehlikeli olabilecek işlevleri açmaktan kaçınmalısınız. Örneğin, bir kullanıcının sisteminde bulunan dosyaları okumayı ve dosyalara yazmayı sağlayan bir arabirimi açtıysanız, uzak içeriğe, kullanıcılarınıza önemli zararlar verebileceği bir yöntem sağlıyor olabilirsiniz.

JavaScript eval() işlevi

Adobe AIR 1.0 ve üstü

Sayfa, yükleme işlemini bitirdiğinde, `eval()` işlevinin kullanımı uygulama sanal alanıyla sınırlanır. Bazı kullanımlara izin verilir ve böylece JSON biçimli veriler güvenli bir şekilde ayrıştırılabilir, ancak çalıştırılabilir bir ifadeyle sonuçlanan herhangi bir değerlendirme hatayla sonuçlanır. "Farklı sanal alanlardaki içerik için kod kısıtlamaları" sayfa 1094, `eval()` işlevinin izin verilen kullanımlarını açıklar.

İşlev yapıcıları

Adobe AIR 1.0 ve üstü

Uygulama sanal alanında, bir sayfa yükleme işlemini bitirmeden önce işlev yapıcıları kullanılabilir. Tüm sayfa `load` olayı işleyicileri bittikten sonra, yeni işlevler oluşturulamaz.

Harici komut dosyalarını yükleme

Adobe AIR 1.0 ve üstü

Uygulama sanal alanındaki HTML sayfaları, JavaScript dosyalarını uygulama dizininin dışından yüklemek için `script` etiketini kullanamaz. Uygulamanızdaki bir sayfaya uygulama dizininin dışından bir komut dosyası yüklenmesi için,, sayfa uygulama dışı bir sanal alana eşlenmelidir.

XMLHttpRequest nesnesi

Adobe AIR 1.0 ve üstü

AIR, uygulamaların veri istekleri gerçekleştirebilmek için kullanabileceği bir XMLHttpRequest (XHR) nesnesi sağlar. Aşağıdaki örnek basit bir veri isteğini gösterir:

```
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);
```

Bir tarayıcıya karşılık AIR, uygulama sanal alanında çalışan içeriğin herhangi bir etki alanından veri isteğinde bulunmasına olanak tanır. Bir JSON dizesi içeren XHR'nin sonucu, sonuç çalıştırılabilir kod içermediği sürece veri nesneleri olarak değerlendirilebilir. XHR sonucunda çalıştırılabilir ifadeler mevcutsa, bir hata verilir ve değerlendirme denemesi başarısız olur.

Uzak kaynaklardaki kodun yanlışlıkla dahil edilmesini engellemek için bu işlem sayfa, yükleme işlemini bitirmeden önce yapılırsa senkronize XHR'ler boş bir sonuç döndürür. Senkronize olmayan XHR'ler her zaman bir sayfa yüklendikten sonra geri döner.

Varsayılan olarak AIR, uygulama dışı sanal alanlardaki etki alanları arası XMLHttpRequest öğelerini engeller. Uygulama sanal alanındaki bir üst pencere, uygulama dışı sanal alanda içeriğe sahip olan bir alt karedeki etki alanları arası isteklere, AIR tarafından eklenen bir nitelik olan `allowCrossDomainXHR` öğesini içeriğe sahip olan kare veya iframe öğesini `true` olarak ayarlayarak izin vermeyi seçebilir:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

Not: *Uygun olduğunda, AIR URLStream sınıfı da veri yüklemek için kullanılabilir.*

Bir XMLHttpRequest öğesini uzak sanal alana eşlenmiş uygulama içeriğine sahip olan frame veya iframe'den uzak bir sunucuya gönderirseniz, eşleme URL'sinin XHR'de kullanılan sunucu adresini maskelemediğinden emin olun. Örneğin example.com etki alanı için uygulama içeriğini uzak bir sanal alana eşleyen aşağıdaki iframe tanımını göz önünde bulundurun:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

`sandboxRoot` niteliği www.example.com adresinin kök URL'sini yeniden eşlediği için, tüm istekler uzak sunucudan değil, uygulama dizininden yüklenir. Sayfa gezinmesinden veya XMLHttpRequest öğesinden türetilen istekler yeniden eşlenir.

Uzak sunucunuza yapılan veri isteklerini yanlışlıkla engellemeyi önlemek için, `sandboxRoot` öğesini köke değil, uzak URL'nin alt dizinine eşleyin. Dizinin mevcut olması gerekmez. Örneğin, uygulama dizini yerine uzak sunucudan yükleme yapabilmek için www.example.com adresine yapılan veri isteklerine izin vermek üzere, önceki iframe'i şu şekilde değiştirin:

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

Bu durumda yalnızca `air` alt dizinindeki içerik yerel olarak yüklenir.

Sanal alan eşlemesiyle ilgili daha fazla bilgi için, bkz. “HTML frame ve iframe öğeleri” sayfa 983 ve “Adobe AIR'de HTML güvenliği” sayfa 1092.

Tanımlama Bilgileri

Adobe AIR 1.0 ve üstü

AIR uygulamalarında yalnızca uzak sanal alanlardaki içerik (http: ve https: kaynaklarından yüklenen), tanımlama bilgilerini kullanabilir (`document.cookie` özelliği). Uygulama sanal alanında, kalıcı verileri saklamak için başka yollar da mevcuttur, bunlara EncryptedLocalStore, SharedObject ve FileStream sınıfları dahildir.

Clipboard nesnesi

Adobe AIR 1.0 ve üstü

WebKit Clipboard API'si şu olaylarla yürütülür: `copy`, `cut` ve `paste`. Bu olaylarda geçen olay nesnesi `clipboardData` özelliğiyle panoya erişim sağlar. Pano verilerini okumak veya bu verileri yazmak için aşağıdaki `clipboardData` nesnesi yöntemlerini kullanın:

Yöntem	Açıklama
clearData(mimeType)	Pano verilerini temizler. `mimeType` parametresini temizlenecek verilerin MIME türüne ayarlayın.
getData(mimeType)	Pano verilerini alır. Bu yöntem yalnızca `paste` olayının işleyicisinde çağrılabilir. `mimeType` parametresini geri döndürülecek verilerin MIME türüne ayarlayın.
setData(mimeType, data)	Verileri panoya kopyalar. `mimeType` parametresini verilerin MIME türüne ayarlayın.

Uygulama sanal alanının dışındaki JavaScript kodu panoya yalnızca bu olayları kullanarak erişebilir. Ancak uygulama sanal alanındaki içerik, AIR Clipboard sınıfını kullanarak sistem panosuna doğrudan erişebilir. Örneğin panodaki metin biçimi verilerini almak için aşağıdaki ifadeyi kullanabilirsiniz:

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                air.ClipboardTransferMode.ORIGINAL_ONLY);
```

Geçerli veri MIME türleri şunlardır:

MIME türü	Değer
Metin	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
Nokta Esaslı	"image/x-vnd.adobe.air.bitmap"
Dosya listesi	"application/x-vnd.adobe.air.file-list"

Önemli: *Yalnızca uygulama sanal alanındaki içerik panoda bulunan dosya verilerine erişebilir. Uygulama dışı içerik panodaki bir file nesnesine erişme girişiminde bulunursa, bir güvenlik hatası verilir.*

Panoyu kullanmayla ilgili daha fazla bilgi için bkz. "Kopyalama ve yapıştırma" sayfa 602 ve Using the Pasteboard from JavaScript (Apple Geliştirici Merkezi) (JavaScript'te Yapıştırma Alanı'nı Kullanma).

Sürükle ve Bırak

Adobe AIR 1.0 ve üstü

HTML'e doğru veya HTML'den sürükle ve bırak hareketleri şu DOM olaylarını üretir: `dragstart`, `drag`, `dragend`, `dragenter`, `dragover`, `dragleave` ve `drop`. Bu olaylarda geçen olay nesnesi, `dataTransfer` özelliğiyle sürüklenen verilere erişim sağlar. `dataTransfer` özelliği, bir clipboard olayıyla ilişkilendirilen `clipboardData` nesnesiyle aynı yöntemleri sağlayan bir nesneye başvurur. Örneğin bir `drop` olayından metin biçimi verileri almak için şu işlevi kullanabilirsiniz:

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

`dataTransfer` nesnesinin sahip olduğu önemli üyeler şunlardır:

Üye	Açıklama
clearData(mimeType)	Verileri temizler. `mimeType` parametresini temizlenecek veri temsilinin MIME türüne ayarlayın.
getData(mimeType)	Sürüklenen verileri alır. Bu yöntem yalnızca `drop` olayının işleyicisinde çağrılabilir. `mimeType` parametresini alınacak verilerin MIME türüne ayarlayın.
setData(mimeType, data)	Sürüklenecek verileri ayarlayın. `mimeType` parametresini verilerin MIME türüne ayarlayın.
türler	Şu anda `dataTransfer` nesnesinde kullanılabilir olan tüm veri temsillerinin MIME türlerini içeren dizeler dizisi.
effectsAllowed	Sürüklenen verilerin kopyalanabileceğini, taşınabileceğini, verilere bağ verilebileceğini veya bunların bir kombinasyonunun uygulanabileceğini belirtir. `effectsAllowed` özelliğini, `dragstart` olayının işleyicisinde ayarlayın.
dropEffect	İzin verilerin bırakma efektlerinden hangilerinin sürükleme hedefi tarafından desteklendiğini belirtir. `dropEffect` özelliğini `dragEnter` olayının işleyicisinde ayarlayın. Sürükleme sırasında imleç, kullanıcının fareyi bırakması durumunda hangi efektin oluşacağını göstermek için değişir. `dropEffect` belirtilmezse, `effectsAllowed` özellik efekti seçilir. Kopyala efektinin taşı efektine göre önceliği, taşı efektinin de bağ ver efektine göre önceliği vardır. Kullanıcı klavyeyi kullanarak varsayılan önceliği değiştirebilir.

AIR uygulamasına sürükle ve bırak eylemine yönelik destek eklemeyle ilgili daha fazla bilgi için bkz. "AIR uygulamasında taşıma ve bırakma" sayfa 614 ve JavaScript'te Sürükleme ve Bırakmayı Kullanma (Apple Developer Center).

innerHTML ve outerHTML özellikleri

Adobe AIR 1.0 ve üstü

AIR, uygulama sanal alanında çalışan içerik için `innerHTML` ve `outerHTML` özelliklerinin kullanılmasına yönelik güvenlik kısıtlamaları getirir. load olayı işleyicilerinin çalıştırılması sırasında ve sayfa load olayından önce, `innerHTML` ve `outerHTML` özelliklerinin kullanımında hiçbir kısıtlama yoktur. Ancak sayfa yüklendikten sonra, `innerHTML` veya `outerHTML` özelliklerini yalnızca belgeye statik içerik eklemek için kullanabilirsiniz. Çalıştırılabilir kod olarak değerlendirilen `innerHTML` veya `outerHTML` özelliklerine atanan bir dizede bulunan herhangi bir ifade yok sayılır. Örneğin bir olay geri çağırma niteliğini öğe tanımına dahil ederseniz, olay dinleyicisi eklenmez. Benzer şekilde, gömülü `<script>` etiketleri de değerlendirilmez. Daha fazla bilgi için, bkz. "Adobe AIR'de HTML güvenliği" sayfa 1092.

Document.write() ve Document.writeln() yöntemleri

Adobe AIR 1.0 ve üstü

Sayfanın `load` olayından önce uygulama sanal alanında `write()` ve `writeln()` yöntemlerinin kullanımı kısıtlı değildir. Ancak sayfa yüklendikten sonra, bu yöntemlerden birini çağırmak sayfayı temizlemez veya yeni bir sayfa oluşturmaz. Birçok web tarayıcısında olduğu gibi, bir sayfa yüklenmesini bitirdikten sonra uygulama dışı sanal alanda `document.write()` veya `writeln()` yöntemini çağırmak geçerli sayfayı temizler ve yeni, boş bir sayfa açar.

Document.designMode özelliği

Adobe AIR 1.0 ve üstü

Belgedeki tüm öğelerin düzenlenebilir olması için `document.designMode` özelliğini `on` olan bir değere ayarlayın. Yerleşik düzenleyici desteği metin düzenleme, kopyalama, yapıştırma ve sürükleyip bırakmayı içerir. `designMode` özelliğinin `on` olarak ayarlanması, `body` öğesinin `contentEditable` özelliğinin `true` olarak ayarlanmasına eşdeğerdir. Bir belgenin hangi bölümlerinin düzenlenebilir olacağını tanımlamak için çoğu HTML öğesinde `contentEditable` özelliğini kullanabilirsiniz. Ek bilgi için bkz. "HTML contentEditable niteliği" sayfa 985.

unload olayları (body ve frameset nesneleri için)

Adobe AIR 1.0 ve üstü

Bir pencerenin üst düzey `frameset` veya `body` etiketinde (uygulamanın ana penceresi de dahil olmak üzere), kapanan pencereye (veya uygulamaya) yanıt vermek için `unload` olayını kullanmayın. Bunun yerine NativeApplication nesnesinin `exiting` olayını kullanın (uygulamanın ne zaman kapandığını saptamak için). Veya NativeWindow nesnesinin `closing` olayını kullanın (pencerenin ne zaman kapandığını saptamak için). Örneğin aşağıdaki JavaScript kodu, kullanıcı uygulamayı kapattığında (`"Goodbye."`) mesajını görüntüler:

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

Ancak komut dosyaları, bir karenin, iframe'in veya üst düzey pencere içeriğinin gezinmesi sonucu oluşan `unload` olayına başarılı bir şekilde *yanıt verebilir*.

Not: *Adobe AIR'in gelecek sürümlerinde bu sınırlamalar kaldırılabilir.*

JavaScript Window nesnesi

Adobe AIR 1.0 ve üstü

Window nesnesi, JavaScript çalışma bağlamında global nesne olarak kalır. AIR, uygulama sanal alanında, önemli ana bilgisayar nesnelerinin yanı sıra yerleşik AIR sınıflarına erişim sağlamak için JavaScript Window nesnesine yeni özellikler ekler. Ayrıca, bazı yöntemler ve özellikler, uygulama sanal alanının içinde bulunup bulunmamalarına bağlı olarak farklı davranışlar gösterir.

Window.runtime özelliği `runtime` özelliği, yerleşik runtime sınıflarını uygulama sanal alanının içinden başlatmanızı ve kullanmanızı sağlar. Bu sınıflar AIR ve Flash Player API'lerini içerir. (Ancak örneğin Flex çerçevesini içermez.) Örneğin aşağıdaki ifade bir AIR file nesnesi oluşturur:

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

AIR SDK'de sağlanan `AIRAliases.js` dosyası, bu gibi başvuruları kısaltmanızı sağlayan başka ad tanımları içerir. Örneğin `AIRAliases.js` bir sayfaya içe aktarıldığında, aşağıdaki ifade kullanılarak bir File nesnesi oluşturulabilir:

```
var preferencesFile = new air.File();
```

`window.runtime` özelliği, yalnızca uygulama sanal alanındaki içerik ve frame veya iframe'ler içeren bir sayfanın üst belgesi için tanımlanır.

Bkz. "AIRAliases.js dosyasını kullanma" sayfa 998.

Window.nativeWindow özelliği `nativeWindow` özelliği, alttaki yerel window nesnesine bir başvuru sağlar. Bu özellik sayesinde ekran konumu, boyutu ve görünürlüğü gibi window işlevlerini ve özelliklerini komut dosyasına yazabilir ve kapatma, yeniden boyutlandırma ve taşıma gibi window olaylarını işleyebilirsiniz. Örneğin aşağıdaki ifade pencereyi kapatır:

```
window.nativeWindow.close();
```

Not: *NativeWindow nesnesi tarafından sağlanan pencere kontrol özellikleri, JavaScript Window nesnesi tarafından sağlanan özelliklerle örtüşür. Bu gibi durumlarda en uygun bulduğunuz yöntemi kullanabilirsiniz.*

`window.nativeWindow` özelliği, yalnızca uygulama sanal alanındaki içerik ve kareler veya iframe'ler içeren bir sayfanın üst belgesi için tanımlanır.

Window.htmlLoader özelliği `htmlLoader` özelliği, HTML içeriğine sahip olan AIR HTMLLoader nesnesine bir başvuru sağlar. Bu özellik sayesinde, HTML ortamının görünümünü ve davranışını komut dosyasına yazabilirsiniz. Örneğin kontrolün varsayılan, beyaz bir arka plan boyayıp boyamadığını belirlemek için `htmlLoader.paintsDefaultBackground` özelliğini kullanabilirsiniz:

```
window.htmlLoader.paintsDefaultBackground = false;
```

Not: *HTMLLoader nesnesi `window` özelliğine sahiptir, bu özellik, sahip olduğu HTML içeriğinin JavaScript Window nesnesine başvurur. Bu özelliği, içeriğe sahip olan HTMLLoader nesnesine bir başvuru yoluyla JavaScript ortamına erişmek için kullanabilirsiniz.*

`window.htmlLoader` özelliği, yalnızca uygulama sanal alanındaki içerik ve kareler veya iframe'ler içeren bir sayfanın üst belgesi için tanımlanır.

Window.parentSandboxBridge ve Window.childSandboxBridge özellikleri `parentSandboxBridge` ve `childSandboxBridge` özellikleri sayesinde, üst ve alt kareler arasında bir arabirim tanımlayabilirsiniz. Daha fazla bilgi için bkz. "Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği" sayfa 1008.

Window.setTimeout() ve Window.setInterval() işlevleri AIR, `setTimeout()` ve `setInterval()` işlevlerinin uygulama sanal alanında kullanılmasına yönelik güvenlik kısıtlamaları getirir. `setTimeout()` veya `setInterval()` işlevlerini çağırırken bir dize olarak çalıştırılacak kodu tanımlayamazsınız. Bir işlev başvurusu kullanmanız gerekir. Daha fazla bilgi için bkz. "setTimeout() ve setInterval()" sayfa 995.

Window.open() işlevi `open()` yöntemi uygulama dışı bir sanal alanda çalışan bir kod tarafından çağrıldığında, yalnızca kullanıcı etkileşiminin (fareyi tıklatma veya tuşabasma gibi) bir sonucu olarak çağrılırsa bir pencere açar. Ayrıca, uygulama başlığı pencere başlığına önek olarak getirilir (pencerelerin uzak içerik tarafından açılmasını engellemek için). Daha fazla bilgi için bkz."JavaScript window.open() yönteminin çağrılmasına ilişkin kısıtlamalar" sayfa 1097.

air.NativeApplication nesnesi

Adobe AIR 1.0 ve üstü

NativeApplication nesnesi uygulama durumu hakkında bilgi sağlar, uygulama düzeyi birçok önemli olayı gönderir ve uygulama davranışının kontrol edilmesine yönelik faydalı işlevler sağlar. NativeApplication nesnesinin tek bir örneği otomatik olarak oluşturulur ve bu özelliğe sınıf tanımlı `NativeApplication.nativeApplication` özelliği yardımıyla erişilebilir.

Nesneye JavaScript kodundan erişmek için şunu kullanabilirsiniz:

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

Veya `AIRAliases.js` komut dosyası içe aktarılmışsa, daha kısa biçimini kullanabilirsiniz:

```
var app = air.NativeApplication.nativeApplication;
```

NativeApplication nesnesine yalnızca uygulama sanal alanından erişilebilir. NativeApplication nesnesiyle ilgili daha fazla bilgi için bkz. "AIR çalışma zamanı ve işletim sistemi bilgileriyle çalışma" sayfa 897.

JavaScript URL şeması

Adobe AIR 1.0 ve üstü

Bir JavaScript URL şemasında tanımlanan kodu çalıştırma, (`href="javascript:alert('Test')"` gibi) uygulama sanal alanının içinde engellenmiştir. Hata verilmez.

AIR'de HTML

Adobe AIR 1.0 ve üstü

AIR ve WebKit, aşağıdakiler de dahil olmak üzere, standart olmayan birkaç HTML öğesi ve niteliği tanımlar:

"HTML frame ve iframe öğeleri" sayfa 983

"HTML öğesi olay işleyicileri" sayfa 985

HTML frame ve iframe öğeleri

Adobe AIR 1.0 ve üstü

AIR, uygulama sanal alanındaki içeriğin kare ve iframe öğelerine yeni nitelikler ekler:

sandboxRoot niteliği `sandboxRoot` niteliği, kare `src` niteliği tarafından belirtilen dosya için alternatif, uygulama dışı bir kaynak etki alanı belirtir. Dosya, belirtilen etki alanına karşılık gelen uygulama dışı sanal alana yüklenir. Dosyadaki içerik ve belirtilen etki alanından yüklenen içerik, birbirlerinin çapraz dosyasını oluşturabilir.

Önemli: *`sandboxRoot` değerini etki alanının temel URL'sine ayarlarsanız, bu etki alanından gelen tüm içeriğe yönelik istekler uzak sunucu yerine uygulama dizininden yüklenir (isteğin sayfa gezinmesinden mi, XMLHttpRequest öğesinden mi, yoksa başka içerik yükleme yollarından mı kaynaklandığına bakılmaksızın).*

documentRoot niteliği `documentRoot` niteliği, `sandboxRoot` tarafından belirtilen konumdaki dosyalara çözülen URL'lerin yükleneceği yerel dizini belirtir.

URL'leri çözerken, kare `src` niteliğinde veya kareye yüklenen içerikte, URL'nin `sandboxRoot` öğesinde belirtilen değerle eşleşen bölümü `documentRoot` öğesinde belirtilen değerle değiştirilir. Bu nedenle aşağıdaki kare etiketinde:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html`, uygulama yükleme klasörünün `sandbox` altdizininden yüklenir. `child.html` içerisindeki göreceli URL'ler, `sandbox` dizini temel alınarak çözülür. AIR, dosyaları app:/sandbox/ dizininden yükleme girişiminde bulunacağı için, `www.example.com/air` adresinde bulunan uzak sunucudaki tüm dosyalara kareden erişilemediğini unutmayın.

allowCrossDomainXHR niteliği Karedeki içeriğin herhangi bir uzak etki alanına XMLHttpRequest'leri yapmasını sağlamak için, `allowCrossDomainXHR="allowCrossDomainXHR"` öğesini açılış kare etiketine dahil edin. Varsayılan olarak uygulama dışı içerik, yalnızca kendi kaynak etki alanından bu tür isteklerde bulunabilir. Bunlar etki alanları arası XHR'lere izin vermekten doğan ciddi güvenlik sonuçlarıdır. Sayfadaki kod, herhangi bir etki alanıyla veri alışverişi yapabilir. Kötü amaçlı içerik bir şekilde sayfaya dahil olursa, geçerli sanal alanda kodun erişimi olan tüm veriler tehlikeye girebilir. Etki alanları arası XHR'leri yalnızca oluşturduğunuz ve kontrol ettiğiniz sayfalarda ve yalnızca etki alanları arası veri yükleme gerçekten gerekli olduğunda etkinleştirin. Ayrıca, kod dahil etmeyi veya başka yollarla yapılan saldırıları önlemek için, sayfa tarafından yüklenen tüm harici verileri dikkatli bir şekilde doğrulayın.

Önemli: *`allowCrossDomainXHR` niteliği bir kare veya iframe öğesine dahil edilmişse, etki alanları arası XHR'ler etkindir (atanan değer "0" olmadığı veya "f" ya da "n" harfleriyle başlamadığı sürece). Örneğin, `allowCrossDomainXHR` değerini `"deny"` olarak ayarlamak da etki alanları arası XHR'leri etkinleştirir. Etki alanları arası istekleri etkinleştirmek istemiyorsanız, niteliği tamamen öğe bildiriminin dışında bırakın.*

ondominitialize niteliği Bir karenin `dominitialize` olayının olay işleyicisini belirtir. Bu olay, karenin pencere ve belge nesneleri oluşturulduğunda ancak herhangi bir komut dosyası ayrıştırılmadan veya belge öğeleri oluşturulmadan önce çalışan, AIR'ye özgü bir olaydır.

Kare, `dominitialize` olayını yükleme dizisinde, alt sayfadaki komut dosyaları alt belgeye `dominitialize` işleyicisi tarafından eklenen nesnelere, değişkenlere ve işlevlere başvurmadan önce, yeterince erken bir zamanda gönderir. Bir alt belgeye doğrudan nesne eklemek veya alt belgedeki nesnelere doğrudan erişmek için üst sayfayla alt sayfa aynı sanal alanda bulunmalıdır. Ancak uygulama sanal alanındaki bir üst öğe, uygulama dışı sanal alandaki içerikle iletişim kurmak için sanal alan köprüsü kurabilir.

Aşağıdaki örnekler iframe etiketinin AIR'deki kullanımını gösterir:

`child.html` öğesini, uzak sunucudaki gerçek bir etki alanına eşlemeden uzak sanal alana yerleştirin:

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

`child.html` öğesini, XMLHttpRequest'lerinin yalnızca `www.example.com` adresine yapılmasına izin vererek uzak sanal alana yerleştirin:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

`child.html` öğesini, XMLHttpRequest'lerinin herhangi bir uzak etki alanına yapılmasına izin vererek uzak sanal alana yerleştirin:

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

`child.html` öğesini dosya sistemiyle yerel sanal alana yerleştirin:

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

`child.html` öğesini, sanal alan köprüsü kurmak için `dominitialize` olayını kullanarak uzak sanal alana yerleştirin:

```
<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>
```

Aşağıdaki `child.html` belgesi, alt içeriğin üst sanal alan köprüsüne nasıl erişebileceğini gösterir:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Daha fazla bilgi için, bkz. "Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği" sayfa 1008 ve "Adobe AIR'de HTML güvenliği" sayfa 1092.

HTML öğesi olay işleyicileri

Adobe AIR 1.0 ve üstü

AIR ve Webkit'teki DOM nesneleri, standart DOM olayı modelinde bulunmayan bazı olayları gönderir. Aşağıdaki tablo, bu olaylara işleyici belirtmek için kullanabileceğiniz ilgili olay niteliklerini listeler.

Callback nitelik adı	Açıklama
oncontextmenu	Seçili metinde sağ tıklatılarak veya komut tuşuna basıp tıklatılarak bir bağlam menüsü çağrıldığında çağrılır.
oncopy	Bir öğedeki seçim kopyalandığında çağrılır.
oncut	Bir öğedeki seçim kesildiğinde çağrılır.
ondominitialize	Bir kareye veya iframe'e yüklenen belgenin DOM'si oluşturulduğunda, ancak bir DOM öğesi oluşturulmadan veya komut dosyaları ayrıştırılmadan çağrılır.
ondrag	Bir öğe sürüklendiğinde çağrılır.
ondragend	Sürükleme işlemi bırakıldığında çağrılır.
ondragenter	Sürükleme hareketi bir öğenin sınırlarına girdiğinde çağrılır.
ondragleave	Sürükleme hareketi bir öğenin sınırlarından çıktığında çağrılır.
ondragover	Sürükleme hareketi bir öğenin sınırları içerisindeyken sürekli çağrılır.
ondragstart	Sürükleme hareketi başladığında çağrılır.
ondrop	Bir öğenin üzerindeyken sürükleme hareketi bırakıldığında çağrılır.
onerror	Bir öğe yüklenirken hata oluştuğunda çağrılır.
oninput	Bir öğe biçimine metin girildiğinde çağrılır.
onpaste	Öğe, bir öğeye yapıştırıldığında çağrılır.
onscroll	Kaydırılabilir öğenin içeriği kaydırıldığında çağrılır.
onselectstart	Bir seçim başladığında çağrılır.

HTML contentEditable niteliği

Adobe AIR 1.0 ve üstü

Kullanıcıların öğenin içeriğini düzenlemelerine olanak sağlamak için, `contentEditable` niteliğini herhangi bir HTML öğesine ekleyebilirsiniz. Örneğin aşağıdaki örnek HTML kodu, ilk `p` öğe dışında belgenin tamamını düzenlenebilir olarak ayarlar:

```
<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>
```

Not: *`document.designMode` özelliğini `on` olarak ayarlarsanız, tek bir öğe için `contentEditable` ayarına bakılmaksızın belgedeki tüm öğeler düzenlenebilir hale gelir. Ancak `designMode` özelliğini `off` olarak ayarlamak, `contentEditable` değerinin `true` olduğu öğelerin düzenlenme özelliğini devre dışı bırakmaz. Ek bilgi için bkz.* *"Document.designMode özelliği" sayfa 980.*

Veri: URL'ler

Adobe AIR 2 ve üstü

AIR aşağıdaki öğeler için `data:` URL'lerini destekler:

- img
- input type="image"
- Görüntülere (arka plan görüntüsü gibi) izin veren CSS kuralları

Data URL'leri bir CSS veya HTML belgesine ikili görüntü verilerini base64 kodlu dize olarak doğrudan eklemenize imkan verir. Aşağıdaki örnekte, tekrarlayan arka plan olarak bir data: URL kullanılmaktadır:

```
<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>
```

data: URL'lerini kullanırken, fazladan bir boşluk gerektiğini unutmayın. Örneğin, veri dizesi tek ve aralıksız bir satır olarak girilmelidir. Aksi takdirde, satır kesmeleri verinin parçası olarak görülür ve görüntünün kodu çözülemez.

AIR'de CSS

Adobe AIR 1.0 ve üstü

WebKit birçok genişletilmiş CSS özelliğini destekler. Bu uzantıların birçoğu `-webkit` önekini kullanır. Bu uzantılardan bazılarının deneysel nitelikte olduğunu ve WebKit'in daha sonraki bir sürümünden kaldırılabileceğini unutmayın. Webkit'in CSS desteği ve CSS uzantıları hakkında daha fazla bilgi edinmek için bkz. Safari CSS Reference (Safari CSS Başvurusu).

AIR'de desteklenmeyen WebKit özellikleri

Adobe AIR 1.0 ve üstü

AIR, WebKit veya Safari 4'ün aşağıdaki özelliklerini desteklemez:

- window.postMessage üzerinden etki alanları arası mesajlaşma (AIR kendi etki alanları arası iletişim API'lerini sağlar)
- CSS değişkenleri
- Web Open Font Format (WOFF) ve SVG fontları.
- HTML video ve ses etiketleri
- Medya aygıtı sorguları
- Çevrimdışı uygulama önbelleği
- Yazdırılıyor (AIR kendi PrintJob API'sini sağlar)
- Yazım ve dilbilgisi denetimleri
- SVG
- WAI-ARIA
- Web Soketleri (AIR kendi soket API'lerini sağlar)
- Web eklentileri
- WebKit SQL API (AIR kendi API'sini sağlar)
- WebKit coğrafi konum belirleme API'si (AIR, desteklenen aygıtlar üzerinde kendi coğrafi konum belirleme API'sini sağlar)
- WebKit çoklu dosya yükleme API'si
- WebKit dokunmatik olayları (AIR kendi dokunmatik olaylarını sağlar)
- Kablosuz Biçimlendirme Dili (WML)

Aşağıdaki listede AIR tarafından desteklenmeyen JavaScript API'leri, HTML öğeleri ve CSS özellik ve değerleri yer almaktadır:

Desteklenmeyen JavaScript Window nesnesi üyeleri:

- applicationCache()
- console
- openDatabase()
- postMessage()
- document.print()

Desteklenmeyen HTML etiketleri:

- audio
- video

Desteklenmeyen HTML nitelikleri:

- aria-*
- draggable
- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange
- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove
- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

Desteklenmeyen JavaScript olayları:

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove
- touchend
- touchcancel
- webkitbeginfullscreen

- webkitendfullscreen

Desteklenmeyen CSS özellikleri:

- background-clip
- background-origin (-webkit-background-origin kullanın)
- background-repeat-x
- background-repeat-y
- background-size (-webkit-background-size kullanın)
- 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

Desteklenmeyen CSS değerleri:

- görünüm özelliği değerleri:
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button
- border-box (background-clip ve background-origin)
- contain (background-size)
- content-box (background-clip ve background-origin)
- cover (background-size)
- liste özellik değerleri:
  - 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 (görüntüleme özelliği)

Bölüm 59: AIR uygulamasında HTML ve JavaScript programlama

Adobe AIR 1.0 ve üstü

Adobe® AIR® uygulamalarını HTML ve JavaScript ile geliştirmeye özgü bir dizi programlama başlığı mevcuttur. HTML tabanlı bir AIR uygulaması veya HTMLLoader sınıfını (veya mx:HTML Flex™ bileşenini) kullanarak HTML veya JavaScript çalıştıran SWF tabanlı bir AIR uygulamasını programladığınızda, aşağıdaki bilgiler önemlidir.

HTMLLoader sınıfı hakkında

Adobe AIR 1.0 ve üstü

Adobe AIR HTMLLoader sınıfı, AIR uygulamasında HTML içeriğini görüntüleyebilen görüntüleme nesnesini tanımlar. SWF tabanlı uygulamalar, var olan bir pencereye HTMLLoader denetimi ekleyebilir veya `HTMLLoader.createRootWindow()` öğesine sahip HTMLLoader nesnesini otomatik olarak içeren bir HTML penceresi oluşturabilir. HTMLLoader nesnesine yüklenen HTML sayfasından JavaScript `window.htmlLoader` özelliği aracılığıyla erişilebilir.

URL'den HTML içeriği yükleme

Adobe AIR 1.0 ve üstü

Aşağıdaki kod, HTMLLoader nesnesine bir URL yükler (uygulamanızdaki HTML içeriğini görüntülemek için HTMLLoader nesnesini sahne veya diğer görüntü nesnelerinin alt öğesi olarak ekleyin):

```
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);
```

HTMLLoader nesnesinin `width` ve `height` özellikleri varsayılan olarak 0'a ayarlanmıştır. Sahneye HTMLLoader nesnesi eklerken bu boyutları ayarlamak isteyeceksiniz. HTMLLoader öğesi, sayfa yüklenirken birçok olay gönderir. Yüklenen sayfayla etkileşim kurmanın ne zaman güvenli olduğunu belirlemek için bu olauları kullanabilirsiniz. Bu olaylar, “AIR'de HTML bağlantılı olayları işleme” sayfa 1030 bölümünde anlatılmıştır.

Not: *Flex çerçevesinde, yalnızca UIComponent sınıfını geliştiren sınıflar Flex Container bileşenlerinin alt öğesi olarak eklenebilir. Bu nedenle, bir HTMLLoader öğesini doğrudan Flex Container bileşeninin alt öğesi olarak ekleyemezsiniz; ancak Flex mx:HTML denetimini kullanabilir, UIComponent bileşeninin alt öğesi olarak UIComponent öğesini geliştiren HTMLLoader öğesini içeren özel bir sınıf oluşturabilir veya HTMLLoader öğesini UIComponent öğesinin alt öğesi olarak ekleyip UIComponent öğesini Flex kabına ekleyebilirsiniz.*

Ayrıca TextField sınıfını kullanarak HTML metni oluşturabilirsiniz, ancak bunun özellikleri sınırlıdır. Adobe® Flash® Player'ın TextField sınıfı, HTML işaretleme alt kümesini destekler, ancak boyut sınırlamaları nedeniyle özellikleri sınırlıdır. (Adobe AIR'de bulunan HTMLLoader sınıfı Flash Player'da mevcut değildir.)

Bir dizeden HTML içeriğini yükleme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesinin `loadString()` yöntemi, HTML içeriğinin bir dizesini HTMLLoader nesnesine yükler:

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

`loadString()` yöntemi yoluyla yüklenen içerik varsayılan olarak aşağıdaki karakteristiklere sahip uygulama dışı bir sanal alanda konumlandırılır:

- Ağdan içerik yüklemek için erişime sahiptir (Ancak dosya sisteminden değil.)
- Verileri XMLHttpRequest kullanarak yükleyemez.
- `window.location` özelliği `"about:blank"` olarak ayarlanır.
- İçerik, `window.runtime` özelliğine (herhangi bir uygulama dışı sanal alanda yer alan içeriğin erişebildiği biçimde) erişemez.

AIR 1.5'te, HTMLLoader sınıfı bir `placeLoadStringContentInApplicationSandbox` özelliği içerir. Bu özellik HTMLLoader nesnesi için `true` olarak ayarlandığında, `loadString()` yöntemiyle yüklenen içerik uygulama sanal alanında konumlandırılır. (Varsayılan değer, `false` değeridir.) Bu, `loadString()` yöntemiyle yüklenen içerik için `window.runtime` özelliğine ve diğer tüm AIR API'lerine erişim sağlar. Bu özelliği `true` olarak ayarladığınızda, `loadString()` yöntemine yapılan çağrıda kullanılan dizeye ilişkin veri kaynağının güvenilir olduğundan emin olun. Bu özellik `true` olarak ayarlandığında, HTML dizesindeki kod ifadeleri tam uygulama ayrıcalıklarıyla yürütülür. Bu özelliği yalnızca dizenin zararlı kodlar içermediğinden emin olduğunuzda `true` olarak ayarlayın.

AIR 1.0 veya AIR 1.1 SDK'larıyla derlenen uygulamalarda, `loadString()` yöntemiyle yüklenen içerik uygulama sanal alanında konumlandırılır.

HTML ve AIR uygulamalarının kullanımında önemli güvenlik kuralları

Adobe AIR 1.0 ve üstü

AIR uygulamasıyla kurduğunuz dosyalar AIR API'lerine erişime sahiptir. Güvenlik nedenleriyle, diğer kaynaklardan gelen içerik bu erişime sahip değildir. Örneğin bu kısıtlama uzak bir etki alanından (http://ornek.com gibi) gelen içeriğin kullanıcının masaüstü dizini içeriğini okumasını (veya daha kötüsünü) engeller.

Çünkü `eval()` işlevinin (ve ilgili API'lerin) çağrılması yoluyla yararlanılabilecek güvenlik boşlukları mevcuttur, uygulamayla yüklenen içerik, varsayılan olarak bu yöntemleri kullanamaz. Ancak, bazı Ajax çerçeveleri `eval()` işlevini ve ilgili API'leri çağırma yolunu kullanır.

İçeriği bir AIR uygulamasında çalışmak üzere düzgün biçimde yapılandırmak için farklı kaynaklardan gelen içerikteki güvenlik kısıtlamalarına ilişkin kuralları dikkate almalısınız. Farklı kaynaklardan gelen içerik, sanal alan adı verilen ayrı güvenlik sınıflandırmalarına yerleştirilir (bkz. "Güvenlik sanal alanları" sayfa 1055). Varsayılan olarak, uygulamayla yüklenen içerik *uygulama* sanal alanı olarak bilinen bir sanal alana yüklenir ve bu sayede AIR API'lerine erişebilir. Uygulama sanal alanı güvenilmeyen kodların yürütülmesini engellemek üzere tasarlanmış kısıtlamalar içerdiğinden, genellikle en güvenli sanal alandır.

Çalışma zamanı, uygulamanızla yüklenen içeriği uygulama sanal alanından farklı bir sanal alana yüklemenize olanak verir. Uygulama dışı sanal alanlardaki içerik, tipik web tarayıcılarınınkine benzer bir güvenlik ortamında çalışır. Örneğin, uygulama dışı sanal alanlardaki kod `eval()` yöntemini ve ilgili yöntemleri kullanabilir. (Ancak aynı zamanda AIR API'lerine erişme izni yoktur.) Çalışma zamanı, içeriğin farklı sanal alanlarda güvenli biçimde iletişim kurmasına yönelik yollar içerir (örneğin, AIR API'lerini uygulama dışı içeriğe göstermeden). Ayrıntılar için bkz. "Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği" sayfa 1008.

Güvenlik nedenleriyle sanal alanda kullanımı kısıtlanmış olan kodu çağırdığınızda çalıştırma zamanı bir JavaScript hatası gönderir: "Uygulama güvenlik sanal alanında JavaScript koduna ilişkin Adobe AIR çalıştırma zamanı güvenlik ihlali."

Bu hatayı önlemek için sonraki "Güvenlikle ilişkili JavaScript hatalarını önleme" sayfa 993 bölümünde anlatılan kodlama uygulamalarını takip edin.

Daha fazla bilgi için, bkz. "Adobe AIR'de HTML güvenliği" sayfa 1092.

Güvenlikle ilişkili JavaScript hatalarını önleme

Adobe AIR 1.0 ve üstü

Bu güvenlik kısıtlamaları nedeniyle sanal alanda kullanımı kısıtlanmış olan kodu çağırdığınızda çalıştırma zamanı bir JavaScript hatası gönderir: "Uygulama güvenlik sanal alanında JavaScript koduna ilişkin Adobe AIR çalıştırma zamanı güvenlik ihlali." Bu hatayı önlemek için, bu kodlama uygulamalarını takip edin.

Güvenlik ilişkili JavaScript hatalarının nedenleri

Adobe AIR 1.0 ve üstü

Belge `load` olayı tetiklendiğinde ve herhangi bir `load` olay işleyicisinden çıkıldığında, uygulama sanal alanında yürütülen kod, değerlendirme ve yürütme dizeleri içeren çoğu uygulamadan kısıtlanır. Güvenli olmayabilecek dizeleri değerlendiren ve yürüten aşağıdaki JavaScript ifade türlerinin kullanılmaya çalışılması JavaScript hataları oluşturur:

- eval() işlevi
- setTimeout() ve setInterval()
- İşlev yapıcısı

  Ayrıca, aşağıdaki JavaScript ifade türleri güvenli olmayan JavaScript hatası oluşturmadan başarısız olur:

- javascript: URL'ler
- innerHTML ve outerHTML ifadelerinde onevent nitelikleri aracılığıyla atanan olay geri çağrıları
- JavaScript dosyalarını uygulama yükleme dizini dışından yükleme
- document.write() ve document.writeln()
- load olayından önce veya load olay işleyicisi sırasında senkronize XMLHttpRequests
- Dinamik olarak oluşturulan komut dosyası öğeleri

  Not: *Bazı sınırlı durumlarda, dizelerin değerlendirilmesine izin verilir. Daha fazla bilgi için bkz. "Farklı sanal alanlardaki içerik için kod kısıtlamaları" sayfa 1094.*

  Adobe http://www.adobe.com/go/airappsandboxframeworks_tr adreslerinde uygulama güvenlik sanal alanını desteklemek üzere bilinen bir dizi Ajax çerçevesi tutar.

Aşağıdaki bölümler, güvenli olmayan JavaScript hatalarını ve uygulama sanal alanında çalışan koda ilişkin sessiz hataları önlemek için komut dosyalarının nasıl yeniden yazılacağını anlatır.

Uygulama içeriğini farklı bir sanal alana eşleme

Adobe AIR 1.0 ve üstü

Çoğu durumda, güvenlik ilişkili JavaScript hatalarını önlemek amacıyla bir uygulamayı yeniden yazabilir veya yeniden yapılandırabilirsiniz. Ancak, yeniden yazma veya yeniden yapılandırma olanaklı olmadığında uygulama içeriğini "Uygulama içeriğini uygulama dışı sanal alana yükleme" sayfa 1009 bölümünde anlatılan tekniği kullanarak farklı bir sanal alana yükleyebilirsiniz. Bu içeriğin AIR API'lerine de erişmesi gerekiyorsa, "Sanal alan köprü arabirimini kurma" sayfa 1010 bölümünde anlatıldığı biçimde bir sanal alan köprüsü oluşturabilirsiniz.

eval() işlevi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Uygulama sanal alanında, `eval()` işlevi yalnızca `load` olayı sayfasından önce veya `load` olay işleyicisi sırasında kullanılabilir. Sayfa yüklendikten sonra, `eval()` işlevine yapılan çağrılar kodu yürütmez. Ancak, aşağıdaki durumlarda `eval()` işlevinin kullanımını önlemek için kodunuzu yeniden yazabilirsiniz.

Nesneye özellikler atama

Adobe AIR 1.0 ve üstü

Özellik erişimcisi oluşturmak için dizeyi ayrıştırmak yerine:

```
eval("obj." + propName + " = " + val);
```

köşeli parantez açıklaması içeren access özellikleri:

```
obj[propName] = val;
```

Bağlamda var olan değişkenler içeren işlev oluşturma

Adobe AIR 1.0 ve üstü

İfadeleri aşağıdaki şekilde değiştirin:

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

şununla:

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

Sınıf adını dize parametresi olarak kullanarak nesne oluşturma

Adobe AIR 1.0 ve üstü

Aşağıdaki kodla tanımlanan varsayımsal bir JavaScript sınıfı düşünün:

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

Örnek oluşturmanın en basit yolu `eval()` işlevini kullanmaktır:

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

Ancak, sınıf adının her bileşenini ayrıştırma ve köşeli parantez açıklaması kullanarak yeni nesne oluşturma yoluyla, `eval()` işlevinin çağrılmasını önleyebilirsiniz:

```
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;
}
```

Örneği oluşturmak için şunu kullanın:

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() ve setInterval()

Adobe AIR 1.0 ve üstü

İşleyici işlevi olarak onaylanan dizeyi bir işlev başvurusu veya nesneyle değiştirin. Örneğin, ifadeyi şu şekilde değiştirin:

```
setTimeout("alert('Timeout')", 100);
```

şununla:

```
setTimeout(function(){alert('Timeout')}, 100);
```

Veya işlev `this` nesnesinin çağıran tarafından ayarlanmasını gerektirdiğinde ifadeyi şu şekilde değiştirin:

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

şununla:

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

İşlev yapıcısı

Adobe AIR 1.0 ve üstü

`new Function(param, body)` çağrıları bir satır içi işlev bildirimiyle değiştirilebilir veya yalnızca sayfa `load` olayı işlenmeden önce kullanılabilir.

javascript: URL'ler

Adobe AIR 1.0 ve üstü

javascript kullanılarak linkte tanımlanan kod: Uygulama sanal alanında URL şeması yoksayılır. Güvenli olmayan JavaScript hatası oluşmadı. javascript aracılığıyla linkleri değiştirebilirsiniz: URL'ler, örneğin:

```
<a href="javascript:code()">Click Me</a>
```

şununla:

```
<a href="#" onclick="code()">Click Me</a>
```

innerHTML ve outerHTML ifadelerinde onevent nitelikleri aracılığıyla atanan olay geri çağrıları

Adobe AIR 1.0 ve üstü

Belge DOM'una öğe eklemek için innerHTML veya outerHTML kullandığınızda, ifadede atanan `onclick` veya `onmouseover` gibi olay geri çağrıları yoksayılır. Güvenlik hatası oluşmadı. Bunun yerine, `addEventListener()` yöntemini kullanarak yeni öğelere `id` niteliği atayabilir ve olay işleyici geri çağrı işlevlerini ayarlayabilirsiniz.

Örneğin, belgede şu şekilde bir hedef öğe belirlendiğinde:

```
<div id="container"></div>
```

İfadeleri şu şekilde değiştirin:

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

şununla:

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

JavaScript dosyalarını uygulama yükleme dizini dışından yükleme

Adobe AIR 1.0 ve üstü

Uygulama sanal alanının dışından komut dosyaları yüklemeye izin verilmez. Güvenlik hatası oluşmadı. Uygulama sanal alanında çalışan tüm komut dosyaları uygulama dizinine yüklenmiş olmalıdır. Sayfada harici komut dosyaları kullanmak için, sayfayı farklı bir sanal alana eşlemelisiniz. Bkz. "Uygulama içeriğini uygulama dışı sanal alana yükleme" sayfa 1009.

document.write() ve document.writeln()

Adobe AIR 1.0 ve üstü

Sayfa `load` olayı işlendikten sonra `document.write()` veya `document.writeln()` çağrıları yoksayılır. Güvenlik hatası oluşmadı. Alternatif olarak yeni bir dosya yükleyebilir veya DOM işleme tekniklerini kullanarak belge gövdesini değiştirebilirsiniz.

load olayından önce veya load olay işleyicisi sırasında senkronize XMLHttpRequests

Adobe AIR 1.0 ve üstü

`load` olayından önce veya `load` olay işleyicisi sırasında başlatılan senkronize XMLHttpRequest'ler herhangi bir içerik döndürmez. Senkronize olmayan XMLHttpRequest'ler başlatılabilir, ancak `load` olayının sonrasına kadar dönmez. `load` olayı işlendikten sonra senkronize XMLHttpRequest'ler normal davranır.

Dinamik olarak oluşturulan komut dosyası öğeleri

Adobe AIR 1.0 ve üstü

Dinamik olarak oluşturulan komut dosyası öğeleri, örneğin innerHTML ile veya`document.createElement()` yöntemiyle oluşturulduklarında yoksayılır.

AIR API sınıflarına JavaScript'ten erişme

Adobe AIR 1.0 ve üstü

Standart ve gelişmiş Webkit öğelerine ek olarak, HTML ve JavaScript kodu da çalışma zamanı tarafından sağlanan host sınıflarına erişebilir. Bu sınıflar AIR'in sağladığı gelişmiş özelliklere erişmenizi sağlar, bu özellikler:

- Dosya sistemine erişim
- Yerel SQL veri tabanları kullanımı
- Uygulama ve pencere menülerinin denetimi
- Ağ iletişimi için soketlere erişim
- Kullanıcı tanımlı sınıfların ve nesnelerin kullanımı
- Ses özellikleri

Örneğin, AIR dosya API'si flash.filesystem paketinde bulunan bir File sınıfını içerir. JavaScript'te aşağıdaki şekilde bir File nesnesi oluşturabilirsiniz:

```
var myFile = new window.runtime.flash.filesystem.File();
```

`runtime` nesnesi, uygulama sanal alanındaki AIR'de çalışan HTML içeriği tarafından kullanılabilen özel bir JavaScript nesnesidir. JavaScript'ten runtime sınıflarına erişebilmenizi sağlar. `runtime` nesnesinin `flash` özelliği, flash paketine erişim sağlar. Buna karşılık, `runtime` nesnesinin `flash.filesystem` özelliği flash.filesystem paketine erişim sağlar (ve bu paket File sınıfını içerir). Paketler, ActionScript'te kullanılan sınıfları düzenlemenin bir yoludur.

Not: *`runtime` özelliği kareye veya iframe'e yüklenen sayfaların window nesnelerine otomatik olarak eklenmez. Ancak, alt belge uygulama sanal alanında olduğu sürece, alt öğe üst öğenin `runtime` özelliğine erişebilir.*

Runtime sınıflarının paket yapısı geliştiricilerin her sınıfa erişmek için uzun JavaScript kod dizelerini yazmasını gerektirdiğinden (`window.runtime.flash.desktop.NativeApplication` içinde olduğu gibi), AIR SDK runtime sınıflarına çok daha kolay erişmenizi sağlayan AIRAliases.js dosyasını içerir (örneğin yalnızca `air.NativeApplication` yazarak).

Bu kılavuzda AIR API sınıfları tartışılmıştır. HTML geliştiricilerini ilgilendirebilecek Flash Player API dışındaki sınıflar *HTML Geliştiricileri için Adobe AIR API Başvurusu* bölümünde anlatılmıştır. ActionScript, SWF (Flash Player) içeriğinde kullanılan dildir. Ancak, JavaScript ve ActionScript sözdizimleri birbirine benzer. (ECMAScript dilinin sürümlerini temel alır.) Tüm yerleşik sınıflar hem JavaScript'te (HTML içeriğinde) hem de ActionScript'te (SWF içeriğinde) mevcuttur.

Not: *JavaScript kodu ActionScript'te bulunan Dictionary, XML, and XMLList sınıflarını kullanamaz.*

AIRAliases.js dosyasını kullanma

Adobe AIR 1.0 ve üstü

Runtime sınıfları, paket yapısında aşağıdaki şekilde düzenlenmiştir:

- `window.runtime.flash.desktop.NativeApplication`
- `window.runtime.flash.desktop.ClipboardManager`
- `window.runtime.flash.filesystem.FileStream`
- `window.runtime.flash.data.SQLDatabase`

AIR SDK içinde, runtime sınıflarına daha az yazı yazarak erişebilmenizi sağlayacak “başka ad” tanımlarını sunan bir AIRAliases.js dosyası yer alır. Örneğin, yukarıda listelenen sınıflara yalnızca şunu yazarak erişebilirsiniz:

- `air.NativeApplication`
- `air.Clipboard`
- `air.FileStream`
- `air.SQLDatabase`

Bu liste, AIRAliases.js dosyasındaki sınıfların kısa bir alt kümesidir. Sınıfların ve paket düzeyindeki işlevlerin tam listesi *HTML Geliştiricileri için Adobe AIR API Başvurusu* içinde bulunur.

AIRAliases.js dosyası, yaygın olarak kullanılan runtime sınıflarının yanı sıra, yaygın olarak kullanılan paket düzeyindeki işlevlere işlevler için başka adları da içerir: `air.trace()`, `air.navigateToURL()` ve `air.sendToURL()` olarak farklı adlandırılan `window.runtime.trace()`, `window.runtime.flash.net.navigateToURL()` ve `window.runtime.flash.net.sendToURL()`.

AIRAliases.js dosyasını kullanmak için aşağıdaki `script` başvurusunu HTML sayfanıza ekleyin:

```
<script src="AIRAliases.js"></script>
```

Gerektiğinde `src` başvurusundaki yolu ayarlayın.

Önemli: *Bu belgedeki JavaScript örnek kodu, aksi belirtilmediği sürece AIRAliases.js dosyasını HTML sayfanıza dahil ettiğinizi varsayar.*

AIR'deki URL'ler hakkında

Adobe AIR 1.0 ve üstü

AIR'den çalışan HTML içeriğinde, `link` etiketinin `href` niteliğinde veya URL girebileceğiniz herhangi bir yerde `img`, `frame`, `iframe` ve `script` etiketlerine ilişkin `src` niteliklerini tanımlamak için aşağıdaki URL şemalarından birini kullanabilirsiniz.

URL şeması	Açıklama	Örnek
file	Dosya sisteminin köküne ilişkin yol.	`file:///c:/AIR Test/test.txt`
app	Yüklenen uygulamanın kök dizinine ilişkin yol.	`app:/images`
app-storage	Uygulama depolama dizinine ilişkin yol. AIR her yüklenen uygulama için, o uygulamaya özgü verilerin saklanmasına yönelik kullanışlı bir yer olan benzersiz bir uygulama depolama dizini tanımlar.	`app-storage:/settings/prefs.xml`
http	Standart bir HTTP isteği.	`http://www.adobe.com`
https	Standart bir HTTPS isteği.	`https://secure.example.com`

URL şemalarının AIR'de kullanımı hakkında daha fazla bilgi için bkz. “URI şemaları” sayfa 822.

File, Loader, URLStream ve Sound sınıfları dahil çoğu AIR API'leri URL içeren bir dize yerine URLRequest nesnesini kullanır. URLRequest nesnesi, aynı url şemalarından herhangi birini kullanabilen bir dizeyle başlatılır. Örneğin, aşağıdaki ifade Adobe ana sayfasını istemek için kullanılabilecek bir URLRequest nesnesi oluşturur.

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

URLRequest nesneleri hakkında bilgi için bkz. “HTTP iletişimi” sayfa 820.

ActionScript nesnelerini JavaScript için kullanılabilir duruma getirme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi tarafından yüklenen HTML sayfasındaki JavaScript, ActionScript yürütme bağlamında HTML sayfasının `window.runtime`, `window.htmlLoader` ve `window.nativeWindow` özellikleri kullanılarak tanımlanan sınıfları, nesneleri ve işlevleri çağırabilir. Ayrıca JavaScript yürütme bağlamında ActionScript nesneleri ve işlevlerine başvurular oluşturarak bunları JavaScript için kullanılabilir duruma getirebilirsiniz.

JavaScript nesnelerine ActionScript üzerinden erişmenin basit bir örneği

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, ActionScript nesnelerine başvuru yapan özelliklerin bir HTML'sayfasının global window nesnesine nasıl eklendiğini gösterir:

```
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;
}
```

Önceki örnekte HTMLLoader nesnesine yüklenen HTML içeriği (test.html adlı bir dosyada) üst SWF dosyasında tanımlanan `foo` özelliği ve `helloFromJS()` yöntemine erişebilir:

```
<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>
```

Yüklenen bir belgenin JavaScript bağlamına erişirken, sayfa yapım sırasında nesneleri sayfada tanımlanan tüm komut dosyalarının erişebileceği kadar erken oluşturmak için `htmlDOMInitialize` olayını kullanabilirsiniz. `Complete` olayını beklerseniz, yalnızca sayfa `load` olayından sonra çalışan sayfadaki komut dosyaları eklenen nesnelere erişebilir.

Sınıf tanımlarını JavaScript için kullanılabilir duruma getirme

Adobe AIR 1.0 ve üstü

Uygulamanızın ActionScript sınıflarını JavaScript'te kullanılabilir duruma getirmek için, yüklenen HTML içeriğini sınıf tanımlarını içeren uygulama etki alanına atayabilirsiniz. JavaScript yürütme bağlamının uygulama etki alanı, HTMLLoader nesnesinin `runtimeApplicationDomain` özelliğiyle ayarlanabilir. Uygulama etki alanını birincil uygulama etki alanı olarak ayarlamak için `runtimeApplicationDomain` öğesini aşağıdaki kodda gösterildiği şekilde `ApplicationDomain.currentDomain` olarak ayarlayın:

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

`runtimeApplicationDomain` özelliği ayarlandığında, JavaScript bağlamı sınıf tanımlarını atanan etki alanıyla paylaşır. JavaScript'teki özel bir sınıfın örneğini oluşturmak için `window.runtime` özelliği aracılığıyla sınıf tanımına başvuru yapın ve `new` operatörü kullanın:

```
var customClassObject = new window.runtime.CustomClass();
```

HTML içeriği uyumlu bir güvenlik etki alanından geliyor olmalıdır. HTML içeriği atadığınız uygulama etki alanından farklı bir güvenlik etki alaından geliyorsa, sayfa bunun yerine varsayılan uygulama etki alanını kullanır. Örneğin, İnternet'ten uzak bir sayfa yüklediğinizde ApplicationDomain.currentDomain öğesini sayfanın uygulama etki alanı olarak atayamazsınız.

Olay dinleyicilerini kaldırma

Adobe AIR 1.0 ve üstü

Runtime nesneleri, yüklenen SWF içeriğindeki nesneler ve hatta diğer sayfalarda çalışan JavaScript nesneleri de dahil güncel sayfa dışındaki nesnelere JavaScript olay dinleyicileri eklediğinizde, sayfa boşaldığında her zaman bu olay dinleyicilerini kaldırmalısınız. Aksi halde, olay dinleyicisi artık var olmayan bir işleyici işlevine olay gönderir. Bu gerçekleşirse, şu hata mesajını görürsünüz: “Uygulama artık yüklenmeyen bir HTML sayfasındaki JavaScript nesnesine başvuruda bulunmaya çalıştı." Gerekli olmayan olay dinleyicilerinin kaldırılması, AIR'in ilişkilendirilmiş belleği geri istemesine olanak verir. Daha fazla bilgi için bkz. “Gezinilen HTML sayfalarındaki olay dinleyicilerini kaldırma” sayfa 1035.

HTML DOM ve JavaScript nesnelerine ActionScript'ten erişme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi `complete` olayını gönderdikten sonra, sayfaya ilişkin HTML DOM'daki (belge nesnesi modeli) tüm nesnelere erişebilirsiniz. Erişilebilir nesneler görüntü öğelerininin (sayfadaki `div` ve `p` nesneleri gibi) yanı sıra JavaScript değişkenleri ve işlevlerini kapsar. `complete` olayı, JavaScript sayfası `load` olayına karşılık gelir. `complete` gönderilmeden önce DOM öğeleri, değişkenleri ve işlevleri ayrıştırılmamış veya oluşturulmamış olabilir. Mümkünse HTML DOM'a erişmeden önce `complete` olayını bekleyin.

Örneğin, aşağıdaki HTML sayfasına bakın:

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

Bu basit HTML sayfası *foo* adlı bir JavaScript değişkeni ve *test()* adlı bir JavaScript işlevi tanımlar. Bunların her ikisi de sayfanın genel `window` nesnesinin özellikleridir. Ayrıca, `window.document` nesnesi, `getElementById()` yöntemini kullanarak erişebileceğiniz, (*p1* kimliğiyle) adlandırılan P öğesini içerir. Sayfa yüklendiğinde (HTMLLoader nesnesi `complete` olayını gönderdiğinde), bu nesnelerin her birine aşağıdaki ActionScript kodunda gösterildiği şekilde ActionScript'ten erişebilirsiniz:

```
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.
}
```

HTML öğesinin içeriğine erişmek için `innerHTML` özelliğini kullanın. Örneğin, önceki kod *p1* adlı HTML öğesinin içeriğini getirmek için `html.window.document.getElementById("p1").innerHTML` kullanır.

HTML sayfasının özelliklerini ActionScript'ten de ayarlayabilirsiniz. Örneğin, aşağıdaki örnek var olan HTMLLoader nesnesine ilişkin başvuruyu kullanarak `p1` öğesinin içeriğini ve sayfadaki `foo` JavaScript değişkeninin değerini ayarlar:

```
html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

SWF içeriğini HTML'e gömme

Adobe AIR 1.0 ve üstü

SWF içeriğini tarayıcıda olduğu gibi AIR uygulamasında da HTML içeriğine gömebilirsiniz. SWF içeriğini `object` etiketini, `embed` etiketini veya her ikisini de kullanarak gömün.

Not: *SWF içeriğini HTML sayfasında görüntülemek için `object` etiketi ve `embed` etiketinin her ikisini de kullanmak, yaygın bir web geliştirme uygulamasıdır. Bu uygulamanın AIR'de hiçbir yararı yoktur. AIR'de görüntülenmesi için içerikteki W3C standart `object` etiketinin kendisini kullanabilirsiniz. Aynı zamanda, tarayıcıda da görüntülenen HTML içeriği için gerektiğinde `object` ve `embed` etiketlerinin ikisini birlikte kullanmaya devam edebilirsiniz.*

HTML ve SWF içeriğini görüntüleyen NativeWindow nesnesinde saydamlığı etkinleştirdiyseniz, içeriği gömmek için kullanılan pencere modu (`wmode`) `window` değerine ayarlandığında, AIR SWF içeriğini görüntülemez. Saydam bir pencerenin HTML sayfasında SWF içeriğini görüntülemek için `wmode` parametresini `opaque` veya `transparent` olarak ayarlayın. `window`, `wmode` için varsayılan değerdir, bu yüzden bir değer belirtmezseniz içeriğiniz görüntülenemeyebilir.

Aşağıdaki örnek, bir SWF dosyasını HTML içeriğinde görüntülemek için HTML `object` etiketinin kullanımını gösterir. `wmode` parametresi `opaque` olarak ayarlanmıştır, bu nedenle temeldeki NativeWindow nesnesi saydam olsa bile içerik görüntülenir. SWF dosyası uygulama dizininden yüklenir, ancak AIR tarafından desteklenen herhangi bir URL şemasını kullanabilirsiniz. (SWF dosyasının yüklendiği kaynak, AIR'in içeriği yerleştirdiği güvenlik sanal alanını belirler.)

```
<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>
```

İçeriği dinamik olarak yüklemek için bir komut dosyası da kullanabilirsiniz. Aşağıdaki örnek, `urlString` parametresinde belirtilen SWF dosyasını görüntülemek üzere bir `object` düğümü oluşturur. Örnek, düğümü `elementID` parametresiyle belirtilen kimliğe sahip sayfa öğesinin alt öğesi olarak ekler.

```
<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>
```

HTMLLoader nesnesi ölçeklenirse veya döndürülürse ya da `alpha` değeri 1.0'dan başka bir değere ayarlanırsa SWF içeriği görüntülenmez. AIR 1.5.2'den önce, SWF içeriği hangi `wmode` değeri ayarlı olursa olsun saydam pencerede görüntülenmezdi.

Not: *Gömülü bir SWF nesnesi video dosyası gibi harici bir varlık yüklemeyi denediğinde, HTML dosyasında video dosyasına kesin bir yol sağlanmamışsa SWF içeriği düzgün bir şekilde oluşturulamayabilir. Ancak, gömülü bir SWF nesnesi göreli bir yol kullanarak harici bir görüntü dosyası yükleyebilir.*

Aşağıdaki örnek, harici varlıkların HTML içeriğinde gömülü bulunan SWF nesnesi aracılığıyla nasıl yüklenebileceğini gösterir:

```
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");

}
```

Aşağıdaki ActionScript örneğinde, HTML dosyası tarafından iletilen görüntü yolu okunur ve görüntü sahne alanına yüklenir:

```
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;
                  }
            }
      }
}
```

HTML sayfasında ActionScript kütüphaneleri kullanma

Adobe AIR 1.0 ve üstü

AIR, sayfanın ActionScript sınıflarını derlenmiş bir SWF dosyasına yükleyebilmesi için HTML komut dosyası öğesini genişletir. Kök uygulama klasörünün `lib` alt dizininde yer alan *myClasses.swf* adlı kitaplığı içe aktarmak için HTML dosyasında yer alan aşağıdaki komut dosyasını dahil edin:

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

Önemli: *Kitaplığın düzgün biçimde yüklenmesi için tür niteliği* `type="application/x-shockwave-flash"` *olmalıdır.*

SWF içeriği Flash Player 10 veya AIR 1.5 olarak derlendiyse, uygulama açıklayıcı dosyasının XML ad alanını AIR 1.5 ad alanına ayarlamanız gerekir.

AIR dosyası paketlendiğinde `lib` dizini ve `myClasses.swf` dosyası da dahil edilmelidir.

İçe aktarılan sınıflara JavaScript Window nesnesinin `runtime` özelliği aracılığıyla erişin:

```
var libraryObject = new window.runtime.LibraryClass();
```

SWF dosyasındaki sınıflar paketler halinde düzenlenmişse, paket adını da eklemelisiniz. Örneğin, LibraryClass tanımı *utilities* adlı paketteyse, aşağıdaki ifadeyi içeren sınıf örneğini oluşturursunuz:

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

Not: *ActionScript SWF kitaplığını AIR'de yer alan bir HTML sayfasının parçası olarak kullanmak üzere derlemek için* `acompc` *derleyicisini kullanın. Acompc yardımcı programı, Flex SDK'nin parçasıdır ve Flex SDK belgesinde açıklanmıştır.*

HTML DOM ve JavaScript nesnelerine içe aktarılan bir ActionScript dosyasından erişme

Adobe AIR 1.0 ve üstü

`<script>` etiketi kullanılarak sayfaya aktarılmış bir SWF dosyasındaki ActionScript'ten HTML sayfasındaki nesnelere erişmek için, `window` veya `document` gibi bir JavaScript nesnesine ilişkin başvuruyu ActionScript kodunda tanımlanan bir işleve iletin. JavaScript nesnesine (veya iletilen başvuru aracılığıyla erişilebilen diğer nesnelere) erişmek için işlevdeki başvuruyu kullanın.

Örneğin, aşağıdaki HTML sayfasına bakın:

```
<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>
```

Bu basit HTML sayfası *num* adlı bir JavaScript değişkeni ve *getStatus()* adlı bir JavaScript işlevi içerir. Bunların her ikisi de sayfanın `window` nesnesinin özellikleridir. Ayrıca, `window.document` nesnesi (*p1* kimliğiyle) adlandırılan bir P öğesi içerir.

Sayfa, ASClass sınıfını içeren "ASLibrary.swf" adlı ActionScript dosyasını yükler. ASClass, bu JavaScript nesnelerinin değerlerini izleyen `accessDOM()` adlı bir işlev tanımlar. `accessDOM()` yöntemi JavaScript Window nesnesini argüman olarak alır. Window başvurusunu kullanarak, aşağıdaki tanımda gösterildiği şekilde değişkenler, işlevler ve DOM öğeleri dahil sayfadaki diğer nesnelere erişebilir:

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

İçe aktarılan ActionScript sınıfından HTML sayfasının özelliklerini getirebilir ve ayarlayabilirsiniz. Örneğin, aşağıdaki işlev sayfadaki `p1` öğesinin içeriğini ayarlar ve `foo` JavaScript değişkeninin değerini belirler:

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Date ve RegExp nesnelerini dönüştürme

Adobe AIR 1.0 ve üstü

JavaScript ve ActionScript dillerinin her ikisi de Date ve RegExp sınıflarını tanımlar, ancak bu türdeki nesneler iki yürütme bağlamı arasında otomatik olarak dönüştürülmez. Date ve RegExp nesnelerini farklı yürütme bağlamlarındaki özellikleri ve işlev parametrelerini ayarlamak için kullanmadan önce eşdeğer türe dönüştürmelisiniz.

Örneğin, aşağıdaki ActionScript kodu, `jsDate` adlı JavaScript Date nesnesini ActionScript Date nesnesine dönüştürür:

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

Aşağıdaki ActionScript kodu, `jsRegExp` adlı JavaScript RegExp nesnesini ActionScript RegExp nesnesine dönüştürür:

```
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);
```

HTML stil sayfasını ActionScript'ten değiştirme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi `complete` olayını gönderdiğinde, sayfadaki CSS stillerini inceleyebilir ve değiştirebilirsiniz.

Örneğin, aşağıdaki basit HTML belgesine bakın:

```
<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>
```

HTMLLoader nesnesi bu içeriği yükledikten sonra sayfadaki CSS stillerini burada gösterildiği şekilde `window.document.styleSheets` dizisinin `cssRules` dizisi yoluyla değiştirebilirsiniz:

```
 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";
}
```

Bu kod CSS stillerini, elde edilen HTML belgesinin aşağıdaki gibi görüneceği biçimde ayarlar:

Style 1A

Style 1B

Style 2

HTMLLoader nesnesi `complete` olayını gönderdikten sonra kodun sayfaya stiller ekleyebileceğini göz önünde bulundurun.

Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği

Adobe AIR 1.0 ve üstü

Çalıştırma zamanı güvenlik modeli kodu farklı kaynaklardan ayırır. Farklı güvenlik sanal alanlarında içerik için komut dosyası oluşturarak bir güvenlik sanal alanındaki içeriğin diğer sanal alandaki seçilen özellikler ve yöntemlere erişmesini sağlayabilirsiniz.

AIR güvenlik sanal alanları ve JavaScript kodu

Adobe AIR 1.0 ve üstü

AIR, bir etki alanındaki kodun diğer etki alanındaki içerikle etkileşime girmesini engelleyen bir aynı kaynak politikası uygular. Tüm dosyalar, kaynakları temel alınarak bir sanal alana yerleştirilir. Genellikle uygulama sanal alanındaki içerik aynı kaynak ilkesini ihlal edemez ve uygulama yükleme dizininin dışından yüklenen içerik için çapraz komut dosyası oluşturur. Ancak, AIR uygulama dışı içeriğe ilişkin çapraz komut dosyası oluşturmanızı sağlayan birkaç teknik sunar.

Bir teknik, uygulama içeriğini farklı bir güvenlik sanal alanına eşlemek için çerçeveleri veya iframe'leri kullanır. Uygulamanın sanal alanlı bölümünden yüklenen tüm sayfalar, uzak etki alanından yüklenmiş gibi davranır. Örneğin, bu içerik uygulama içeriğini *example.com* etki alanına yükleyerek example.com'dan yüklenen sayfalar için çapraz komut dosyası oluşturabilir.

Teknik uygulama içeriğini farklı bir sanal alana yerleştirdiğinden, bu içerikte bulunan kod artık değerlendirilen dizelerdeki kod yürütmesine ilişkin kısıtlamalara tabi değildir. Bu sanal alan eşleme tekniğini, uzak içerik için çapraz komut dosyası oluşturmanız gerekmese bile, bu kısıtlamaları kolaylaştırmak için kullanabilirsiniz. İçeriği bu şekilde eşlemek özellikle birden çok JavaScript çerçevesinden biriyle veya dizelerin değerlendirilmesine dayanan mevcut kodla çalışırken kullanışlıdır. Ancak, güvenli olmayan içeriğin uygulama sanal alanı dışında çalışırken yerleştirilip ve yürütülebiliyor olması riskini göz önünde bulundurun ve buna karşı önlem alın.

Aynı zamanda, başka bir sanal alana eşlenen uygulama içeriği AIR API'lerine erişimini kaybeder, bu nedenle sanal alan eşleme tekniği AIR işlevlerini uygulama sanal alanının dışında yürütülen koda göstermek için kullanılamaz.

Diğer bir çapraz komut dosyası oluşturma tekniği, uygulama dışı sanal alandaki içerikle bunun uygulama sanal alanında bulunan üst belgesindeki içerik arasında *sanal alan köprüsü* adı verilen bir arabirim oluşturmanıza olanak verir. Köprü, alt içeriğin üst içerik tarafından tanımlanan özellikler ve yöntemlere erişmesini, üst içeriğin alt içerik tarafından tanımlanan özellikler ve yöntemlere erişmesini, veya bunların her ikisini sağlar.

Son olarak, uygulama sanal alanından ve isteğe bağlı olarak diğer sanal alanlardan etki alanları arası XMLHttpRequests de gerçekleştirebilirsiniz.

Daha fazla bilgi için, bkz. "HTML frame ve iframe öğeleri" sayfa 983, "Adobe AIR'de HTML güvenliği" sayfa 1092, ve "XMLHttpRequest nesnesi" sayfa 977.

Uygulama içeriğini uygulama dışı sanal alana yükleme

Adobe AIR 1.0 ve üstü

Uygulama içeriğinin uygulama yükleme dizininin dışından yüklenen içerik için güvenli biçimde çapraz komut dosyası oluşturmasını sağlamak amacıyla, uygulama içeriğini harici içerikle aynı güvenlik sanal alanına yüklemek için `frame` veya `iframe` öğelerini kullanın. Uzak içerik için çapraz komut dosyası oluşturmanız gerekmiyorsa, ancak yine de uygulamanızın bir sayfasını uygulama sanal alanının dışından yüklemek istiyorsanız kaynak etki alanı olarak `http://localhost/` veya başka bir zararsız değer kullanın.

AIR, çerçeveye yüklenen uygulama dosyasının uygulama dışı bir sanal alana eşlenmesi gerekip gerekmediğini belirlemenizi sağlayan kare öğesine yeni `sandboxRoot` ve `documentRoot` niteliklerini ekler. `sandboxRoot` URL'sinin altındaki bir yola çözümlenen dosyalar bunun yerine `documentRoot` dizininden yüklenir. Güvenlik nedenleriyle, bu şekilde yüklenen uygulama içeriği, `sandboxRoot` URL'sinden yüklenmiş gibi ele alınır.

`sandboxRoot` özelliği, kare içeriğinin yerleştirileceği sanal alan ve etki alanını belirlemek için kullanılacak URL'yi belirtir. URL şemasının kullanması gereken `file:`, `http:` veya `https:`. İlişkili bir URL belirtirseniz, içerik uygulama sanal alanında kalır.

`documentRoot` özelliği, çerçeve içeriğinin yükleneceği kaynak dizini belirtir. URL şemasının kullanması gereken `file:`, `app:` veya `app-storage:`.

Aşağıdaki örnek, uygulamanın `sandbox` alt dizinine yüklenen içeriği uzak sanal alanda ve `www.example.com` etki alanında çalışacak biçimde eşler:

```
 <iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

`ui.html` sayfası aşağıdaki komut dosyası etiketini kullanarak yerel `sandbox` klasöründen bir javascript dosyası yükleyebilir:

```
<script src="http://www.example.com/local/ui.js"></script>
```

Ayrıca, aşağıdaki gibi komut dosyası etiketini kullanarak uzak sunucudaki bir dizinden de içerik yükleyebilir:

```
<script src="http://www.example.com/remote/remote.js"></script>
```

`sandboxRoot` URL'si, uzak sunucuda bulunan aynı URL'deki tüm içeriği maskeler. Yukarıdaki örnekte, AIR, isteği yerel uygulama dizinine yeniden eşlediğinden `www.example.com/local/` linkindeki (veya onun alt dizinlerinden birindeki) uzak içeriğe erişemezsiniz. İstekler, sayfada gezinmesinden, XMLHttpRequest'ten veya tüm diğer içerik yükleme yöntemlerinden türetilseler de yeniden eşlenirler.

Sanal alan köprü arabirimini kurma

Adobe AIR 1.0 ve üstü

Uygulama sanal alanındaki içeriğin uygulama dışı bir sanal alandaki içerik tarafından tanımlanan özellikler ve yöntemlere erişmesi veya uygulama dışı içeriğin uygulama sanal alanındaki içerik tarafından tanımlanan özellikler veya yöntemlere erişmesi gerektiğinde sanal alan köprüsü kullanmalısınız. Herhangi bir alt belgenin `window` nesnesine ait `childSandboxBridge` ve `parentSandboxBridge` özellikleriyle bir köprü oluşturun.

Alt sanal alan köprüsü oluşturma

Adobe AIR 1.0 ve üstü

`childSandboxBridge` özelliği, alt belgenin üst belgedeki içeriğe bir arabirim göstermesine olanak verir. Arabirim göstermek için `childSandbox` özelliğini alt belgedeki bir işleme veya nesneye ayarlarsınız. Böylece nesneye veya işleve üst belgedeki içerikten erişebilirsiniz. Aşağıdaki örnek, alt belgede çalışan bir komut dosyasının bir işlev ve bir özellik içeren nesneyi üst öğesine nasıl gösterdiğini açıklar:

```
var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Bu alt içerik, “alt” kimliği atanmış bir iframe'e yüklendiyse, karenin `childSandboxBridge` özelliğini okuyarak üst içerikten arabirime erişebilirsiniz.

```
var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

Üst sanal alan köprüsü oluşturma

Adobe AIR 1.0 ve üstü

`parentSandboxBridge` özelliği, üst belgenin alt belgedeki içeriğe bir arabirim göstermesine olanak verir. Üst belge, arabirim göstermek için alt belgenin `parentSandbox` özelliğini üst belgede tanımlanmış bir işleve veya nesneye ayarlar. Böylece nesneye veya işleve alt belgedeki içerikten erişebilirsiniz. Aşağıdaki örnek, üst karede çalışan bir komut dosyasının işlev içeren bir nesneyi alt belgeye nasıl gösterdiğini açıklar:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

Alt karedeki içerik, bu arabirimi kullanarak metni `save.txt` adlı bir dosyaya kaydedebilir, ancak dosya sistemine bunun dışında herhangi bir erişimi olmayacaktır. Alt içerik saklama işlevini aşağıdaki şekilde çağırabilir:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Uygulama içeriği, diğer sanal alanlara mümkün olan en dar arabirimi göstermelidir. Uygulama dışı içerik, yanlışlıkla veya kötü amaçla kod yerleştirmesine maruz kalabileceğinden yapısı gereği güvenilmez olarak ele alınmalıdır. Üst sanal alan köprüsü aracılığıyla gösterdiğiniz arabirimin hatalı kullanımını engellemek için uygun korumaları yerine koymalısınız.

Sayfa yükleme sırasında üst sanal alan köprüsüne erişme

Adobe AIR 1.0 ve üstü

Alt belgedeki komut dosyasının bir üst sanal alan köprüsüne erişmesi için, köprü komut dosyası çalıştırılmadan kurulmalıdır. Yeni bir sayfa DOM'u oluşturulduğunda, ancak herhangi bir komut dosyası ayrıştırılmadan veya DOM öğeleri eklenmeden pencere, kare ve iframe bir `dominitialize` olayı gönderir. Sayfa yapım sırasında, köprüyü alt belgedeki tüm komut dosyalarının erişebileceği kadar erken oluşturmak için `dominitialize` olayını kullanabilirsiniz.

Aşağıdaki örnek, alt kareden gönderilen `dominitialize` olayına karşılık olarak üst sanal alan köprüsünün nasıl oluşturulacağını gösterir:

```
<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>
```

Aşağıdaki `child.html` belgesi, alt içeriğin üst sanal alan köprüsüne nasıl erişebileceğini gösterir:

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

Kare yerine alt penceredeki `dominitialize` olayını dinlemek için dinleyiciyi `window.open()` işleviyle oluşturulan yeni alt window nesnesine eklemelisiniz:

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

Bu durumda, uygulama içeriğini uygulama dışı bir sanal alana eşlemenin yolu yoktur. Bu teknik, yalnızca `child.html` öğesi uygulama dizininin dışından yüklendiğinde kullanışlıdır. Yine de penceredeki uygulama içeriğini uygulama dışı bir sanal alana eşleyebilirsiniz, ancak öncesinde alt belgeyi yüklemek ve bunu istenen sanal alana eşlemek için kendisi çerçeveler kullanan bir ara sayfa yüklemelisiniz.

Pencere oluşturmak için HTMLLoader sınıfının `createRootWindow()` işlevini kullandığınızda, yeni pencere `createRootWindow()` öğesinin çağrıldığı kaynak belgenin alt öğesi değildir.. Bu nedenle, çağıran pencere ve yeni pencereye yüklenen uygulama dışı içerik arasında sanal alan köprüsü oluşturamazsınız. Bunun yerine, alt belgeyi yüklemek için kendisi çerçeveler kullanan yeni pencereye bir ara pencere yükleme işlevini kullanmalısınız. Yeni pencerenin üst belgesi ve kareye yüklenen alt belge arasında köprü oluşturabilirsiniz.

Bölüm 60: AIR HTML Kabı için Komut Dosyası Oluşturma

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfı, Adobe® AIR® içindeki HTML içeriği için kap olarak görev yapar. Bu sınıf, ActionScript® 3.0 görüntü listesindeki nesnenin davranışı ve görünümünün denetlenmesi için Sprite sınıfından miras alınan birçok özellik ve yöntem sağlar. Sınıf ayrıca, HTML içeriğinin yüklenmesi, HTML içeriğiyle etkileşim kurulması ve geçmişin yönetilmesi gibi görevlere ilişkin özellikleri ve yöntemleri tanımlar.

HTMLHost sınıfı bir HTMLLoader öğesinin varsayılan davranış kümesini ayarlar. HTMLLoader nesnesini oluşturduğunuzda, HTMLHost uygulaması sağlanmaz. Bu nedenle HTML içeriği pencere konumunun veya pencere başlığının değiştirilmesi gibi varsayılan davranışlardan birini tetiklediğinde, hiçbir şey gerçekleşmez. Uygulamanız için istenen davranışları tanımlamak için HTMLHost sınıfını genişletebilirsiniz.

AIR tarafından oluşturulan HTML pencereleri için varsayılan bir HTMLHost uygulaması sağlanmıştır. `True` olarak ayarlanmış `defaultBehavior` parametresiyle oluşturulan yeni HTMLHost nesnesini kullanarak nesnenin `htmlHost` özelliğini ayarlama yoluyla, varsayılan HTMLHost uygulamasını başka bir HTMLLoader nesnesine atayabilirsiniz.

Not: *Adobe® Flex™ Çerçevesi'nde HTMLLoader nesnesi mx:HTML bileşeni tarafından kaydırılır. Flex ile birlikte HTML bileşiğini kullanın.*

HTMLLoader nesnelerinin görüntü özellikleri

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi, Adobe® Flash® Player Sprite sınıfının görüntü özelliklerini miras alır. Örneğin arka plan rengini değiştirebilir, yeniden boyutlandırabilir, taşıyabilir ve gizleyebilirsiniz. Veya filtreler, maskeler, ölçeklendirme ve döndürme gibi gelişmiş efektler uygulayabilirsiniz. Efektleri uygularken, okunaklılık üzerindeki etkilerini göz önüne alın. HTML sayfasına yüklenen SWF ve PDF içeriği, bazı efektler uygulandığında görüntülenemez.

HTML pencereleri, HTML içeriğini oluşturan bir HTMLLoader nesnesi içerir. Bu nesne pencere alanı içinde sınırlanmıştır, bu nedenle boyut, konum, döndürme veya ölçek faktörünün değiştirilmesi her zaman istenen sonuçları vermez.

Temel görüntü özellikleri

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesinin temel görüntü özellikleri, boyutu ayarlamak ve denetimi göstermek veya gizlemek için üst display nesnesi içinde denetimi konumlandırmanıza olanak verir. HTML penceresinin HTMLLoader nesnesi için bu özellikleri değiştirmemelisiniz.

Temel özellikler:

Özellik	Notlar
`x`, `y`	Nesneyi üst kabı içinde.
`width`, `height`	Görüntü alanının boyutlarını değiştirir.
`visible`	Nesnenin ve sahip olduğu tüm içeriğin görünürlüğünü denetler.

HTML penceresi dışında, HTMLLoader nesnesinin `width` ve `height` özellikleri varsayılan olarak 0'dır. Yüklenen HTML içeriği görünür olmadan önce genişlik ve yüksekliği ayarlamalısınız. HTML içeriği HTMLLoader boyutuna çizilir, içerikteki HTML ve CSS özelliklerine göre şekillendirilir. HTMLLoader boyutunun değiştirilmesi içeriği yeniden akıtır.

Yeni bir HTMLLoader nesnesine (halen 0 olarak ayarlanmış `width` içeren) içerik yüklerken HTMLLoader görüntü `width` ve `height` özelliklerini `contentWidth` ve `contentHeight` özelliklerini kullanarak ayarlamak size uygun gelebilir. Bu teknik, HTML ve CSS akış kurallarına göre şekillendirildiğinde belirli bir minimum genişliğe sahip olan sayfalar için çalışır. Ancak, bazı sayfalar HTMLLoader tarafından sağlanan belirli bir genişlik olmadığında uzun ve dar bir mizanpajın içine akar.

Not: *HTMLLoader nesnesinin genişliğini ve yüksekliğini değiştirdiğinizde, scaleX ve scaleY değerleri diğer birçok display nesnesindekinin aksine değişmez.*

HTMLLoader içeriğinin şeffaflığı

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesinin varsayılan olarak `true` olan `paintsDefaultBackground` özelliği HTMLLoader nesnesinin opak bir arka plan çizip çizmeyeceğini belirler. `paintsDefaultBackground` öğesi `false` olduğunda, arka plan boştur. Display nesnesi kabı veya HTMLLoader nesnesinin altındaki diğer display nesneleri, HTML içeriğinin ön plan öğelerinin arkasında görünür.

HTML belgesinin gövde öğesi veya diğer öğelerinden biri bir arka plan rengi belirlediğinde (örneğin, `style="background-color:gray"`), HTML'in bu bölümünün arka plan rengi opaktır ve belirlenen arka plan rengiyle oluşturulur. HTMLLoader nesnesinin `opaqueBackground` özelliğini ayarladığınızda ve `paintsDefaultBackground` öğesi `false` olduğunda, `opaqueBackground` renk kümesi görünür.

Not: *HTML belgesindeki öğe için alfa karışımlı bir arka plan sağlamak için şeffaf, PNG biçiminde bir grafik kullanabilirsiniz. HTML öğesinin opaklık stilini ayarlama desteklenmez.*

HTMLLoader içeriğini ölçekleme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesini 1.0 dışında bir ölçek faktörüyle ölçeklemekten kaçının. HTMLLoader içeriğindeki metin belirli bir çözünürlükte oluşturulur ve HTMLLoader nesnesi ölçeklendiğinde pikseller halinde görünebilir. Pencere yeniden boyutlandırıldığında HTMLLoader nesnesi ve içeriğinin ölçeklenmesini engellemek için Sahne'nin `scaleMode` özelliğini `StageScaleMode.NO_SCALE` olarak ayarlayın.

HTML sayfasına SWF ve PDF içeriği yüklenirken dikkate alınması gerekenler

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesine yüklenen SWF ve PDF içeriği aşağıdaki koşullarda kaybolur:

- HTMLLoader nesnesini 1.0 dışında bir faktöre ölçeklerseniz.
- HTMLLoader nesnesinin alfa özelliğini 1.0 dışında bir değere ayarlarsanız.
- HTMLLoader içeriğini döndürürseniz.

Sorunlu özellik ayarını ve etkin filtreleri kaldırdığınızda içerik yeniden görünüz.

Ayrıca çalışma zamanı, saydam pencerelerde PDF içeriğini görüntüleyemez. Çalışma zamanı, HTML sayfasına gömülü SWF içeriğini yalnızca nesnenin veya gömme etiketinin `wmode` parametresi `opaque` veya `transparent` olarak ayarlandığında görüntüler. `wmode` öğesinin varsayılan değeri `window` olduğundan, wmode parametresini özellikle ayarlamadığınız sürece SWF içeriği saydam pencerelerde görüntülenmez.

Not: *AIR 1.5.2'den önce, HTML'de gömülü SWF hangi wmode değeri kullanılırsa kullanılsın görüntülenemiyordu.*

HTMLLoader nesnesinde bu tür medyaların yüklenmesi hakkında daha fazla bilgi için bkz. "SWF içeriğini HTML'e gömme" sayfa 1002 ve "AIR'ye PDF içeriği ekleme" sayfa 557.

Gelişmiş görüntü özellikleri

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfı, özel efektler için kullanılabilecek birçok yöntemi miras alır. Bu efektler HTMLLoader görüntüsüyle kullanıldıklarında genellikle sınırlamalar içerir, ancak geçişler veya diğer geçici efektler için kullanışlı olabilir. Örneğin, kullanıcı girdisi toplamak için bir iletişim penceresi görüntülediğinizde, kullanıcı iletişim kutusunu kapatana kadar ana pencere görüntüsünü bulanıklaştırabilirsiniz. Benzer şekilde, bir pencereyi kapatırken görüntünün yavaşça kapanmasını sağlayabilirsiniz.

Gelişmiş görüntü özellikleri şunları içerir:

Özellik	Sınırlamalar
`alpha`	HTML içeriğinin okunaklılığını azaltabilir
`filters`	HTML penceresinde, dış efektler pencere kenarı tarafından kesilir.
`graphics`	Varsayılan arka plan dahil grafik komutlarıyla çizilen şekiller HTML içeriğinin altında görünür. Çizilen şekillerin görünür olması için paintsDefaultBackground özelliği false olmalıdır.
`opaqueBackground`	Varsayılan arka planın rengini değiştirmez. Bu renk katmanının görünür olması için paintsDefaultBackground özelliği false olmalıdır.
`rotation`	Dikdörtgen HTMLLoader alanının köşeleri pencere kenarı tarafından kesilebilir. HTML içeriğine yüklenen SWF ve PDF içeriği görüntülenmez.
`scaleX`, `scaleY`	Oluşturulan görüntü, 1'den büyük ölçek faktörlerinde pikseller halinde görünebilir. HTML içeriğine yüklenen SWF ve PDF içeriği görüntülenmez.
`transform`	HTML içeriğinin okunaklılığını azaltabilir. HTML görüntüsü pencere kenarı tarafından kesilebilir. Dönüştürme döndürme, ölçekleme veya eğme içeriyorsa, HTML içeriğine yüklenen SWF ve PDF içeriği görüntülenmez.

Aşağıdaki örnek, tüm HTML görüntüsünü bulanıklaştırmak için `filters` dizisinin nasıl ayarlanacağını gösterir:

```
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;
```

HTML içeriğini kaydırma

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfı, HTML içeriği kaydırmayı denetlemenizi sağlayan aşağıdaki özellikleri içerir:

Özellik	Açıklama
`contentHeight`	HTML içeriğinin piksel cinsinden yüksekliği.
contentWidth	HTML içeriğinin piksel cinsinden genişliği.
scrollH	HTMLLoader nesnesinde HTML içeriğinin yatay kaydırma konumu.
scrollV	HTMLLoader nesnesinde HTML içeriğinin dikey kaydırma konumu.

Aşağıdaki kod, `scrollV` özelliğini HTML içeriğinin sayfanın altına kaydırılacağı şekilde ayarlar.

```
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;
}
```

HTMLLoader, yatay ve dikey kaydırma çubukları içermez. Kaydırma çubuklarını ActionScript'te veya Flex bileşenini kullanarak uygulayabilirsiniz. Flex HTML bileşeni, otomatik olarak HTML içeriği için kaydırma çubukları içerir. Kaydırma çubuklarına sahip bir HTMLLoader nesnesi içeren bir pencere oluşturmak için `HTMLLoader.createRootWindow()` yöntemini de kullanabilirsiniz (bkz. "HTML içeriğini kaydırarak pencereler oluşturma" sayfa 1028).

HTML geçmiş listesine erişme

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesine yeni sayfalar yüklendiğinde, çalışma zamanı nesneye ilişkin bir geçmiş listesi tutar. Geçmiş listesi, HTML sayfasındaki `window.history` nesnesine karşılık gelir. HTMLLoader nesnesi HTML geçmiş listesiyle çalışmanızı sağlayan aşağıdaki özellikler ve yöntemleri içerir:

Sınıf üyesi	Açıklama
`historyLength`	Geri ve ileri girişler dahil, geçmiş listesinin tam uzunluğu.
`historyPosition`	Geçmiş listesindeki geçerli konum. Bu konumun önündeki geçmiş öğeleri "geri" gezinmeyi, arkasındaki öğeler "ileri" gezinmeyi temsil eder.
`getHistoryAt()`	Geçmiş listesinde belirtilen konumdaki geçmiş girişine karşılık gelen URLRequest nesnesini döndürür.
`historyBack()`	Mümkünse geçmiş listesinde geriye doğru gezinir.
`historyForward()`	Mümkünse geçmiş listesinde ileriye doğru gezinir.
`historyGo()`	Tarayıcı geçmişinde belirtilen adım sayısı kadar gezinir. Pozitifse ileriye, negatifse geriye doğru gezinir. Sıfıra doğru gezinme, sayfayı yeniden yükler. Liste sonunun ardındaki bir konum belirtildiğinde, listenin sonuna gider.

Geçmiş listesindeki öğeler HTMLHistoryItem öğesinin nesne türleri olarak depolanır. HTMLHistoryItem sınıfı, aşağıdaki özelliklere sahiptir:

Özellik	Açıklama
`isPost`	HTML sayfası POST verileri içeriyorsa `true` olarak ayarlanır.
`originalUrl`	HTML sayfasının tüm yeniden yönlendirmelerden önceki orijinal URL'si.
`title`	HTML sayfasının başlığı.
`url`	HTML sayfasının URL'si.

HTML içeriği yüklenirken kullanılan kullanıcı aracısını ayarlama

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfı, HTMLLoader tarafından kullanılan kullanıcı aracısı dizesini ayarlamanızı sağlayan `userAgent` özelliğini içerir. `load()` yöntemini çağırmadan önce HTMLLoader nesnesinin `userAgent` özelliğini ayarlayın. HTMLLoader nesnesinde bu özelliği ayarladığınızda, `load()` yöntemine iletilen URLRequest öğesinin `userAgent` özelliği *kullanılmaz*.

Bir uygulama etki alanındaki tüm HTMLLoader nesneleri tarafından kullanılan varsayılan kullanıcı aracısını `URLRequestDefaults.userAgent` özelliğini kullanarak ayarlayabilirsiniz. Statik URLRequestDefaults özellikleri, yalnızca HTMLLoader nesnelerinin `load()` yöntemiyle kullanılan URLRequest nesneleri için değil, tüm URLRequest nesneleri için varsayılan olarak geçerlidir. HTMLLoader nesnesinin `userAgent` özelliğinin ayarlanması, varsayılan `URLRequestDefaults.userAgent` ayarını geçersiz kılar.

HTMLLoader nesnesinin `userAgent` özelliği veya `URLRequestDefaults.userAgent` için kullanıcı aracı değeri ayarlamazsanız, varsayılan AIR kullanıcı aracı değeri kullanılır. Bu varsayılan değer, aşağıdaki iki örnekte olduğu gibi çalışma zamanı işletim sistemine (Mac OS veya Windows gibi), çalışma zamanı diline ve çalışma zamanı sürümüne bağlı olarak değişir:

- `"Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) AdobeAIR/1.0"`
- `"Mozilla/5.0 (Windows; U; en) AppleWebKit/420+ (KHTML, like Gecko) AdobeAIR/1.0"`

HTML içeriğinde kullanım için karakter kodlamasını ayarlama

Adobe AIR 1.0 ve üstü

HTML sayfası, aşağıdaki şekilde `meta` etiketini dahil ederek kullandığı karakter kodlamasını belirleyebilir:

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

HTMLLoader nesnesinin `textEncodingOverride` özelliğini ayarlayarak belirli bir karakter kodlamasının kullanılmasını sağlamak için sayfa ayarını geçersiz kılın:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

HTML sayfası, HTMLLoader nesnesinin `textEncodingFallback` özelliğini içeren bir ayar belirtmediğinde kullanılacak HTMLLoader içeriğine ilişkin karakter kodlamasını belirleyin:

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

`textEncodingOverride` özelliği, HTML sayfasındaki ayarı geçersiz kılar. Ve `textEncodingOverride` özelliği ile HTML sayfasındaki ayar, `textEncodingFallback` özelliğini geçersiz kılar.

HTML içeriğini yüklemeden önce `textEncodingOverride` özelliğini veya `textEncodingFallback` özelliğini ayarlayın.

HTML içeriği için tarayıcı benzeri kullanıcı arabirimleri tanımlama

Adobe AIR 1.0 ve üstü

JavaScript, HTML içeriğini görüntüleyen pencerenin denetlenmesi için birkaç çeşitli API'ler sağlar. AIR'de, bu API'ler özel bir HTMLHost sınıfı uygulayarak geçersiz kılınabilir.

HTMLHost sınıfının genişletilmesi hakkında

Adobe AIR 1.0 ve üstü

Örneğin, uygulamanız sekmeli arabirimde birden çok HTMLLoader nesnesi sunuyorsa, ana pencere başlığını değil, sekmenin etiketini değiştirmek için yüklenen HTML sayfaları tarafından başlık değişikliklerinin yapılmasını isteyebilirsiniz. Benzer şekilde kodunuz üst görüntüleme nesnesi kabındaki HTMLLoader nesnesini yeniden konumlandırarak, HTMLLoader nesnesini içeren pencereyi taşıyarak, hiçbir şey yapmayarak veya tamamen farklı bir şey yaparak bir `window.moveTo()` çağrısına yanıt verebilir.

AIR HTMLHost sınıfı, aşağıdaki JavaScript özellikleri ve yöntemlerini denetler:

- `window.status`
- `window.document.title`
- `window.location`
- `window.blur()`
- `window.close()`
- `window.focus()`
- `window.moveBy()`
- `window.moveTo()`
- `window.open()`
- `window.resizeBy()`
- `window.resizeTo()`

`new HTMLLoader()` kullanarak bir HTMLLoader nesnesi oluşturduğunuzda, listelenen JavaScript özellikleri ve yöntemleri etkinleştirilmez. HTMLHost sınıfı, bu JavaScript API'lerinin varsayılan, tarayıcı benzeri bir uygulamasını sağlar. Davranışı özelleştirmek için ayrıca HTMLHost sınıfını genişletebilirsiniz. Varsayılan davranışı destekleyen bir HTMLHost nesnesi oluşturmak için HTMLHost yapıcısındaki `defaultBehaviors` parametresini true olarak ayarlayın:

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

AIR'de HTMLLoader sınıfının `createRootWindow()` yöntemiyle bir HTML penceresi oluşturduğunuzda, varsayılan davranışları destekleyen bir HTMLHost örneği otomatik olarak atanır. HTMLLoader nesnesinin `htmlHost` özelliğine farklı bir HTMLHost uygulaması atayarak ana bilgisayar nesne davranışını değiştirebilir veya özellikleri tamamen devre dışı bırakmak için `null` değerini atayabilirsiniz.

Not: *AIR, HTML tabanlı bir AIR uygulaması için oluşturulan ilk pencereye ve JavaScript `window.open()` yönteminin varsayılan uygulamasıyla oluşturulan tüm pencerelere varsayılan bir HTMLHost nesnesi atar.*

Örnek: HTMLHost sınıfını genişletme

Adobe AIR 1.0 ve üstü

Aşağıdaki örnek, HTMLHost sınıfının genişletilmesi yoluyla HTMLLoader nesnesinin kullanıcı arabirimini etkileme biçiminin nasıl özelleştirildiğini gösterir:

Flex örneği:

1 HTMLHost sınıfını genişleten bir sınıf (alt sınıf) oluşturun.

2 Kullanıcı arabirimi ilişkili ayarlardaki değişiklikleri işlemek için yeni sınıfın yöntemlerini geçersiz kılın. Örneğin, aşağıdaki CustomHost sınıfı `window.open()` çağrılarına ilişkin davranışları tanımlar ve `window.document.title` olarak değişir. `window.open()` yöntemine yapılan çağrılar, HTML sayfasını yeni bir pencerede açar ve bu pencerenin başlığını `window.document.title` (HTML sayfasının `<title>` öğesinin ayarı dahil) olarak ayarlamak üzere değiştirir.

```
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 HTMLLoader nesnesini içeren kodda (HTMLHost yeni alt sınıfının kodu değil), yeni sınıf nesnesi oluşturun. Yeni nesneyi HTMLLoader nesnesinin `htmlHost` özelliğine atayın. Aşağıdaki Flex kodu, önceki adımda tanımlanan CustomHost sınıfını kullanır:

```
<?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>
```

Burada açıklanan kodu test etmek için, aşağıdaki içeriğe sahip bir HTML dosyasını uygulama dizinine dahil edin:

```
<html>
    <head>
        <title>Test</title>
    </head>
    <script>
        function openWindow()
        {
            window.runtime.trace("in");
            document.title = "foo"
            window.open('Test.html');
            window.runtime.trace("out");
        }
    </script>
    <body>
        <a href="#" onclick="openWindow()">window.open('Test.html')</a>
    </body>
</html>
```

Flash Professional örneği:

1. AIR için bir Flash dosyası oluşturun. Bu dosyanın document sınıfını CustomHostExample olarak ayarlayın ve daha sonra dosyayı CustomHostExample.fla olarak kaydedin.
2. HTMLHost sınıfını (alt sınıf) genişleten bir sınıf içeren CustomHost.as adlı bir ActionScript dosyası oluşturun. Bu sınıf, kullanıcı arabirimi ilişkili ayarlardaki değişiklikleri işlemek için yeni sınıfın yöntemlerini geçersiz kılar. Örneğin, aşağıdaki CustomHost sınıfı `window.open()` çağrılarına ilişkin davranışları tanımlar ve `window.document.title` olarak değişir. `window.open()` yöntemine yapılan çağrılar HTML sayfasını yeni bir pencerede açar ve bu pencerenin başlık `window.document.title` özelliğini (HTML sayfasının `<title>` öğesi dahil) olarak ayarlamak üzere değiştirir.

```
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 Uygulamaya ilişkin document sınıfını içerecek CustomHostExample.as adlı başka bir ActionScript dosyası oluşturun. Bu sınıf HTMLLoader nesnesini oluşturur ve bu nesnenin host özelliğini önceki adımda tanımlanan CustomHost sınıfı örneğine ayarlar:

```
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;
        }
    }
}
```

Burada açıklanan kodu test etmek için, aşağıdaki içeriğe sahip bir HTML dosyasını uygulama dizinine dahil edin:

```
<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>
```

Window.location özelliğindeki değişiklikleri işleme

Adobe AIR 1.0 ve üstü

HTML sayfasının URL'sindeki değişiklikleri işlemek için `locationChange()` yöntemini geçersiz kılın. Bir sayfadaki JavaScript `window.location` değerini değiştirdiğinde `locationChange()` yöntemi çağrılır. Aşağıdaki örnek istenen URL'yi yükler:

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

Not: *Geçerli HTMLLoader nesnesine başvurmak için HTMLHost nesnesinin htmlLoader özelliğini kullanabilirsiniz.*

window.moveBy(), window.moveTo(), window.resizeTo(), window.resizeBy() için JavaScript çağrılarını işleme

Adobe AIR 1.0 ve üstü

HTML içeriğinin sınırlarındaki değişiklikleri işlemek için `set windowRect()` yöntemini geçersiz kılın. Bir sayfadaki JavaScript `window.moveBy()`, `window.moveTo()`, `window.resizeTo()` veya `window.resizeBy()` çağırdığında, `set windowRect()` yöntemi çağrılır. Aşağıdaki örnek, masaüstü penceresinin sınırlarını günceller:

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

window.open() için JavaScript çağrılarını işleme

Adobe AIR 1.0 ve üstü

`window.open()` için JavaScript çağrılarını işlemek üzere `createWindow()` yöntemini geçersiz kılın. `createWindow()` yöntemi uygulamaları, yeni bir HTMLLoader nesnesi oluşturmak ve döndürmekle sorumludur. HTMLLoader nesnesini genellikle yeni bir pencerede görüntülersiniz, ancak pencere oluşturma zorunlu değildir.

Aşağıdaki örnek, window ve HTMLLoader nesnesi oluşturmak için `HTMLLoader.createRootWindow()` kullanan `createWindow()` işlevinin nasıl uygulanacağını gösterir. Buna ek olarak, bir NativeWindow nesnesini ayrıca oluşturabilir ve HTMLLoader nesnesini pencere sahnesine ekleyebilirsiniz.

```
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;
}
```

Not: *Bu örnek, `window.open()` ile oluşturulan tüm yeni pencerelere özel HTMLHost uygulamasını atar. Ayrıca farklı bir uygulama kullanabilir veya istediğinizde yeni pencereler için `htmlHost` özelliğini null değerine ayarlayabilirsiniz.*

`createWindow()` yöntemine parametre olarak iletilen nesne bir HTMLWindowCreateOptions nesnesidir. HTMLWindowCreateOptions sınıfı, `window.open()` çağrısındaki `features` parametre dizesinde değerleri bildiren özellikleri içerir:

HTMLWindowCreateOptions özelliği	window.open() JavaScript çağrısındaki features dizesinde karşılık gelen ayar
`fullscreen`	`fullscreen`
`height`	`height`
`locationBarVisible`	`location`
`menuBarVisible`	`menubar`
`resizeable`	`resizable`
`scrollBarsVisible`	`scrollbars`
`statusBarVisible`	`status`
`toolBarVisible`	`toolbar`
`width`	`width`
`x`	`left` veya `screenX`
`y`	`top` veya `screenY`

HTMLLoader sınıfı, özellik dizesinde belirtilebilecek tüm özellikleri uygulamaz. Uygulamanız gerektiğinde kaydırma çubukları, konum çubukları, menü çubukları, durum çubukları ve araç çubukları sağlamalıdır.

JavaScript `window.open()` yöntemine ilişkin diğer argümanlar sistem tarafından işlenir. `createWindow()` uygulaması, HTMLLoader nesnesine içerik yükleyemez veya pencere başlığını ayarlayamaz.

window.close() için JavaScript çağrılarını işleme

Adobe AIR 1.0 ve üstü

`Window.close()` yöntemine JavaScript çağrılarını işlemek için `windowClose()` yöntemini geçersiz kılın. Aşağıdaki örnek, `window.close()` yöntemi çağrıldığında masaüstü penceresini kapatır:

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

`window.close()` için JavaScript çağrılarının varolan pencereyi kapatması gerekmez. Örneğin, aşağıdaki örnekte olduğu gibi pencereyi (başka bir içeriğe sahip olabilir) açık bırakarak görüntü listesinden HTMLLoader nesnesini kaldırabilirsiniz:

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

window.status özelliğindeki değişiklikleri işleme

Adobe AIR 1.0 ve üstü

`window.status` değerindeki JavaScript değişikliklerini işlemek için `updateStatus()` yöntemini geçersiz kılın. Aşağıdaki örnek durum değerini izler:

```
 override public function updateStatus(status:String):void
{
    trace(status);
}
```

İstenen durum dize olarak `updateStatus()` yöntemine iletilir.

HTMLLoader nesnesi durum çubuğu sağlamaz.

window.document.title özelliğindeki değişiklikleri işleme

Adobe AIR 1.0 ve üstü

`window.document.title` değerindeki JavaScript değişikliklerini işlemek için `updateTitle()` yöntemini geçersiz kılın. Aşağıdaki örnek, pencere başlığını değiştirir ve başlığa "Örnek," dizesini ekler:

```
 override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

HTML sayfasında `document.title` ayarlandığında, istenen başlık dize olarak `updateTitle()` yöntemine iletilir.

`document.title` öğesindeki değişiklikler, HTMLLoader nesnesini içeren pencerenin başlığını değiştirmek zorunda değildir. Örneğin metin alanı gibi başka bir arabirim öğesini değiştirebilirsiniz.

window.blur() ve window.focus() için JavaScript çağrılarını işleme

Adobe AIR 1.0 ve üstü

Aşağıdaki örnekte gösterildiği gibi `window.blur()` ve `window.focus()` için JavaScript çağrılarını işlemek için `windowBlur()` ve `windowFocus()` yöntemlerini geçersiz kılın:

```
 override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

Not: *AIR, bir pencerenin veya bir uygulamanın devre dışı bırakılması için API sağlamaz.*

HTML içeriğini kaydırarak pencereler oluşturma

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfı, bir HTMLLoader nesnesi içeren ve bu pencere için bazı kullanıcı arabirimi ayarları tanımlayan yeni bir pencere (bir NativeWindow nesnesi tarafından temsil edilen) açmanızı sağlayan, `HTMLLoader.createRootWindow()` adında statik bir yöntem içerir. Bu yöntem, kullanıcı arabirimini tanımlamanızı sağlayan dört parametre içerir:

Parametre	Açıklama
`visible`	Pencerenin başlangıçta görünür olup olmadığını belirten bir Boolean değeri (`true` veya `false`).
`windowInitOptions`	NativeWindowInitOptions nesnesi. NativeWindowInitOptions sınıfı, NativeWindow nesnesi için aşağıdaki başlatma seçeneklerini tanımlar: pencerenin küçültülebilir, büyütülebilir veya yeniden boyutlandırılabilir olup olmadığı, pencerenin sistem kromu veya özel krom içerdiği, pencerenin şeffaf olup olmadığı (sistem kromu kullanmayan pencereler için) ve pencerenin türü.
`scrollBarsVisible`	Kaydırma çubuklarının olup olmadığı (`true` veya `false`).
`bounds`	Yeni pencerenin konumunu ve boyutunu tanımlayan bir Rectangle nesnesi.

Örneğin aşağıdaki kod, dikey kaydırma çubukları kullanan HTMLLoader içeriğine sahip bir pencere oluşturmak için `HTMLLoader.createRootWindow()` yöntemini kullanır.

```
 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();
```

Not: *Doğrudan JavaScript içinde `createRootWindow()` çağrılarak oluşturulan pencereler açılan HTML penceresinden bağımsız kalır. Örneğin, JavaScript Window `opener` ve `parent` özellikleri `null` değerindedir. Ancak, `createRootWindow()` yöntemini dolaylı olarak `createRootWindow()` çağıran `createWindow()` yöntemini geçersiz kılarak çağırırsanız, `opener` ve `parent` açılan HTML penceresine başvurur.*

HTMLLoader sınıfının alt sınıflarını oluşturma

Adobe AIR 1.0 ve üstü

Yeni davranışlar oluşturmak için, HTMLLoader sınıfının alt sınıflarını oluşturabilirsiniz. Örneğin, HTMLLoader olayları için varsayılan olay dinleyicilerini tanımlayan bir alt sınıf oluşturabilirsiniz (HTML oluşturulduğunda veya kullanıcı bir bağı tıklattığında gönderilen olaylar gibi).

Aşağıdaki örnek, JavaScript `window.open()` yöntemi çağrıldığında *normal* davranış sağlamak için HTMLHost sınıfını genişletir. Örnek daha sonra özel HTMLHost uygulama sınıfını kullanan bir HTMLLoader alt sınıfı tanımlar:

```
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
       }
}
```

Aşağıdaki örnek, `htmlHost` özelliğine bir MyHTMLHost nesnesi atayan HTMLLoader sınıfının alt sınıfını tanımlar:

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

Bu örnekte kullanılan HTMLHost sınıfı ve `HTMLLoader.createRootWindow()` yöntemine ilişkin ayrıntılar için bkz. "HTML içeriği için tarayıcı benzeri kullanıcı arabirimleri tanımlama" sayfa 1018.

Bölüm 61: AIR'de HTML bağlantılı olayları işleme

Adobe AIR 1.0 ve üstü

Olay işleme sistemi, programcıların kullanıcı girişlerine ve sistem olaylarına pratik bir şekilde yanıt vermesini sağlar. Adobe® AIR® olay modeli hem pratik, hem de standartlara uygundur. Belge Nesnesi Modeli (DOM) Düzey 3 Olaylar Tanımlaması'na dayalı olarak, endüstri standartlarında bir olay işleme mimarisi olan olay modeli, programcılara güçlü ve sezgisel bir olay işleme aracı sunar.

HTMLLoader olayları

Adobe AIR 1.0 ve üstü

HTMLLoader nesnesi aşağıdaki Adobe® ActionScript® 3.0 olaylarını gönderir:

Olay	Açıklama
`htmlDOMInitialize`	HTML belgesi oluşturulduğunda, ancak herhangi bir komut ayrıştırılmadan veya DOM düğümleri sayfaya eklenmeden gönderilir.
`complete`	HTML sayfasındaki `onload` olayından hemen sonra, HTML DOM bir yükleme işlemine yanıt olarak yaratıldığında gönderilir.
`htmlBoundsChanged`	`contentWidth` ve `contentHeight` özelliklerinden biri veya her ikisi de değiştiğinde gönderilir.
`locationChange`	HTMLLoader öğesinin konum özelliği değiştiğinde gönderilir.
`locationChanging`	Kullanıcı gezinmesi, JavaScript çağrısı veya yeniden yönlendirme nedeniyle HTMLLoader öğesinin konumu değişmeden önce gönderilir. `load()`, `loadString()`, `reload()`, `historyGo()`, `historyForward()` veya `historyBack()` yöntemlerini çağırdığınızda `locationChanging` olayı gönderilmez. Gönderilen olay nesnesinin `preventDefault()` yöntemini çağırmak gezinmeyi iptal eder. Sistem tarayıcısında bir bağlantı açılırsa HTMLLoader öğesi konumunu değiştirmediği için locationChanging olayı gönderilmez.
`scroll`	HTML motoru kaydırma konumunu her değiştirdiğinde gönderilir. Scroll olaylarının nedeni sayfadaki çapa linklerinde (`#` linkleri) gezinme veya `window.scrollTo()` yöntemine yapılan çağrılar olabilir. Bir metin girişine veya metin alanına metin girmek de scroll olayına neden olabilir.
`uncaughtScriptException`	HTMLLoader öğesinde bir JavaScript istisnası oluştuğunda ve istisna JavaScript kodunda yakalanmadığında gönderilir.

Bir ActionScript işlevini JavaScript olayı için de kaydedebilirsiniz (`onClick` gibi). Ayrıntılar için bkz. “DOM olaylarını ActionScript ile işleme” sayfa 1031.

DOM olaylarını ActionScript ile işleme

Adobe AIR 1.0 ve üstü

ActionScript işlevlerini JavaScript olaylarına yanıt vermesi için kaydedebilirsiniz. Örneğin aşağıdaki HTML içeriğini düşünün:

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

Bir ActionScript işlevini sayfadaki herhangi bir olay için işleyici olarak kaydedebilirsiniz. Örneğin aşağıdaki kod `clickHandler()` işlevini, HTML sayfasındaki `testLink` öğesinin `onclick` olayının dinleyicisi olarak ekler:

```
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 );
}
```

Gönderilen olay nesnesi flash.events.Event türünde veya Event alt sınıflarından değildir. Olay işleyicisi işlev argümanı için bir tür bildirmek için Object sınıfını kullanın.

Bu olaylar için kayıt yaparken `addEventListener()` yöntemini de kullanabilirsiniz. Örneğin önceki örnekteki `completeHandler()` yöntemini aşağıdaki kodla değiştirebilirsiniz:

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

Bir dinleyici belirli bir DOM öğesine başvurduğunda, olay dinleyicilerini eklemeden önce HTMLLoader üst öğesinin `complete` olayını göndermesini beklemek yerinde bir uygulama olur. HTML sayfaları genellikle birden çok dosya yükler ve tüm dosyalar yüklenip ayrıştırılana kadar HTML DOM tamamen oluşturulmaz. Tüm öğeler oluşturulduğunda, HTMLLoader `complete` olayını gönderir.

Yakalanmayan JavaScript istisnalarına yanıt verme

Adobe AIR 1.0 ve üstü

Aşağıdaki HTML'ye bakın:

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

`melbaToast` adlı bilinmeyen değişkene başvuran `throwError()` JavaScript işlevini içerir:

```
var x = 400 * melbaToast;
```

JavaScript işlemi, JavaScript kodunda `try/catch` yapısıyla yakalanmayan geçersiz bir işlemle karşılaştığında, sayfayı içeren HTMLLoader nesnesi bir HTMLUncaughtScriptExceptionEvent olayı gönderir. Aşağıdaki kodda olduğu gibi, bu olay için bir işleyici kaydedebilirsiniz:

```
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);
    }
}
```

JavaScript'in içinde, window.htmlLoader özelliğini kullanarak aynı olayı işleyebilirsiniz:

```
<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>
```

`htmlErrorHandler()` olay işleyicisi, olayın varsayılan davranışını (JavaScript hata mesajını AIR izleme çıktısına göndermeyi) iptal eder ve kendi çıktı mesajını oluşturur. HTMLUncaughtScriptExceptionEvent nesnesinin `exceptionValue` öğesinin değerini çıkarır. `stackTrace` dizisindeki her bir nesnenin özelliklerini çıkarır:

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

Çalışma zamanı olaylarını JavaScript'le işleme

Adobe AIR 1.0 ve üstü

runtime sınıfları, `addEventListener()` yöntemiyle olay işleyicileri eklemeyi destekler. Bir olay için işleyici işlevi eklemek üzere, olay türünü ve işleme işlevini sağlayarak olayı gönderen nesnenin `addEventListener()` yöntemini çağırın. Örneğin kullanıcı başlık çubuğunda pencereyi kapatma düğmesini tıkladığında gönderilen `closing` olayını dinlemek için, aşağıdaki ifadeyi kullanın:

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

Olay işleyici işlevi yaratma

Adobe AIR 1.0 ve üstü

Aşağıdaki kod, ana pencerenin konumu hakkında bilgi görüntüleyen basit bir HTML dosyası oluşturur. `moveHandler()` adlı bir işleyici işlevi, ana pencerenin move olayını dinler (NativeWindowBoundsEvent sınıfı tarafından tanımlanan).

```
<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>
```

Bir kullanıcı pencereyi taşıdığında, metin alanı öğeleri pencerenin güncellenen X ve Y konumlarını görüntüler.

Olay nesnesinin bir argüman olarak `moveHandler()` yöntemine iletildiğine dikkat edin. Olay parametresi işleyici işlevinizin olay nesnesini incelemesini sağlar. Bu örnekte olay nesnesinin `type` özelliğini, olayın bir `move` olayı olduğunu bildirmek için kullanırsınız.

Olay dinleyicilerini kaldırma

Adobe AIR 1.0 ve üstü

Artık ihtiyacınız olmayan bir olay dinleyicisini kaldırmak için `removeEventListener()` yöntemini kullanabilirsiniz. Artık kullanılmayacak olan olay dinleyicilerinin tümünü kaldırmak yararlı olacaktır. Gerekli parametreler `eventName` ve `listener` parametrelerini içerir, bunlar `addEventListener()` yöntemi için gereken parametrelerle aynıdır.

Gezinilen HTML sayfalarındaki olay dinleyicilerini kaldırma

Adobe AIR 1.0 ve üstü

HTML içeriği gezindiğinde veya içeriğe sahip pencere kapatıldığı için HTML içeriği atıldığında, boşaltılan sayfadaki nesnelere başvuran olay dinleyicileri otomatik olarak kaldırılmaz. Bir nesne, işleyiciye önceden boşaltılmış bir olay gönderdiğinde, şu hata mesajını görürsünüz: "Uygulama, artık yüklü olmayan bir HTML sayfasındaki JavaScript nesnesine başvurma girişiminde bulundu." (The application attempted to reference a JavaScript object in an HTML page that is no longer loaded.)

Bu hatayı önlemek için, atılmadan önce HTML sayfasındaki JavaScript olay dinleyicilerini kaldırın. Sayfa gezinmesi durumunda (HTMLLoader nesnesinin içinde), `window` nesnesinin `unload` olayı sırasında olay dinleyicisini kaldırın.

Örneğin aşağıdaki JavaScript kodu, `uncaughtScriptException` olayı için bir olay dinleyicisini kaldırır:

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

HTML içeriğine sahip pencereler kapatılırken hata oluşmasını önlemek için, NativeWindow nesnesinin (`window.nativeWindow`) `closing` olayına yanıt olarak bir temizleme işlevi çağırın. Örneğin aşağıdaki JavaScript kodu, `uncaughtScriptException` olayı için bir olay dinleyicisini kaldırır:

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

Ayrıca, bir olay dinleyicisini çalıştığı anda kaldırarak bu hatanın oluşmasını önleyebilirsiniz (olayın yalnızca bir defa işlenmesi gerekiyorsa). Örneğin aşağıdaki JavaScript kodu, HTMLLoader nesnesinin `createRootWindow()` yöntemini çağırarak bir html penceresi oluşturur ve `complete` olayı için bir olay dinleyicisi ekler. `complete` olay işleyicisi çağrıldığında, `removeEventListener()` işlevini kullanarak kendi olay dinleyicisini kaldırır:

```
 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"));
```

Gerekmeyen olay dinleyicilerini kaldırmak da sistem çöp toplayıcısının bu dinleyicilerle ilişkilendirilen tüm belleği geri istemesini sağlar.

Mevcut olay dinleyicilerini kontrol etme

Adobe AIR 1.0 ve üstü

`hasEventListener()` yöntemi sayesinde bir nesnede olay dinleyicisi olup olmadığını kontrol edebilirsiniz.

Bölüm 62: Mobil uygulamalarda HTML içeriğini görüntüleme

Adobe AIR 2.5 ve üstü

StageWebView sınıfı HTML içeriğini mobil aygıtlarda sistem tarayıcısı kontrolünü ve masaüstü bilgisayarlarda standart Adobe® AIR® HTMLLoader kontrolünü kullanarak görüntüler. Sınıfın geçerli aygıtta desteklenip desteklenmediğini belirlemek için `StageWebView.isSupported` özelliğini kontrol edin. Mobil profilinde tüm aygıtlar için destek garantilenmez.

Tüm profillerde, StageWebView sınıfı HTML içeriği ve uygulamanın geri kalanı arasındaki etkileşimlerin sınırlı bir kısmını destekler. Gezinmeyi kontrol edebilirsiniz ancak çapraz komut dosyasına veya doğrudan veri alış verişine izin verilmez. Yerel veya uzak bir URL'den içerik yükleyebilir ya da bir HTML dizesi sunabilirsiniz.

StageWebView nesneleri

Bir StageWebView nesnesi ekran nesnesi değildir ve görüntüleme listesine eklenemez. Bunun yerine, doğrudan sahne alanına bağlı bir görüntü kapısı olarak işlev görür. StageWebView içeriği çizim işlemini görüntüleme listesi içeriğinin üzerinde gerçekleştirir. Birden fazla StageWebView nesnesinin çizim sırasını kontrol etmenin bir yolu yoktur.

Bir StageWebView nesnesini görüntülemek için, nesnenin StageWebView nesnesinin `stage` özelliğine görüneceği sahne alanını atarsınız. `viewPort` özelliğini kullanarak ekranın boyutunu ayarlayın.

`viewPort` özelliğinin x ve y koordinatlarını -8192 ve 8191 arasında ayarlayın. Sahne alanı genişliği ve yüksekliğinin maksimum değeri 8191'dir. Boyut maksimum değerleri aşarsa bir istisna atılır.

Aşağıdaki örnek bir StageWebView nesnesi oluşturur, `stage` ve `viewPort` özelliklerini ayarlar ve bir HTML dizesi görüntüler:

```
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 );
```

Bir StageWebView nesnesini gizlemek için `stage` özelliğini `null` olarak ayarlayın. Nesneyi tamamen yok etmek için `dispose()` yöntemini çağırın. `dispose()` yöntemini çağırmak isteğe bağlıdır ancak çöp toplayıcısının nesne tarafından kullanılan belleği daha çabuk geri kazanmasına yardımcı olur.

İçerik

İçeriği bir StageWebView nesnesine iki yöntem kullanarak yükleyebilirsiniz: `loadURL()` ve `loadString()`.

`loadURL()` yöntemi belirtilen URL'de bir kaynak yükler. Şunlarda dahil olmak üzere sistem web tarayıcısı kontrolü tarafından desteklenen herhangi bir URI şemasını kullanabilirsiniz: data:, file:, http:, https: ve javascript:. app: ve app-storage: şemaları desteklenmez. AIR URL dizesini doğrulamaz.

`loadString()` yöntemi HTML içeriğine sahip değişmez bir dize yükler. Bu yöntemle yüklenen bir sayfanın konumu şu şekilde ifade edilir:

- Masaüstünde: about:blank
- iOS'ta: *htmlString*
- Android'de: kodlanmış *htmlString* öğesinin veri URI biçimi.

Gömülü içerik veya veri yükleme kurallarını URI şeması belirler.

URI şeması	Yerel kaynağı yükleme	Uzak kaynağı yükleme	Yerel XMLHttpRequest	Uzak XMLHttpRequest
data:	Hayır	Evet	Hayır	Hayır
file:	Evet	Evet	Evet	Evet
http:, https:	Hayır	Evet	Hayır	Aynı etki alanı
about: (loadString() method)	Hayır	Evet	Hayır	Hayır

Not: *Sahne alanının `displayState` özelliği `FULL_SCREEN` olarak ayarlanırsa Masaüstünde StageWebView öğesinde görüntülenen bir metin alanına yazamazsınız. Ancak, iOS ve Android'de sahne alanının `displayState` öğesi `FULL_SCREEN` olsa bile StageWebView öğesinde metin alanına yazabilirsiniz.*

Aşağıdaki örnek Adobe'nin web sitesini görüntülemek için bir StageWebView nesnesi kullanır:

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

Android aygıtlarında, uygulamanın uzak kaynakları başarıyla yüklemesini sağlamak için Android INTERNET iznini belirtmeniz gerekir.

Android 3.0 ve üstünde, bir StageWebView nesnesindeki eklenti içeriğini görüntülemek için uygulamanın, AIR uygulama tanımlayıcısının Android manifestAdditions öğesinde donanım hızlandırmayı etkinleştirmesi gerekir. Bkz. StageWebView nesnesinde Flash Player ve diğer eklentileri etkinleştirme.

JavaScript URI'si

StageWebView nesnesi tarafından yüklenen HTML sayfasında tanımlanan bir işlevi çağırmak için bir JavaScript URI'si kullanabilirsiniz. JavaScript URI'yi kullanarak çağırdığınız işlev yüklü web sayfasının bağlamında çalışır. Aşağıdaki örnek bir JavaScript işlevini çağırmak için StageWebView nesnesi kullanır:

```
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 );
        }
    }
}
```

Daha fazla Yardım konusu

Sean Voisen: Making the Most of StageWebView (StageWebView Öğesinden En İyi Şekilde Faydalanma)

Gezinme olayları

Bir kullanıcı HTML'de bir bağlantıyı tıklattığında, StageWebView nesnesi bir `locationChanging` olayı gönderir. Gezinmeyi durdurmak için olay nesnesinin `preventDefault()` yöntemini çağırabilirsiniz. Aksi takdirde, StageWebView nesnesi yeni sayfaya gider ve bir `locationChange` olayı gönderir. Sayfanın yüklenmesi tamamlandığında, StageWebView bir `complete` olayı gönderir.

Her HTML yeniden yönlendirmesinde bir `locationChanging` olayı gönderilir. `locationChange` ve `complete` olayları uygun zamanda gönderilir.

iOS'da ilk `loadURL()` veya `loadString()` yöntemleri dışında bir `locationChange` olayından önce `locationChanging` olayı gönderilir. iFrames ve Frames aracılığıyla gezinme değişiklikleri için de bir `locationChange` olayı gönderilir.

Aşağıdaki örnek bir konum değişikliğini nasıl önleyebileceğinizi ve bunun yerine yeni sayfayı sistem tarayıcısında nasıl açacağınızı gösterir.

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

Geçmiş

Kullanıcı StageWebView nesnesinde görüntülenen içerikte bağlantıları tıklatırken kontrol geriye doğru ve ileri doğru geçmiş yığınlarını kaydeder. Aşağıdaki örnek iki geçmiş yığını arasında nasıl gezileceğini gösterir. Örnek Geri ve Ara yazılım tuşlarını kullanır.

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

Odak

Görüntüleme nesnesi olmasa da, StageWebView sınıfı denetime giren ve denetimden çıkan odak geçişlerini yönetmenize izin veren üyeler içerir.

StageWebView nesnesi odak kazandığında bir `focusIn` olayı gönderir. Bu olayı gerekirse uygulamanızdaki odak öğelerini yönetmek için kullanabilirsiniz.

StageWebView nesnesi odağı bıraktığında bir `focusOut` olayı gönderir. Bir StageWebView örneği kullanıcı web sayfasındaki ilk veya son kontrolü aygıt izleme topu veya yön okları kullanarak sekme tuşu ile geçtiğinde odağı bırakabilir. Olay nesnesinin `direction` özelliği odak akışının sayfanın üstünden geçerek arttığını mı yoksa sayfanın altından geçerek azaldığını mı bildirir. Bu bilgileri odağı StageWebView nesnesinin üstünde veya altında uygun görüntüleme nesnesine atamak için kullanın.

iOS'ta odak programlı olarak ayarlanamaz. StageWebView `FocusEvent` öğesinin yön özelliği `none` olarak ayarlıyken `focusIn` ve `focusOut` olaylarını gönderir. Kullanıcı StageWebView öğesinin içine dokunursa `focusIn` olayı gönderilir. Kullanıcı StageWebView öğesinin dışına dokunursa `focusOut` olayı gönderilir.

Aşağıdaki örnek odağın StageWebView nesnesinden Flash görüntülüme nesnelerine nasıl geçtiğini gösterir:

```
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;
        }

    }
}
```

Bitmap yakalama

Bir StageWebView nesnesi tüm görüntüleme listesi içeriğinin üstünde oluşturulur. Bir StageWebView nesnesinin üstüne içerik ekleyemezsiniz. Örneğin, bir açılır listeyi StageWebView içeriğinin üzerine genişletemezsiniz. Bu sorunu çözmek için StageWebView öğesinin anlık görüntüsünü alın. Ardından, StageWebView öğesini gizleyin ve bunun yerine bitmap anlık görüntüyü ekleyin.

Aşağıdaki örnek `drawViewPortToBitmapData` yöntemini kullanarak bir StageWebView nesnesinin anlık görüntüsünün nasıl alınacağını gösterir. Sahne alanını null olarak ayarlayarak StageWebView nesnesini gizler. Web sayfası tamamen yüklendikten sonra bitmap'i yakalayan ve görüntüleyen bir işlev çağırır. Çalıştırdığınızda kod iki etiket gösterir: Google ve Facebook. Etiketi tıklatmak karşılık gelen web sayfasını yakalar ve bunu sahne alanında anlık görüntü olarak gösterir.

```
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);
        }
    }
}
```

Bölüm 63: Eş zamanlılık için worker (işçi) öğelerini kullanma

Masaüstü platformları için Flash Player 11.4 ve sonraki sürümleri, Adobe AIR 13.4 ve sonraki sürümleri

ActionScript worker (işçi) öğeleri, kodun eş zamanlı olarak yürütülmesine, başka bir deyişle, kodun ana kodun yürütülmesini kesintiye uğratmadan arka planda yürütülmesine olanak tanır.

ActionScript eş zamanlılık api'leri, masaüstü platformları için yalnızca Flash Player 11.4 ve sonraki sürümleri ile AIR 3.4 ve sonraki sürümlerinde kullanılabilir. Eş zamanlılık, AIR uygulamasında mobil platformlar için desteklenmez.

Worker (işçi) öğelerini ve eş zamanlılığı anlama

Masaüstü platformları için Flash Player 11.4 ve sonraki sürümleri, Adobe AIR 13.4 ve sonraki sürümleri

Bir uygulama worker (işçi) öğelerini kullanmadığında, uygulamanın kodu, yürütme *iş parçacığı* olarak bilinen tek bir doğrusal yürütme adımları bloğunda yürütülür. İş parçacığı, geliştiricinin yazdığı kodu yürütür. Ayrıca, başta görüntüleme nesnelerinin özellikleri değiştiğinde ekranı güncelleyen kod olmak üzere, çalışma zamanının parçası olan kodun büyük kısmını da yürütür. Kod, yöntem ve sınıf olarak öbekler halinde yazılsa da çalışma zamanında kod, tek bir uzun adımlar dizisi olarak yazılmış gibi bir defada bir satır yürütür. Bir uygulamanın yürüttüğü adımlara ilişkin şu varsayımsal örneği ele alalım:

1. Kareye gir: Çalışma zamanı, herhangi bir sayıda `enterFrame` olay işleyicisi çağırır ve kodlarını tek tek çalıştırır
2. Fare olayı: Kullanıcı fareyi hareket ettirir ve çalışma zamanı, çeşitli üzerine gelme ve üzerinden gitme olayları oluştukça herhangi bir sayıda fare olay işleyicisini çağırır.
3. Yükleme tamamlandı olayı: Bir url'den xml dosyası yükleme isteği, yüklü dosya verileriyle döner. Olay işleyicisi çağrılır ve xml içeriğini okuyarak ve xml verilerinden bir nesne kümesi oluşturarak adımlarını çalıştırır.
4. Fare olayı: Fare tekrar hareket ettirildiğinden, çalışma zamanı ilgili fare olayı işleçlerini çağırır.
5. Görüntü oluşturma: Artık bekleyen olay kalmadığından, çalışma zamanı ekranı görüntüleme nesnelerinde yapılan değişikliklere göre günceller.
6. Kareye gir: Döngü tekrar başlar

Örnekte açıklandığı gibi, 1-5 arası varsayımsal adımlar, kare adı verilen tek bir zaman bloğunda sırayla çalıştırılır. Bunlar tek bir iş parçacığında sırayla çalıştırıldığı için, çalışma zamanı, işlemin bir adımını, başka bir adımını çalıştırmak üzere kesintiye uğratamaz. 30 kare/saniye değerindeki kare hızıyla, çalışma zamanı, tüm bu işlemleri yürütmek için saniyenin otuzda birinden az zamana sahiptir. Birçok durumda bu zaman, kodun çalıştırılması için yeterlidir ve çalışma zamanı kalan zaman boyunca yalnızca bekler. Ancak, adım 3'te yüklenen xml verilerinin çok büyük ve derin yuvalanmış bir xml yapısı olduğunu varsayalım. Kod, xml üzerinde döngü uyguladıkça ve nesneler oluşturdukça, bu işin yapılmasının saniyenin otuzda birinden uzun sürmesi olasıdır. Bu durumda, sonraki adımlar (fareye yanıt vermek ve ekranı yeniden çizmek) olması gereken hızda gerçekleştirilmez. Bu, kullanıcının fareyi hareket ettirmesine yanıt olarak ekran gereken hızda yeniden çizilmediğinden, ekranın donmasına ve takılmasına neden olur.

Kodun tamamı aynı iş parçacığında yürütülürse, zaman zaman takılma ve donma yaşanmasını önlemenin yalnızca bir yolu vardır. Bu yol, büyük bir veri kümesi üzerinde döngü uygulamak gibi uzun süre çalışan işlemleri yapmamaktır. ActionScript worker (işçi) öğeleri de bir başka çözüm sunmaktadır. Worker (işçi) kullanarak, uzun süre çalışan kodu ayrı bir worker (işçi) öğesinde çalıştırabilirsiniz. Her bir worker (işçi) ayrı bir iş parçacığında çalışır, böylece arka plandaki worker (işçi) öğesi, uzun süre çalışan işlemi kendi iş parçacığında gerçekleştirir. Bu durum, ana worker (işçi) öğesinin yürütme iş parçacığını, diğer işler tarafından engellenmeden her karede ekranı yeniden çizebilmesi için boşaltır.

Bu yolla aynı anda birden çok kod işlemi çalıştırma yeteneğine *eş zamanlılık* adı verilir. Arka plandaki worker (işçi) işini tamamladığında veya işlem boyunca "ilerleme" noktalarında olduğunda, ana worker (işçi) bildirimlerini ve verilerini gönderebilirsiniz. Bu şekilde, karmaşık veya zaman alan işlemler gerçekleştiren kodlar yazabilir ve aynı zamanda ekran donması gibi kötü bir kullanıcı deneyiminin önüne geçebilirsiniz.

Worker (işçi) öğeleri, ana görüntü oluşturma iş parçacığının başka bir kod tarafından engellenmesi nedeniyle kare hızının düşmesi olasılığını azalttığı için faydalıdır. Ancak, worker (işçi) öğeleri ek sistem belleği ve işlemci kullanımı gerektirir ve bu da genel uygulama performansı için maliyetli olabilir. Her worker (işçi) kendi çalışma zamanı sanal makinesi örneğini kullandığından, önemsiz bir worker (işçi) öğesinin ek yükü bile büyük olabilir. Worker (işçi) öğelerini kullanırken, kodunuzu tüm hedef platformlarınızda test ederek sisteme yüklenen taleplerin çok fazla olmadığından emin olun. Adobe, tipik bir senaryoda bir veya ikiden fazla arka plan worker (işçi) öğesi kullanmamanızı önerir.

Worker (işçi) öğeleri oluşturma ve yönetme

Masaüstü platformları için Flash Player 11.4 ve sonraki sürümleri, Adobe AIR 13.4 ve sonraki sürümleri

Eş zamanlılık için worker (işçi) kullanmanın ilk adımı, bir arka plan worker (işçi) öğesi oluşturmaktır. Bir worker (işçi) oluşturmak için iki tür nesne kullanırsınız. Bunlardan ilki, oluşturduğunuz öğe olan Worker örneğidir. Diğeri ise, Worker öğesini oluşturan ve bir uygulamada çalışan Worker nesnelerini yöneten WorkerDomain nesnesidir.

Çalışma zamanı yüklendiğinde, WorkerDomain nesnesini otomatik olarak oluşturur. Çalışma zamanı ayrıca, uygulamanın ana swf'si için otomatik olarak bir worker (işçi) öğesi oluşturur. İlk worker (işçi) öğesine *esas worker (işçi)* adı verilir.

Bir uygulama için yalnızca bir WorkerDomain nesnesi bulunduğundan, WorkerDomain örneğine statik `WorkerDomain.current` özelliğini kullanarak erişebilirsiniz.

Geçerli Worker örneğine (geçerli kodun çalıştığı worker (işçi)) istediğiniz zaman statik `Worker.current` örneğini kullanarak erişebilirsiniz.

Bir swf'den Worker nesnesi oluşturma

Ana swf'nin esas worker (işçi) öğesiyle çalışması gibi, bir arka plan worker (işçi) öğesi de tek bir swf dosyasının kodunu yürütür. Bir arka plan worker (işçi) öğesini kullanmak için, worker (işçi) öğesinin kodunu swf dosyası olarak yazmanız ve derlemeniz gerekir. Arka plan worker (işçi) öğesi oluşturmak için, üst worker (işçi) öğesinin söz konusu swf dosyasının baytlarına ByteArray nesnesi olarak erişmesi gerekir. Worker (işçi) öğesini gerçekten oluşturmak için ByteArray nesnesini WorkerDomain nesnesinin `createWorker()` yöntemine iletirsiniz.

Arka plan worker (işçi) swf'sini ByteArray nesnesi olarak almanın üç ana yolu mevcuttur:

Worker (işçi) swf'sini gömme

Worker (işçi) swf'sini ana swf'nin içine ByteArray olarak gömmek için [Embed] meta etiketini kullanın:

```
[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
}
```

Worker (işçi) swf'si ana swf'nin içine BgWorker_ByteClass adlı bir ByteArray alt sınıfı olarak derlenir. Bu sınıfın bir örneğinin oluşturulması size, worker (işçi) swf'sinin baytlarıyla önceden doldurulmuş bir ByteArray sağlar.

Harici bir worker (işçi) swf'si yükleme

Harici bir swf dosyasını yüklemek için bir URLLoader nesnesi kullanın. Söz konusu swf dosyası, ana swf ile aynı internet etki alanından yüklenmiş veya bir AIR uygulama paketine dahil edilmiş bir swf dosyası gibi aynı güvenlik etki alanından gelmelidir.

```
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
}
```

URLLoader, swf dosyasını yüklemeyi tamamladığında, swf'nin baytları URLLoader nesnesinin `data` özelliğinde (örnekte `event.target.data`) kullanılabilir hale gelir.

Ana swf'yi worker (işçi) swf'si olarak kullanma

Tek bir swf'yi hem ana swf hem de worker (işçi) swf'si olarak kullanabilirsiniz. Ana görüntüleme sınıfının `loaderInfo.bytes` özelliğini, swf'nin baytlarına erişmek için kullanın.

```
// 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
    }
}
```

Bu tekniği kullanıyorsanız, ana sınıfın oluşturucusu veya çağırdığı bir yöntem içindeki swf dosya kodunu kollara ayırmak için bir `if` ifadesi kullanın. Kodun ana worker (işçi) öğesinde mi, yoksa arka plan worker (işçi) öğesinde mi çalıştığını belirlemek için, örnekte gösterildiği şekilde, geçerli Worker nesnesinin `isPrimordial` özelliğini denetleyin.

Worker (işçi) öğesinin yürütülmesini başlatma

Bir worker (işçi) oluşturduktan sonra, bu worker (işçi) öğesinin kod yürütme işlemini, Worker nesnesinin `start()` yöntemini çağırarak başlatırsınız. `start()` işlemi hemen gerçekleştirilmez. Worker (işçi) çalıştırıldığında haberdar olmak için, Worker nesnesinin `workerState` olayı için bir dinleyici kaydedin. Bu olay, Worker nesnesi yaşam döngüsünde durumunu değiştirdiğinde; örneğin kod yürütmeye başladığında gönderilir. `workerState` olay işleyicinizde, Worker nesnesinin `state` özelliğinin `WorkerState.RUNNING` olup olmadığını denetleyin. Bu noktada, worker (işçi) çalışır durumdadır ve ana sınıfının oluşturucusu çalıştırılmıştır. Aşağıdaki kod listesi, `workerState` olayını kaydetmeye ve `start()` yöntemini çağırmaya yönelik bir örneği göstermektedir:

```
// 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.
    }
}
```

Worker (işçi) yürütmeyi yönetme

WorkerDomain sınıfının `listWorkers()` yöntemini kullanarak, istediğiniz zaman uygulamanızda çalışan worker (işçi) kümesine erişebilirsiniz. Bu yöntem, esas worker (işçi) öğesi de dahil olmak üzere, `state` özelliği `WorkerState.RUNNING` olan worker (işçi) kümesini döndürür. Bir worker (işçi) başlatılmamışsa veya yürütülmesi durdurulmuşsa, bu worker (işçi) dahil edilmez.

Bir worker (işçi) öğesine artık gereksinim duymuyorsanız, Worker nesnesinin `terminate()` yöntemini worker (işçi) öğesini kapatması ve belleği ile diğer sistem kaynaklarını boşaltması için çağırabilirsiniz.

Worker (işçi) öğeleri arasında iletişim

Masaüstü platformları için Flash Player 11.4 ve sonraki sürümleri, Adobe AIR 13.4 ve sonraki sürümleri

Worker (işçi) öğeleri kodlarını ayrı yürütme iş parçacıklarında çalıştırsa da birbirlerinden tamamen izole olmuş olsalardı, hiçbir fayda sağlayamazlardı. Worker (işçi) öğeleri arasındaki iletişim, temelde verilerin worker (işçi) öğeleri arasında geçirilmesi anlamına gelir. Bir worker (işçi) öğesinden diğerine veri aktarımını sağlayan üç ana mekanizma bulunur.

Belirli bir veri aktarımı ihtiyacı için hangi veri paylaşımı tekniklerinin uygun olduğuna karar verirken, farklılık gösterdikleri iki durumu dikkate alın. Bunların arasındaki bir fark, yeni verinin kullanılabilir olduğuna dair alıcıyı uyaran bir olay olup olmadığı veya alıcı worker (işçi) öğesinin güncelleme kontrolü yapmasının gerekip gerekmediğidir. Bu veri paylaşım teknikleri arasındaki diğer bir fark da verilerin fiilen iletilmesi ile ilişkilidir. Bazı durumlarda alıcı worker (işçi) paylaşılan verinin bir kopyasını alır, bu da daha fazla nesne oluşturulduğu ve bununla birlikte daha fazla bellek ve işlemci döngüsü kullanıldığı anlamına gelir. Diğer durumlarda worker (işçi) öğeleri aynı temel sistem belleğine başvuran nesnelere erişir, bu da daha az nesne oluşturulduğu ve toplamda daha az bellek kullanıldığı anlamına gelir. Bu farklar burada vurgulanmıştır:

İletişim tekniği	**Veri alırken olayı gönderir**	**Worker (işçi) öğeleri arasında bellek paylaşır**
Worker (İşçi) paylaşımlı özellikler	Hayır	Hayır, nesneler kopyalardan oluşuyor, referanslardan değil
MessageChannel	Evet	Hayır, nesneler kopyalardan oluşuyor, referanslardan değil
Paylaşılabilir ByteArray	Hayır	Evet, bellek paylaşılıyor

Verileri paylaşılan özellikle iletme

Worker (işçi) öğeleri arasında veri paylaşmanın en temel yolu, paylaşılan özellik kullanmaktır. Her worker (işçi), paylaşılan özellik değerlerine yönelik dahili bir sözlük tutar. Özellikler birbirinden ayırt edilebilmesi için Dize anahtar adlarıyla saklanır. Bir worker (işçi) üzerindeki bir nesneyi paylaşılan özellik olarak saklamak için, saklanacak anahtar adı ve değerden oluşan iki argümanla Worker nesnesinin `setSharedProperty()` yöntemini çağırın:

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

Paylaşılan özellik ayarlandıktan sonra, değer, anahtar adı iletilecek şekilde Worker nesnesinin `getSharedProperty()` yöntemini çağırarak okunabilir:

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

Hangi worker (işçi) öğesinin özellik değerini okuyacağı veya ayarlayacağı konusunda bir kısıtlama yoktur. Örneğin, arka plan worker (işçi) öğesindeki kod, bir değeri saklamak için `setSharedProperty()` yöntemini çağırabilir. Üst worker (işçi) öğesinde çalışan kod, verileri almak için `getSharedProperty()` öğesini kullanabilir.

`setSharedProperty()` yöntemine iletilen değer, neredeyse her türden bir nesne olabilir. `getSharedProperty()` yöntemini çağırdığınızda, döndürülen nesne, birkaç özel durum dışında, `setSharedProperty()` öğesine iletilen nesnenin bir kopyasıdır ve aynı nesneye yönelik bir başvuru değildir. Verilerin nasıl paylaşıldığına ilişkin ayrıntılar, "Paylaşılan başvurular ve kopyalanan değerler" sayfa 1051 konusunda açıklanmıştır.

Worker (işçi) öğeleri arasında veri iletmek üzere paylaşılan özellik kullanmanın en büyük avantajı, worker (işçi) çalıştırılmadan önce bile kullanılabilir olmasıdır. Worker (işçi) henüz çalıştırılmadan, paylaşılan özellik ayarlamak üzere arka plan Worker nesnesinin `setSharedProperty()` yöntemini çağırabilirsiniz. Üst worker (işçi) öğesi, Worker nesnesinin `start()` yöntemini çağırır; çalışma zamanı, alt worker (işçi) öğesinin ana sınıfının oluşturucusunu çağırır. `start()` çağrılmadan önce ayarlanan tüm paylaşılan özellikler, alt worker (işçi) öğesinin okuması amacıyla kod için kullanılabilir durumdadır.

Verileri MessageChannel ile iletme

Mesaj kanalı, iki worker (işçi) öğesi arasında tek yönlü bir veri iletme bağlantısı sağlar. Worker (işçi) öğeleri arasında veri geçirmek için MessageChannel nesnesinin kullanılması tek bir önemli avantaj sunar. Bir mesaj kanalını kullanarak mesaj (nesne) gönderdiğinizde, MessageChannel nesnesi bir `channelMessage` olayı gönderir. Alıcı worker (işçi) öğesindeki kod, veriler kullanılabilir olduğunda haberdar olmak için bu olayı dinleyebilir. Bu şekilde, alıcı worker (işçi) öğesinin sürekli olarak veri güncellemelerini denetlemesi gerekmez.

Bir mesaj kanalı, bir gönderen ve bir alıcı olmak üzere yalnızca iki worker (işçi) öğesiyle ilişkilidir. Bir MessageChannel nesnesi oluşturmak için, alıcı worker (işçi) öğesini bir argüman olarak geçirerek, gönderen Worker nesnesinin `createMessageChannel()` yöntemini çağırın.

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

Her iki worker (işçi) öğesinin de MessageChannel nesnesine erişimi olmalıdır. Bunu yapmanın en basit yolu, MessageChannel nesnesini `setSharedProperty()` yöntemini kullanarak iletmektir:

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

Alıcı worker (işçi) öğesinde, MessageChannel nesnesinin `channelMessage` olayı için bir dinleyici kaydedin. Bu olay, gönderen worker (işçi) öğesi, mesaj kanalı üzerinden veri gönderdiğinde gönderilir.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

Verileri gerçekten göndermek için, gönderen worker (işçi) öğesi, MessageChannel nesnesinin `send()` yöntemini çağırır:

```
// In the sending worker swf
sendChannel.send("This is a message");
```

Alıcı worker (işçi) öğesinde, MessageChannel, `channelMessage` olay işleyicisini çağırır. Böylece alıcı worker (işçi) öğesi, verileri MessageChannel nesnesinin `receive()` yöntemini çağırarak alabilir.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

Alma yöntemi tarafından döndürülen nesne, `send()` yöntemine iletilen nesneyle aynı veri türüne sahiptir. Alınan nesne, "Paylaşılan başvurular ve kopyalanan değerler" sayfa 1051 konusunda açıklandığı şekilde, birkaç veri türünden birine ait olmadığı sürece, gönderen tarafından iletilen nesnenin bir kopyasıdır ve gönderen worker (işçi) öğesindeki nesneye yönelik bir başvuru değildir.

Paylaşılabilir ByteArray kullanarak veri paylaşımı

Bir nesnenin iki worker (işçi) öğesi arasında iletilmesi sırasında, alıcı worker (işçi) orijinal nesnenin bir kopyası olan yeni bir nesne alır. İki nesne sistem belleğinde farklı yerlerde saklanır. Sonuç olarak, nesneye ait alınan her bir kopya çalışma zamanı tarafından kullanılan toplam belleği arttırır. Buna ek olarak, bir worker (işçi) öğesindeki nesne üzerinde yapacağınız herhangi bir değişiklik diğer bir worker (işçi) öğesinde bulunan kopyayı etkilemez. Verilerin nasıl kopyalandığına dair daha ayrıntılı bilgi için "Paylaşılan başvurular ve kopyalanan değerler" sayfa 1051 bölümüne göz atın.

ByteArray nesnesi varsayılan olarak aynı davranışı kullanır. Worker (işçi) nesnesinin `setSharedProperty()` yöntemine veya MessageChannel nesnesinin `send()` yöntemine bir ByteArray örneği gönderildiğinde, çalışma zamanı bilgisayarın belleğinde yeni bir ByteArray oluşturur ve alıcı worker (işçi) bu yeni ByteArray öğesine referans olacak bir ByteArray örneği alır. Ancak, bir ByteArray nesnesi için bu davranışı nesneye ait `shareable` özelliğini `true` olarak ayarlayarak değiştirebilirsiniz.

Paylaşılabilir bir ByteArray nesnesi bir worker (işçi) öğesinden diğerine iletilirken, alıcı worker (işçi) öğesindeki ByteArray örneği, gönderici worker (işçi) öğesindeki ByteArray örneği tarafından kullanılan aynı temel işletim sistemi belleğine referans oluşturur. Bir worker (işçi) öğesindeki kod, bayt dizisindeki içeriği değiştirdiğinde, bu değişiklikler paylaşılan bayt dizisine erişimi olan diğer tüm worker (işçi) öğelerinde de anında kullanılabilir duruma gelir.

Worker (işçi) öğeleri kodlarını eş zamanlı olarak yürüttüğünden, iki worker (işçi) öğesinin bir bayt dizisindeki aynı baytlara aynı anda erişmeye çalışması mümkündür. Bu durum veri kaybına veya bozulmasına neden olabilir. Paylaşılan kaynaklara erişimi yönetmek ve bu sorunların önüne geçmek için kullanabileceğiniz birden çok API bulunur.

ByteArray sınıfı, bayt dizisinin içeriğini tek işlemde doğrulamanıza ve değiştirmenize olanak sağlayan yöntemler içerir:

- atomicCompareAndSwapIntAt() method
- atomicCompareAndSwapLength() method

Ek olarak, flash.concurrent paketi paylaşılan kaynaklarla çalışmak için erişim kontrolü sağlayan sınıflar içerir:

- Mutex sınıfı
- Condition sınıfı

Paylaşılan başvurular ve kopyalanan değerler

Normal durumda, `Worker.setSharedProperty()` veya `MessageChannel.send()` öğesini çağırırsanız, alıcı worker (işçi) öğesine iletilen nesne, AMF formatında serileştirilerek iletilir. Bu, birkaç sonucu beraberinde getirir:

- `getSharedProperty()` yöntemi çağrıldığında alıcı worker (işçi) öğesinde oluşturulan nesnenin AMF baytlarından serileştirmesi kaldırılır. Bu, orijinal nesnenin bir kopyasıdır, nesneye yönelik bir başvuru değildir. İki worker (işçi) öğesinden herhangi birinde nesne üzerinde yapılan değişiklikler, diğer worker (işçi) öğesindeki kopyada uygulanmaz.
- Görüntüleme nesneleri gibi AMF formatında serileştirilemeyen nesneler, `Worker.setSharedProperty()` veya `MessageChannel.send()` kullanılarak bir worker (işçi) öğesine iletilemez.
- Özel bir sınıfın doğru şekilde serileştirmesinin kaldırılması için, sınıf tanımının `flash.net.registerClassAlias()` işlevi veya `[RemoteClass]` meta verileri kullanılarak kaydedilmesi gerekir. Her iki worker (işçi) öğesinin sınıf sürümleri için de aynı başka ad kullanılmalıdır.

Worker (işçi) öğeleri arasında kopyalanmak yerine gerçek anlamda paylaşılan beş özel nesne durumu bulunur:

- Worker nesneleri
- MessageChannel nesneleri
- paylaşılabilir bayt dizisi (`shareable` özelliği `true` olan bir ByteArray nesnesi)
- Mutex nesneleri
- Condition nesneleri

Bu nesnelerden birine ait bir örneği `Worker.setSharedProperty()` yöntemini veya `MessageChannel.send()` yöntemini kullanarak iletirseniz, worker (işçi) öğelerinin her biri aynı alttaki nesneye yönelik bir başvuruya sahip olur. Bir worker (işçi) öğesinde örnek üzerinde yapılan değişiklikler, anında diğer worker (işçi) öğelerinde de uygulanır. Ayrıca, bu nesnelerden birine ait aynı örneği bir worker (işçi) öğesine bir defadan çok iletmeniz durumunda, çalışma zamanı, alıcı worker (işçi) öğesinde nesnenin yeni bir kopyasını oluşturmaz. Bunun yerine, aynı başvuru yeniden kullanılır.

Ek veri paylaşımı teknikleri

Worker (işçi) öğesine özel veri geçirme mekanizmalarına ek olarak, worker (işçi) öğeleri iki swf uygulaması arasında veri paylaşımını destekleyen şu gibi varolan api'lerden herhangi birini kullanarak da veri alışverişi yürütebilir:

- yerel paylaşılan nesneler
- bir worker (işçi) öğesindeki bir dosyaya veri yazma ve başka bir worker (işçi) öğesindeki bir dosyadan okuma
- SQLite veritabanı üzerinde veri saklama ve veri okuma işlemlerini yapma

Bir kaynağı iki veya daha fazla worker (işçi) öğesi arasında paylaştığınızda, genellikle birden fazla worker (işçi) öğesinin aynı anda kaynağa erişmesini önlemeniz gerekir. Örneğin, birden çok worker (işçi) öğesinin yerel dosya sistemi üzerindeki bir dosyaya erişmesi, veri kaybına veya bozulmasına neden olabilir ve işletim sistemi tarafından desteklenmeyebilir.

Eşzamanlı erişim problemlerine karşı önlem almak ve paylaşılan kaynaklarla çalışmak için erişim kontrolü sağlamak üzere flash.concurrent paketindeki Mutex ve Condition sınıflarını kullanın.

Diğer veri paylaşma mekanizmalarının aksine, SQLite veritabanı motoru, eş zamanlı erişim için tasarlanmıştır ve yerleşik olarak kendi işlem desteğine sahiptir. Birden fazla worker (işçi) öğesi, verileri bozma riski olmadan bir SQLite veritabanına erişebilir. Worker (işçi) öğeleri farklı SQLConnection örnekleri kullandığından, her worker (işçi) veritabanına ayrı bir işlem içinde erişir. Aynı anda gerçekleştirilen veri düzenleme işlemleri, verilerin bütünlüğünü etkilemez.

Ayrıca bkz.

"AIR'de yerel SQL veritabanlarıyla çalışma" sayfa 722
flash.concurrent package

Bölüm 64: Güvenlik

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Adobe, kullanıcılar, web sitesi sahipleri ve içerik geliştiricileri için güvenlik önemli bir konudur. Bu nedenle, Adobe® Flash® Player ve Adobe®AIR™ kullanıcıyı, web sitesi sahibini ve içerik geliştiricisini korumak için bir güvenlik kuralları ve denetimler kümesi içerir. Bu tartışma, aksi belirtilmediği sürece ActionScript 3.0'da yayınlanmış ve Flash Player 9.0.124.0 veya üzerinde çalıştırılan SWF dosyaları ile AIR 1.0 veya üzerinde çalıştırılan SWF, HTML ve JavaScript dosyaları için güvenlik modeli hakkındadır.

Bu bölüm güvenlik konusuna bir genel bakış sağlar; belirli API'lerin kullanımına yönelik tüm uygulama ayrıntılarını, kullanım senaryolarını veya sonuçlarını ayrıntılı şekilde içermeye çalışmaz. Flash Player güvenliği kavramlarının daha ayrıntılı bir açıklaması için www.adobe.com/go/devnet_security_tr adresinde "Güvenlik" başlıklı Flash Player Developer Center konusuna bakın.

Flash Platform güvenliğine genel bakış

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve AIR çalışma zamanlarında kullanılan güvenlik modelinin çoğunun temelinde, yüklenen SWF dosyaları, HTML, medya ve diğer varlıklar için orijin alanı bulunur. www.example.com gibi belirli bir internet alanından gelen bir dosyadaki yürütülebilir kod her zaman o alandaki tüm verilere erişebilir. Bu varlıklar, *güvenlik sanal alanı* olarak bilinen aynı güvenlik grubuna koyulur. (Daha fazla bilgi için, bkz. "Güvenlik sanal alanları" sayfa 1055.)

Örneğin bir SWF dosyasındaki ActionScript kodu SWF dosyalarını, bitmap'leri, ses dosyalarını, metin dosyalarını ve diğer tüm varlıkları kendi alanından yükleyebilir. Ayrıca, her iki dosya da ActionScript 3.0. kullanılarak yazılmışsa, aynı alandaki iki SWF dosyası arasında çapraz komut dosyası oluşturulmasına her zaman izin verilir.*Çapraz komut dosyası oluşturma*, bir dosyadaki bir kodun başka bir dosyadaki özelliklere, yöntemlere ve nesnelere erişim becerisidir.

ActionScript 3.0 kullanılarak yazılmış SWF dosyaları ile önceki ActionScript sürümleri kullanılarak yazılmış SWF dosyaları arasında çapraz komut dosyası oluşturmaya izin verilmez; ancak bu dosyalar LocalConnection sınıfını kullanarak iletişim kurabilir. Ayrıca SWF dosyasının başka etki alanlarındaki ActionScript 3.0 SWF dosyalarının çapraz komut dosyasını oluşturması ve başka etki alanlarından veri yüklemesi yeteneği de varsayılan olarak yasaklanmıştır; ancak yüklenen SWF dosyasında `Security.allowDomain()` öğesine yapılan bir çağrıyla bu erişime izin verilebilir. Daha fazla bilgi için bkz. "Çapraz komut dosyası oluşturma" sayfa 1074.

Aşağıdaki temel güvenlik kuralları her zaman varsayılan olarak geçerlidir:

- Aynı güvenlik sanal alanında bulunan kaynaklar her zaman birbirine erişebilir.
- Uzak bir sanal alanda bulunan SWF dosyaları asla yerel dosyalar ile verilere erişemez.

Flash Player ve AIR çalışma zamanları, aşağıdakileri bağımsız alanlar olarak değerlendirir ve her biri için bağımsız güvenlik sanal kutuları kurar:

- `http://example.com`
- `http://www.example.com`
- `http://store.example.com`

- `https://www.example.com`
- `http://192.0.34.166`

http://example.com gibi adlandırılmış bir alan, http://192.0.34.166 gibi belirli bir IP adresiyle eşlense bile, çalışma zamanı her biri için ayrı güvenlik sanal alanı kurar.

Geliştiricinin bir SWF dosyasına kendi sanal alanı dışındaki sanal alanlarda bulunan varlıklara erişim hakkı vermek için kullanabileceği iki temel yöntem vardır:

- `Security.allowDomain()` yöntemi (bkz. "Yazar (geliştirici) denetimleri" sayfa 1066)
- URL ilke dosyası (bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062)

Flash Player ve AIR çalışma zamanı güvenlik modellerinde içerik yükleme ile verileri ayıklama veya verilere erişme arasında bir ayrım vardır. *İçerik*, çalışma zamanının görüntüleyebildiği görsel medyalar ve görüntülenen ortamın içerdiği ses, video veya SWF dosyalarının bulunduğu medya olarak tanımlanır. *Veri*, yalnızca kodu tarafından erişilebilen herhangi bir şey olarak tanımlanır. İçerik ve veriler farklı şekillerde yüklenir.

- Yükleme içeriği — İçeriği Loader, Sound ve NetStream gibi sınıfları kullanarak, Flex kullanımı sırasında MXML etiketleri veya AIR uygulamasındaki HTML etiketleri üzerinden yükleyin.
- Verileri ayıklama: Bitmap nesnelerini, `BitmapData.draw()` ve `BitmapData.drawWithQuality()` yöntemini, `Sound.id3` özelliğini ya da `SoundMixer.computeSpectrum()` yöntemini kullanarak yüklenen medya içeriğinden verileri ayıklayabilirsiniz. `drawWithQuality` yöntemi Flash Player 11.3 ve üstü ile AIR 3.3 ve üstünde kullanılabilir.
- Verilere erişme - URLStream, URLLoader, FileReference, Socket ve XMLSocket gibi sınıfları kullanarak verileri doğrudan harici dosyadan (örn. bir XML dosyası) yükleyerek o verilere doğrudan erişebilirsiniz. AIR uygulaması, veri yüklemek için FileStream ve XMLHttpRequest gibi ek sınıflar sunar.

Flash Player güvenlik modeli, içerik yüklenmesi ve verilere erişilmesi için farklı kurallar tanımlar. Genellikle, içerik yüklenmesinde, verilere erişilmesine göre daha az kısıtlama vardır.

Genel olarak içerik (SWF dosyaları, bitmap'ler, mp3 dosyaları ile videolar) herhangi bir yerden yüklenebilir ancak içerik, yüklemeyi yapan kodun veya içeriğin bulunduğu alandan başka bir alandaysa, söz konusu içerik ayrı bir güvenlik sanal alanından bölümlenir.

İçerik yüklenmesine yönelik birkaç engel vardır:

- Varsayılan olarak, yerel SWF dosyaları (kullanıcının sabit sürücüsü gibi ağ dışındaki bir adresten indirilen SWF dosyaları), dosya sistemiyle yerel sanal alanda sınıflandırılır. Bu dosyalar, ağdan içerik yükleyemez. Daha fazla bilgi için bkz. "Yerel sanal alanlar" sayfa 1055.
- Real-Time Messaging Protocol (RTMP) sunucuları içeriğe erişimi sınırlar. Daha fazla bilgi için, bkz. "RTMP sunucuları kullanılarak teslim edilen içerik" sayfa 1074.

Yüklenen ortam bir görüntü, ses veya video ise, ortamın piksel verileri ve ses verileri gibi verilerine kendi güvenlik sanal alanı dışındaki bir SWF dosyası tarafından erişilebilmesi için, o SWF dosyasının etki alanının, ortamın kaynak etki alanındaki URL ilke dosyasına dahil edilmiş olması gerekir. Ayrıntılar için bkz. "Veri olarak yüklenen ortama erişme" sayfa 1077.

Diğer yüklenmiş veri şekilleri arasında, URLLoader nesnesiyle yüklenen metin veya XML dosyaları yer alır. Yine bu durumda da, başka bir güvenlik sanal alanındaki verilere erişmek için, kaynak etki alanındaki URL ilke dosyası aracılığıyla izin verilmesi gerekir. Ayrıntılar için, bkz. "URLLoader ve URLStream öğelerini kullanma" sayfa 1080.

Not: *Uzaktaki içeriği ya da verileri yüklemek için AIR uygulamasının sanal alanında kod yürütümü için hiçbir zaman ilke dosyalarına gerek yoktur.*

Güvenlik sanal alanları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İstemci bilgisayarlar birkaç kaynaktan; örneğin harici web sitelerden, yerel dosya sisteminden veya kurulmuş bir AIR uygulamasından gelen kod, içerik ile verilerin yer aldığı bağımsız dosyaları elde edebilirler. Flash Player ve AIR çalışma süreleri kod dosyalarını ve diğer kaynakları, paylaşılan nesneler, bitmap'ler, sesler, videolar ile veri dosyaları gibi diğer kaynakları, Flash Player'a yüklendikleri zamanki kaynaklarını esas alarak tek tek güvenlik sanal alanlarını belirler. İlerleyen bölümlerde belirli bir sanal alanda hangi kodun veya içeriğin yürütüleceğini belirleyen, çalışma zamanları tarafından yürütülen kurallar ele alınmıştır.

Flash Player güvenliği hakkında daha fazla bilgi için, bkz. "Güvenlik" başlıklı Flash Player Developer Center konusu, www.adobe.com/go/devnet_security_tr.

Uzak sanal alanlar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

FlashPlayer ve AIR çalışma zamanı internetten gelen varlıkları (SWF dosyaları dahil) orijin alanlarına denk gelen ayrı sanal alanlarda sınıflandırır. Örneğin, *example.com* öğesinden yüklenen varlıklar, *foo.org* öğesine yüklenen varlıklarınkinden farklı bir güvenlik sanal alanına yerleştirilecektir. Varsayılan olarak, bu dosyaların kendi sunucularındaki herhangi bir kaynağa erişme yetkisi vardır. Uzak SWF dosyalarının, URL ilke dosyaları ve `Security.allowDomain()` yöntemi gibi açık web sitesi ve yazar izinleriyle diğer etki alanlarındaki ek verilere erişmesine izin verilebilir. Ayrıntılar için bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062 ve "Yazar (geliştirici) denetimleri" sayfa 1066.

Uzak SWF dosyaları, yerel dosyaları ve kaynakları yükleyemez.

Flash Player güvenliği hakkında daha fazla bilgi için www.adobe.com/go/devnet_security_tr adresindeki "Güvenlik" başlıklı Flash Player Geliştirici Merkezi konusuna bakın.

Yerel sanal alanlar

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yerel dosya, `file:` protokolü veya Universal Naming Convention (UNC) yolu kullanılarak başvurulan herhangi bir dosyayı açıklar. Yerel SWF dosyaları, dört yerel sanal alandan birine yerleştirilir:

- local-with-filesystem sanal alanı - Güvenlik amaçları açısından Flash Player ve AIR uygulamalarının çalışma zamanları tüm yerel dosyaları varsayılan olarak local-with-filesystem sanal alanına yerleştirir. Yürütülebilir kod bu sanal alandan yerel dosyaları (örneğin, URLLoader sınıfını kullanarak) okuyabilir ama ağ ile herhangi bir biçimde iletişim kuramaz. Böylece yerel verilerin ağdan sızıp uygun olmayan şekilde paylaşılamayacağı konusunda kullanıcıya güvence sağlanır.

- Ağ iletişimiyle yerel sanal alan—Bir SWF dosyasını derlerken, bu dosyanın yerel dosya olarak çalıştırıldığında ağ erişimine sahip olduğunu belirtebilirsiniz (bkz. "Yerel SWF dosyalarının sanal alan türünü ayarlama" sayfa 1058). Bu dosyalar, ağ iletişimiyle yerel sanal alana yerleştirilir. Ağ iletişimiyle yerel sanal alana atanmış SWF dosyaları, yerel dosya erişimini kaybeder. Bunu sonucunda SWF dosyalarının ağdaki verilere erişmesine izin verilir. Ancak, ağ iletişimiyle yerel bir SWF dosyasının bir URL ilke dosyası veya `Security.allowDomain()` yöntemine yapılan bir çağrı yoluyla söz konusu eylem için izni olmadığı sürece ağ kaynaklı verileri okumasına izin verilmez. Bu iznin verilmesi için, bir URL ilke dosyasının `<allow-access-from domain="*"/>` veya `Security.allowDomain("*")` öğesini kullanarak *tüm* etki alanlarına izin vermesi gerekir. Daha fazla bilgi için bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062 ve "Yazar (geliştirici) denetimleri" sayfa 1066.
- Yerel güvenilen sanal alan—Güvenilen olarak kaydedilen (kullanıcılar veya yükleyici programlar tarafından) yerel SWF dosyaları, yerel güvenilen sanal alana yerleştirilir. Sistem yöneticilerinin ve kullanıcılarının da güvenlikle ilgili dikkat edilmesi gereken noktaları esas alarak yerel güvenilen sanal alana yerel bir SWF dosyasını yeniden atayabilme veya bu sanal alandan SWF dosyasını taşıyabilme yeteneğine sahip olması gerekir (bkz. "Yönetimsel denetimler" sayfa 1060 ve "Kullanıcı denetimleri" sayfa 1061). Yerel güvenilen sanal alana atanan SWF dosyaları, diğer tüm SWF dosyalarıyla etkileşim kurabilir ve herhangi bir yerden (uzak veya yerel) veri yükleyebilir.
- AIR uygulaması sanal alanı—Bu sanal alan, çalışan AIR uygulamasıyla yüklenmiş içerikleri barındırır. Varsayılan olarak, AIR uygulamasının sanal alanında kod yürütülmesi, herhangi bir alandaki kodun çapraz dosyasını oluşturabilir. Ancak, AIR uygulamasının sanal alanının dışındaki dosyalara, uygulamanın sanal alanındaki kodun çapraz komut dosyasını oluşturmasına izin verilmez. Varsayılan olarak, AIR uygulamasının sanal alanındaki kod ve içerik, herhangi bir etki alanından içerik ile verileri yükleyebilir.

Ağ iletişimiyle yerel ve dosya sistemiyle yerel sanal alanlar arasında iletişim kurulması ve dosya sistemiyle yerel ve uzak sanal alanlar arasında iletişim kurulması kesinlikle yasaktır. Bu tür bir iletişim izni, Flash Player'da çalışan bir uygulama veya kullanıcı ya da yönetici tarafından verilemez.

Yerel HTML dosyaları ile yerel SWF dosyaları arasında herhangi bir yönde komut dosyası oluşturulması (örneğin, ExternalInterface sınıfı kullanılarak) için, hem HTML dosyasının hem de SWF dosyasının yerel güvenilen sanal alanda olması gerekir. Bunun nedeni, tarayıcılar için yerel güvenlik modellerinin Flash Player yerel güvenlik modelinden farklı olmasıdır.

Ağ iletişimiyle yerel sanal alanda bulunan SWF dosyaları, dosya sistemiyle yerel sanal alanda bulunan SWF dosyalarını yükleyemez. Dosya sistemiyle yerel sanal alanda bulunan SWF dosyaları, ağ iletişimiyle yerel sanal alanda bulunan SWF dosyalarını yükleyemez.

AIR uygulamasının sanal alanı

Adobe AIR 1.0 ve üstü

Adobe AIR çalışma zamanı, Flash Player güvenlik sanal alanına *uygulama* sanal alanı adında ilave bir sanal alan ekler. Uygulamanın sanal alanına AIR uygulamasının yükünün bir bölümü olarak kurulan dosyalar. Uygulama tarafından yüklenen diğer bütün dosyaların, Flash Player güvenlik modeli tarafından belirtilmiş olanlara benzeyen güvenlik sınırlılıkları vardır.

Uygulama yüklendiğinde, AIR paketinde bulunan tüm dosyalar kullanıcının bilgisayarındaki bir uygulama dizinine yüklenir. Geliştiriciler `app:/` URL şeması (bkz. "URI şemaları" sayfa 822) aracılığıyla kodda bu dizine başvuru yapabilir. Uygulama dizin ağacındaki tüm dosyalar, uygulama çalıştığında uygulama sanal alanına atanır. Uygulama sanal alanındaki içerik, yerel dosya sistemiyle etkileşim dahil, bir AIR uygulamasının kullanabileceği tüm ayrıcalıklarla donatılmıştır.

Çoğu AIR uygulaması, uygulamayı çalıştırmak için yalnızca yerel olarak yüklenen bu dosyaları kullanır. Ancak, AIR uygulamaları yalnızca uygulama dizinindeki dosyalarla kısıtlanmamıştır - tüm kaynaklardan tüm dosya türlerini yükleyebilir. Bu, kullanıcı bilgisayarındaki yerel dosyaların yanı sıra, yerel ağda ve İnternet'te bulunan kullanılabilir harici kaynaklardaki dosyaları da içerir. Dosya türünün güvenlik kısıtlamaları üzerinde etkisi yoktur; yüklenen HTML dosyaları, aynı kaynaktan yüklenen SWF dosyalarıyla aynı güvenlik ayrıcalıklarına sahiptir.

Uygulama güvenlik sanal alanındaki içerik, diğer sanal alanlardaki içeriğin kullanımının engellendiği AIR API'lerine erişebilir. Örneğin, uygulamaya ilişkin uygulama tanımlayıcı dosyasının içeriğini döndüren `air.NativeApplication.nativeApplication.applicationDescriptor` özelliği, uygulama güvenlik sanal alanındaki içerikle kısıtlanmıştır. Başka bir kısıtlanmış API örneği, yerel dosya sistemine okuma ve yazma için yöntemler içeren FileStream sınıfıdır.

Yalnızca uygulamanın güvenlik sanal alanındaki içerik tarafından kullanılabilen ActionScript API'leri, *Adobe Flash Platformu için ActionScript 3.0 Başvurusu* içinde AIR logosuyla gösterilir. Bu API'lerin diğer sanal alanlarda kullanılması, çalışma zamanının SecurityError istisnası atmasına neden olur.

HTML içeriği (HTMLLoader nesnesindeki), tüm AIR JavaScript API'leri (AIRAliases.js dosyasının kullanımı sırasında `window.runtime` özelliği veya `air` nesnesi yoluyla kullanılabilen) uygulama güvenlik sanal alanındaki içerik tarafından kullanılabilir. Başka bir sanal alandaki HTML içeriği `window.runtime` özelliğine erişimi yoktur. Bu nedenle bu içerik AIR veya Flash Player API'lerine erişemez.

AIR uygulamasının sanal alanında içerik yürütmenin ek sınırlılıkları:

- Uygulama güvenlik sanal alanındaki HTML içeriği için, kod yüklendikten sonra dizeleri dinamik olarak yürütülebilir koda dönüştürebilen API'lerin kullanımına ilişkin sınırlamalar mevcuttur. Bunun amacı, uygulamanın yanlışlıkla uygulama dışı kaynaklardan (güvenli olmayabilecek ağ etki alanları gibi) kod yerleştirmesini (ve yürütmesini) engellemektir. Bir örnek, `eval()` işlevi kullanımıdır. Ayrıntılar için bkz. “Farklı sanal alanlardaki içerik için kod kısıtlamaları” sayfa 1094.
- Olası kimlik avı saldırılarını önlemek için, uygulama güvenlik sanal alanında yer alan SWF içeriğindeki ActionScript TextField nesnelerinde bulunan HTML içeriğinde `img` etiketleri yoksayılır.
- Uygulama sanal alanındaki içerik, ActionScript 2.0 metin alanlarındaki HTML içeriğinde `asfunction` protokolünü kullanamaz.
- Uygulama sanal alanındaki SWF içeriği, Flash Player 9 Update 3'e eklenmiş bir özellik olan etki alanları arası önbelleği kullanamaz. Bu özellik, Flash Player'ın Adobe platform bileşen içeriğini kalıcı olarak önbelleğe almasını ve bu içeriği talep üzerine yüklenen SWF içeriğinde yeniden kullanmasını sağlar. (İçeriğin birçok kez yüklenme ihtiyacını ortadan kaldırır.)

AIR uygulamasının içindeki JavaScript'in sınırlılıkları

Adobe AIR 1.0 ve üstü

Uygulama güvenlik sanal alanındaki içerikten farklı olarak uygulama dışı bir güvenlik sanal alanında yer alan JavaScript içeriği, dinamik olarak oluşturulan kodu yürütmek için her zaman `eval()` işlevini *çağırabilir*. Diğer taraftan, AIR uygulamasında uygulama dışı güvenlik sanal alanında JavaScript yürütmekle ilgili sınırlılıklar vardır. Bunlar şunları içerir:

- Uygulama dışı sanal alandaki JavaScript kodunun `window.runtime` nesnesine erişimi yoktur, ve aynı şekilde bu kod AIR API'lerini yürütemez.

- Varsayılan olarak, uygulama dışı güvenlik sanal alanındaki içerik, isteği çağıran etki alanı dışındaki etki alanlarından veri yüklemek için XMLHttpRequest çağrılarını kullanamaz. Ancak, uygulama kodu mevcut karede veya iframe'de `allowCrossdomainXHR` niteliğini ayarlayarak uygulama dışı içeriğe bunu yapması için izin verebilir. Daha fazla bilgi için bkz. "Farklı sanal alanlardaki içerik için kod kısıtlamaları" sayfa 1094.
- JavaScript `window.open()` yönteminin çağrılmasına ilişkin kısıtlamalar mevcuttur. Ayrıntılar için bkz. "JavaScript window.open() yönteminin çağrılmasına ilişkin kısıtlamalar" sayfa 1097.
- Uzak (ağ) güvenlik sanal alanlarındaki HTML içeriği uzak etki alanlarından (ağ URL'lerinden) yalnızca `frame`, `iframe` ve `img` içeriği yükleyebilir.
- Dosya sistemiyle yerel, ağ iletişimiyle yerel veya yerel-güvenilir sanal alanlardaki HTML içeriği yerel sanal alanlardan (uygulamadan veya ağ URL'lerinden değil) yalnızca `frame`, `iframe` ve `img` içeriğini okuyabilir.

Ayrıntılar için bkz. "Farklı sanal alanlardaki içerik için kod kısıtlamaları" sayfa 1094.

Yerel SWF dosyalarının sanal alan türünü ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir bilgisayarın son kullanıcısı veya yöneticisi, yerel bir SWF dosyasına güven onayı verildiğini belirterek bu dosyanın hem yerel hem de ağdan, başka bir deyişle, tüm etki alanlarından veri yüklemesine izin verir. Bu özellik, Genel Flash Player Güven Onayı ve Kullanıcı Flash Player Güven Onayı dizinlerinde belirtilmiştir. Daha fazla bilgi için bkz. "Yönetimsel denetimler" sayfa 1060 ve "Kullanıcı denetimleri" sayfa 1061.

Yerel sanal alanlar hakkında daha fazla bilgi almak için, bkz. "Yerel sanal alanlar" sayfa 1055.

Adobe Flash Professional

Geliştirme aracında SWF belgesinin yayınlama ayarlarını yaparak bir SWF dosyasını dosya sistemiyle yerel sanal alan için veya ağ iletişimiyle yerel sanal alan için konfigüre edebilirsiniz.

Adobe Flex

Adobe Flex derleyicisinde `use-network` bayrağını ayarlayarak bir SWF dosyasını dosya sistemiyle yerel sanal alan için veya ağ iletişimiyle yerel sanal alan için konfigüre edebilirsiniz. Daha fazla bilgi için, bkz. "Uygulama derleyicisi seçenekleri hakkında", *Adobe Flex 3 Uygulamaları Oluşturma ve Konuşlandırma*.

Security.sandboxType özelliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir SWF dosyasının yazarı, Flash Player veya AIR uygulamasının SWF dosyasını hangi tür sanal alana atadığını belirlemek için salt okunur statik `Security.sandboxType` özelliğini kullanabilir. Security sınıfı, aşağıdaki gibi, `Security.sandboxType` özelliğinin olası değerlerini temsil eden sabitleri içerir:

- `Security.REMOTE`—SWF dosyası bir Internet URL'sinden gelmiş olup etki alanı tabanlı sanal alan kuralları kapsamında çalışır.
- `Security.LOCAL_WITH_FILE`—SWF dosyası yerel bir dosyadır ancak herhangi bir kullanıcı bu dosyaya güven onayı vermemiştir ve dosya ağ iletişimi atamasıyla yayınlanmamıştır. SWF dosyası yerel veri kaynaklarından okuyabilir ancak Internet üzerinden iletişim kuramaz.

- `Security.LOCAL_WITH_NETWORK`—SWF dosyası yerel bir dosyadır ve herhangi bir kullanıcı bu dosyaya güven onayı vermemiş ancak dosya ağ iletişimi atamasıyla yayınlanmıştır. SWF dosyası Internet üzerinden iletişim kurabilir, ancak yerel veri kaynaklarından okuyamaz.
- `Security.LOCAL_TRUSTED`—SWF dosyası yerel bir dosyadır ve kullanıcı bu dosyaya Ayarlar Yöneticisi veya Flash Player güven onayı konfigürasyon dosyasını kullanarak güven onayı vermiştir. SWF dosyası yerel veri kaynaklarından okuyabilir ve Internet üzerinden iletişim kurabilir.
- `Security.APPLICATION`—SWF dosyası bir AIR uygulamasında çalışmakta olup o uygulamanın paketiyle (AIR dosyası) yüklenmiştir. Varsayılan olarak, AIR uygulaması sanal alanında bulunan dosyalar herhangi bir etki alanındaki dosyanın çapraz komut dosyasını oluşturabilir. Ancak, AIR uygulaması sanal alanı dışındaki dosyaların, AIR dosyasının çapraz komut dosyasını oluşturmasına izin verilmez. Varsayılan olarak, AIR uygulaması sanal alanındaki dosyalar herhangi bir etki alanından içerik ve verileri yükleyebilir.

İzin denetimleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player istemcisi çalışma zamanı güvenlik modeli, SWF dosyaları, yerel veriler ve Internet URL'leri gibi nesneler olan kaynaklar etrafında tasarlanmıştır. *Paydaşlar*, bu kaynaklara sahip olan veya bu kaynakları kullanan taraflardır. Paydaşlar kendi kaynakları üzerinde denetimler (güvenlik ayarları) uygulayabilir ve her kaynak dört paydaşa sahiptir. Flash Player, aşağıdaki resimde gösterildiği gibi, bu denetimler için kesinlikle bir yetki hiyerarşisini zorlar:

Güvenlik denetimlerinin hiyerarşisi

Bu, örneğin bir yönetici bir kaynağa erişimi kısıtladığında diğer paydaşların bu kısıtlamayı geçersiz kılamayacağı anlamına gelir.

AIR uygulamalarında bu izin kontrolleri yalnızca AIR uygulamasının sanal alanının dışında çalıştırılan içeriğe uygulanır.

Yönetimsel denetimler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir bilgisayarın yönetici kullanıcısı (yönetimsel haklarla oturum açan kişi), bilgisayarın tüm kullanıcılarını etkileyen Flash Player güvenlik ayarlarını uygulayabilir. Ev bilgisayarı gibi, kurumsal olmayan bir ortamda genellikle yönetimsel haklara sahip tek bir kişi vardır. Kurumsal ortamda da tek tek kullanıcılar bilgisayarın yönetimsel haklarına sahip olabilir.

İki tür yönetimsel kullanıcı denetimi vardır:

- mms.cfg dosyası
- Genel Flash Player Güven Onayı dizini

mms.cfg dosyası

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

mms.cfg dosyası, yöneticilerin çeşitli yetenekleri etkinleştirmesine veya bu yeteneklere erişimi kısıtlamasına olanak sağlayan bir metin dosyasıdır. Flash Player başlatıldığında, bu dosyadan güvenlik ayarlarını okur ve işlevselliği sınırlamak için bu ayarları kullanır. mms.cfg dosyası, yöneticinin gizlilik denetimleri, yerel dosya güvenliği, soket bağlantıları, vb. gibi yetenekleri yönetmek için kullandığı ayarları içerir.

SWF dosyası, `Capabilities.avHardwareDisable` ve `Capabilities.localFileReadDisable` özelliklerini çağırarak devre dışı bırakılmış yeteneklerle ilgili bazı bilgilere erişebilir. Ancak, mms.cfg dosyasındaki ayarların çoğu, ActionScript'ten sorgulanamaz.

Bir bilgisayar için uygulamadan bağımsız güvenlik ve gizlilik ilkelerini zorlamak için, mms.cfg dosyasının yalnızca sistem yöneticileri tarafından değiştirilmesi gerekir. mms.cfg dosyası, uygulama yükleyicileri tarafından kullanılmak üzere tasarlanmamıştır. Yönetici ayrıcalıklarıyla çalışan bir yükleyici, mms.cfg dosyasının içeriklerini değiştirebilse de Adobe, bu tür bir kullanımın kullanıcı güvenini ihlal ettiğini düşünmekte ve yükleyici oluşturanları asla mms.cfg dosyasını değiştirmemeye teşvik etmektedir.

mms.cfg dosyası şu konumda saklanır:

- Windows: *system*\Macromed\Flash\mms.cfg

  (örneğin, C:\WINDOWS\system32\Macromed\Flash\mms.cfg)

- Mac: *app support*/Macromedia/mms.cfg

  örneğin, /Library/Application Support/Macromedia/mms.cfg)

mms.cfg dosyası hakkında daha fazla bilgi almak için, bkz. Flash Player Yönetim Kılavuzu, www.adobe.com/go/flash_player_admin_tr.

Genel Flash Player Güven Onayı dizini

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yönetici kullanıcılar ve yükleyici uygulamalar belirtilen yerel SWF dosyalarını tüm kullanıcılar için güvenilen olarak kaydedebilir. Bu SWF dosyaları yerel güvenilen sanal alana atanır. Diğer SWF dosyalarıyla etkileşim kurabilir ve uzak ya da yerel olmak üzere herhangi bir yerden veri yükleyebilir. Dosyalar, şu konumda, Genel Flash Player Güven Onayı dizininde güvenilen olarak atanır:

- Windows: *system*\Macromed\Flash\FlashPlayerTrust

  (örneğin, C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac: *app support*/Macromedia/FlashPlayerTrust

  (örneğin, /Library/Application Support/Macromedia/FlashPlayerTrust)

Flash Player Güven Onayı dizini, her biri satırlarda birer yol bulunacak şekilde güvenilen yolları listeleyen herhangi bir sayıda metin dosyası içerebilir. Yolların her biri tek bir SWF dosyası, HTML dosyası veya dizin olabilir. Yorum satırları # sembolüyle başlar. Örneğin, şu metni içeren bir Flash Player güven onayı konfigürasyon dosyası, belirtilen dizindeki ve tüm alt dizinlerdeki dosyaların hepsine güven onayı verir:

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

Güven onayı konfigürasyon dosyasında listelenen yolların her zaman yerel yollar veya SMB ağ yolları olması gerekir. Güven onayı konfigürasyon dosyasındaki tüm HTTP yolları yoksayılır; yalnızca yerel dosyalara güven onayı verilebilir.

Çakışmaları önlemek için, her güven onayı konfigürasyon dosyasına, yükleyici uygulamaya karşılık gelen bir dosya adı verin ve .cfg dosya uzantısını kullanın.

Yükleyici uygulama üzerinden yerel olarak çalışan bir SWF dosyası dağıtan geliştirici olarak siz de yükleyici uygulamanın, Genel Flash Player Güven Onayı dizinine bir konfigürasyon dosyası eklemesini sağlayarak, dağıttığınız dosyaya tam ayrıcalık verebilirsiniz. Yükleyici uygulamanın yönetimsel haklara sahip bir kullanıcı tarafından çalıştırılması gerekir. mms.cfg dosyasının aksine Genel Flash Player Güven Onayı dizini, güven onayı izinleri veren yükleyici uygulamalar için dahil edilmiştir. Hem yönetici kullanıcılar hem de yükleyici uygulamalar, Genel Flash Player Güven Onayı dizinini kullanarak güvenilen yerel uygulamalar atayabilir.

Ayrıca tek tek kullanıcılar için Flash Player Güven Onayı dizinleri de vardır (bkz. "Kullanıcı denetimleri" sayfa 1061).

Kullanıcı denetimleri

Flash Player 9 ve üstü

Flash Player, izinlerin ayarlanmasına yönelik üç farklı kullanıcı düzeyinde mekanizma sağlar: Ayarlar UI'si ve Ayarlar Yöneticisi ile Kullanıcı Flash Player Güven Onayı dizini.

Ayarlar UI'si ve Ayarlar Yöneticisi

Flash Player 9 ve üstü

Ayarlar UI'si, belirli bir etki alanının ayarlarının konfigüre edilmesine yönelik hızlı ve etkileşimli bir mekanizmadır. Ayarlar Yöneticisi, daha ayrıntılı bir arabirim sunar ve etki alanlarının çoğunun veya tümünün izinlerini etkileyen global değişiklikler yapılması yeteneğini sağlar. Ayrıca, SWF dosyası tarafından, güvenlik veya gizlilikle ilgili çalışma zamanı kararları gerektiren yeni bir izin istendiğinde, kullanıcıların bazı Flash Player ayarlarını yapabildiği iletişim kutuları görüntülenir.

Ayarlar Yöneticisi ve Ayarlar Kullanıcı Arabirimi, kamera ve mikrofon ayarları, paylaşılan nesne depolama alanı ayarları, eski içerikle ilgili ayarlar vb. gibi güvenlikle ilgili seçenekler sağlar. AIR uygulamalarında Ayarlar Yöneticisi de Ayarlar kullanıcı arabirimi (UI) de kullanılmaz.

Not: *mms.cfg dosyasında yapılan ayarlar (bkz. "Yönetimsel denetimler" sayfa 1060), Ayarlar Yöneticisi'nde yansıtılmaz.*

Ayarlar Yöneticisi ile ilgili ayrıntılar için, bkz. www.adobe.com/go/settingsmanager_tr.

Kullanıcı Flash Player Güven Onayı dizini

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kullanıcılar ve yükleyici uygulamalar belirtilen yerel SWF dosyalarını güvenilen olarak kaydedebilir. Bu SWF dosyaları yerel güvenilen sanal alana atanır. Diğer SWF dosyalarıyla etkileşim kurabilir ve uzak ya da yerel olmak üzere herhangi bir yerden veri yükleyebilir. Kullanıcı, aşağıdaki konumlarda (geçerli kullanıcıya özgü konumlar) paylaşılan nesne depolama alanıyla aynı dizinde bulunan Kullanıcı Flash Player Güven Onayı dizininde bir dosyayı güvenilen olarak atar:

- Windows: app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (örneğin, Windows XP'de C:\Documents and Settings\CanD\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust veya Windows Vista'da C:\Users\CanD\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust)

  Windows'ta, Application Data klasörü varsayılan olarak gizlenmiştir. Gizli klasörleri ve dosyaları göstermek için Bilgisayarım'ı seçip Windows Gezgini'ni açın ve Araçlar > Klasör Seçenekleri ve Görünüm sekmesini seçin. Görünüm sekmesinin altında, Gizli dosya ve klasörleri göster radyo düğmesini seçin.

- Mac: app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (örneğin, /Users/CanD/Library/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  Bu ayarlar bilgisayarda oturum açan diğer kullanıcıları değil, yalnızca geçerli kullanıcıyı etkiler. Yönetimsel hakları olmayan bir kullanıcı sistemde kendi bölümüne bir uygulama yüklerse, Kullanıcı Flash Player Güven Onayı dizini, yükleyicinin uygulamayı o kullanıcı için güvenilen olarak kaydetmesine olanak sağlar.

  Yükleyici uygulama yoluyla yerel olarak çalışan bir SWF dosyası dağıtan geliştirici olarak siz de yükleyici uygulamanın, Kullanıcı Flash Player Güven Onayı dizinine bir konfigürasyon dosyası eklemesini sağlayarak, dağıttığınız dosyaya tam ayrıcalık verebilirsiniz. Bu durumda bile bir kullanıcı eylemi (yükleme) tarafından başlatıldığı için Kullanıcı Flash Player Güven Onayı dizini dosyası bir kullanıcı denetimi olarak değerlendirilir.

  Yönetici kullanıcı veya yükleyiciler tarafından bir bilgisayarın tüm kullanıcıları için bir uygulamayı kaydetmek üzere kullanılan bir de Genel Flash Player Güven Onayı dizini vardır (bkz. “Yönetimsel denetimler” sayfa 1060).

Web sitesi denetimleri (ilke dosyaları)

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Web sunucunuzdaki verileri, diğer etki alanlarındaki SWF dosyaları için kullanılabilir duruma getirmek üzere sunucunuzda bir ilke dosyası oluşturabilirsiniz. *İlke dosyası*, sunucunuzda belirli bir konuma yerleştirilen bir XML dosyasıdır.

İlke dosyaları, aşağıda örnekleri verilen çok sayıda varlığa erişimi etkiler:

- Bitmap, ses ve videolardaki veriler
- XML ve metin dosyalarının yüklenmesi
- Başka güvenlik etki alanlarındaki SWF dosyalarının yükleyen SWF dosyasının güvenlik etki alanına içe aktarılması
- Soket ve XML soketi bağlantılarına erişim

ActionScript nesneleri iki farklı türde sunucu bağlantısı başlatır: belge tabanlı sunucu bağlantıları ve soket bağlantıları. Loader, Sound, URLLoader ve URLStream gibi ActionScript nesneleri, belge tabanlı sunucu bağlantılarını başlatır ve bu nesneler bir URL'den dosya yükler. ActionScript Socket ve XMLSocket nesneleri, yüklenen belgelerle değil akış verileriyle çalışan soket bağlantılarını oluşturur.

Flash Player iki tür sunucu bağlantısını desteklediğinden, iki tür ilke dosyası vardır—URL ilke dosyaları ve soket ilke dosyaları.

- Belge tabanlı bağlantılar için *URL ilke dosyaları* gerekir. Bu dosyalar, sunucunun verilerinin ve belgelerinin belirli etki alanlarından veya tüm etki alanlarından sunulan SWF dosyaları için kullanılabilir olduğunu belirtmesine olanak sağlar.
- Soket bağlantıları için *soket ilke dosyası* gerekir ve bu, Socket ve XMLSocket sınıflarını kullanarak doğrudan düşük TCP soket düzeyinde ağ iletişimini etkinleştirir.

Flash Player, girişimde bulunulan bağlantının kullanmak istediği aynı protokol kullanılarak ilke dosyalarının iletilmesini gerektirir. Örneğin, HTTP sunucunuza bir ilke dosyası yerleştirdiğinizde, diğer etki alanlarındaki SWF dosyalarının, HTTP sunucusu olarak buradan veri yüklemesine izin verilir. Ancak, aynı sunucuda bir soket ilkesi dosyası sağlamazsanız, diğer etki alanlarındaki SWF dosyalarının soket düzeyinde sunucuya bağlanmasını yasaklarsınız. Başka bir deyişle, bir ilke dosyasının alınma yolunun, bağlanma yoluyla aynı olması gerekir.

İlke dosyası kullanımı ve sözdizimi, Flash Player 10 için yayınlanan SWF dosyaları için geçerli olduğundan, bu bölümün devamında kısaca ele alınmıştır. (Art arda yayınlanan sürümlerde Flash Player güvenliği güçlendirilmiş olduğundan, ilke dosyası uygulaması, önceki Flash Player sürümlerinde biraz daha farklıdır.) İlke dosyaları hakkında daha ayrıntılı bilgi almak için, bkz. “Flash Player 9'da İlke Dosyası Değişiklikleri” başlıklı Flash Player Developer Center konusu, www.adobe.com/go/devnet_security_tr.

AIR uygulamasının sanal alanında kod yürütülmesi URL veya soketten veri almak için ilke dosyası gerekmez. Bir AIR uygulamasında uygulama dışı sanal alanda yürütülen kod için ilke dosyası gerekmez.

Ana ilke dosyaları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Varsayılan olarak, Flash Player (ve AIR uygulama sanal alanında bulunmayan AIR içeriği) ilk olarak sunucunun kök dizininde `crossdomain.xml` adındaki bir URL ilke dosyasını ve bağlantı noktası 843'te bir soket ilke dosyasını arar. Bu konumlardan herhangi birinde bulunan dosya, *ana ilke dosyası* olarak adlandırılır. (Soket bağlantısı olması durumunda Flash Player, ana bağlantıyla aynı bağlantı noktasında bir soket ilke dosyasını da arar. Ancak bu bağlantı noktasında bulunan bir ilke dosyası ana ilke dosyası olarak değerlendirilmez.)

Ana ilke dosyası, erişim izinlerini belirtmenin yanı sıra, bir *meta-ilke* deyimi de içerebilir. Meta-ilke, hangi konumların ilke dosyalarını içerebildiğini belirtir. URL ilke dosyaları için varsayılan meta-ilke “yalnızca ana” olup bu, /crossdomain.xml dosyasının sunucuda izin verilen tek ilke dosyası olduğu anlamına gelir. Soket ilke dosyaları için varsayılan meta-ilke "tümü" olup bu, ana bilgisayardaki herhangi bir soketin bir soket ilke dosyası sunabileceği anlamına gelir.

Not: *Flash Player 9 ve öncesinde, URL ilke dosyaları için varsayılan meta-ilke "tümü" olup bu, herhangi bir dizinin bir ilke dosyası içerebileceği anlamına gelir. Varsayılan /crossdomain.xml dosyası dışındaki konumlardan ilke dosyaları yükleyen uygulamalar konuşlandırdıysanız ve bu uygulamalar şimdi Flash Player 10'da çalışabilirse, sizin (veya sunucu yöneticisinin) ek ilke dosyalarına izin vermek için ana ilke dosyasını değiştirdiğinizden emin olun. Farklı bir meta-ilkenin nasıl belirtildiği hakkında bilgi almak için, bkz. “Flash Player 9'da İlke Dosyası Değişiklikleri” başlıklı Flash Player Developer Center konusu, www.adobe.com/go/devnet_security_tr.*

SWF dosyası, `Security.loadPolicyFile()` yöntemini çağırarak farklı bir ilke dosya adını veya farklı bir dizin konumunu kontrol edebilir. Ancak ana ilke dosyası, hedef konumun ilke dosyaları sunabileceğini belirtmezse, söz konusu konumda bir ilke dosyası olsa da `loadPolicyFile()` öğesine yapılan çağrı herhangi bir etki yaratmaz. İlke dosyası gerektiren herhangi bir ağ işlemini denemeden önce `loadPolicyFile()` öğesini çağırın. Flash Player, ağ iletişimi isteklerini, karşılık gelen ilke dosyası girişimlerinin arkasında otomatik olarak kuyruğa alır. Bu nedenle, örneğin, bir ağ iletişimi işlemi başlatılmadan hemen önce `Security.loadPolicyFile()` öğesinin çağrılması kabul edilebilir.

Flash Player, ana ilke dosyasını kontrol ederken, sunucu yanıtı için üç saniye bekler. Herhangi bir yanıt alınmazsa, Flash Player bir ana ilke dosyasının bulunmadığını varsayar. Ancak, `loadPolicyFile()` öğesine yapılan çağrılar için varsayılan bir zaman aşımı değeri yoktur; Flash Player, çağrılmakta olan dosyanın varolduğunu varsayar ve bunu yüklemek için gereken süre boyunca bekler. Bu nedenle, bir ana ilke dosyasının yüklendiğinden emin olmak için, `loadPolicyFile()` öğesini kullanarak ana ilke dosyasını açıkça çağırın.

Yöntem `Security.loadPolicyFile()` olarak adlandırılsa da, ilke dosyası gerektiren bir ağ çağrısı yayınlanmadan önce ilke dosyası yüklenmez. `loadPolicyFile()` öğesine yapılan çağrılar, Flash Player uygulamasına gerektiğinde ilke dosyalarını nerede arayacağını bildirir.

Bir ilke dosyası isteği başlatıldığında veya tamamlandığında bildirim alamazsınız ve bunun için bir neden de yoktur. Flash Player, ilke denetimlerini eşzamansız olarak gerçekleştirir ve ilke dosyası kontrolleri başarıyla tamamlandıktan sonra otomatik olarak bağlantıların başlatılmasını bekler.

İlerleyen bölümlerde, yalnızca URL ilke dosyaları için geçerli olan bilgiler yer almaktadır. Soket ilke dosyaları hakkında daha fazla bilgi almak için, bkz. "Soketlere bağlanma" sayfa 1080.

URL ilke dosyası kapsamı

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

URL ilke dosyası yalnızca yüklenmiş olduğu dizin ve bu dizinin alt dizinleri için geçerli olur. Kök dizindeki bir ilke dosyası sunucunun tamamı için geçerli olur; rastgele bir alt dizinden yüklenen bir ilke dosyası ise yalnızca o dizin ve o dizinin alt dizinleri için geçerli olur.

İlke dosyası yalnızca bulunduğu belirli sunucuya erişimi etkiler. Örneğin, https://www.adobe.com:8080/crossdomain.xml konumunda bulunan bir ilke dosyası yalnızca bağlantı noktası 8080'de HTTPS üzerinden www.adobe.com adresine yapılan veri yükleme çağrıları için geçerlidir.

URL ilke dosyasında erişim izinlerini belirtme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

İlke dosyası, tek bir `<cross-domain-policy>` etiketi içerir ve bu etiket de sıfır veya daha fazla `<allow-access-from>` etiketi içerir. Her `<allow-access-from>` etiketi, tam bir IP adresini, tam bir etki alanını veya bir joker etki alanını (herhangi bir etki alanı) belirten `domain` adında bir nitelik içerir. Joker etki alanları iki yöntemden biri kullanılarak belirtilir:

- Tüm etki alanlarıyla ve tüm IP adresleriyle eşleşecek şekilde tek bir yıldız işareti (*) ile
- Yalnızca belirtilen sonekle biten etki alanlarıyla eşleşecek şekilde ardından bir sonek gelen yıldız işareti ile

Soneklerin bir nokta ile başlaması gerekir. Ancak sonek içeren joker etki alanları, baştaki nokta olmadan yalnızca soneki içeren etki alanlarıyla eşleşebilir. Örneğin, xyz.com, *.xyz.com'un bir parçası olarak değerlendirilir. IP etki alanı belirtimlerinde joker karakterlere izin verilmez.

Aşağıdaki örnek, *.example.com, www.friendOfExample.com ve 192.0.34.166 konumlarındaki SWF dosyalarına erişilmesine izin veren bir URL ilke dosyasını gösterir:

```
<?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>
```

Bir IP adresi belirtirseniz, yalnızca o IP adresinden yüklenen ve IP sözdizimini kullanan SWF dosyalarına (örneğin, http://65.57.83.12/flashmovie.swf) erişim hakkı verilir. Etki alanı adı sözdizimini kullanan SWF dosyalarına erişim hakkı verilmez. Flash Player, DNS çözünürlüğü uygulamaz.

Aşağıdaki örnekte gösterildiği gibi, herhangi bir etki alanında bulunan belgelere erişilmesine izin verebilirsiniz:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

Her `<allow-access-from>` etiketi aynı zamanda varsayılan olarak `true` değerine ayarlı olan isteğe bağlı `secure` niteliğine de sahiptir. İlke dosyanız bir HTTPS sunucusunda bulunuyorsa ve HTTPS olmayan bir sunucuda bulunan SWF dosyalarının, HTTPS sunucusundan veri yüklemesine izin vermek istiyorsanız, bu niteliği `false` değerine ayarlayabilirsiniz.

`secure` niteliğinin `false` değerine ayarlanması, HTTPS tarafından sağlanan güvenlikten taviz verilmesine neden olabilir. Özellikle, bu niteliğin `false` değerine ayarlanmasıyla güvenli içerik, gözetleme ve sahtekarlık amaçlı saldırılara açık duruma getirilir. Adobe, `secure` niteliğini kesinlikle `false` değerine ayarlamamanızı önerir.

Yüklenecek veriler HTTPS sunucusundaysa ancak verileri yükleyen SWF dosyası HTTP sunucusunda bulunuyorsa, Adobe, yükleyen SWF dosyasını bir HTTPS sunucusuna taşımanızı önerir. Bu işlem, güvenli verilerinizin tüm kopyalarını HTTPS koruması altında tutmanıza olanak sağlar. Ancak, yükleyen SWF dosyasını bir HTTP sunucusunda tutmanız gerektiğini düşünüyorsanız, aşağıdaki kodda gösterildiği gibi, `secure="false"` niteliğini `<allow-access-from>` etiketine ekleyin:

```
<allow-access-from domain="www.example.com" secure="false" />
```

Erişime izin vermek için kullanabileceğiniz başka bir öğe de `allow-http-request-headers-from` etiketidir. Bu öğe, başka bir izin etki alanındaki içerikleri barındıran bir istemciye, etki alanınıza kullanıcı tanımlı başlıklar gönderme izni verir. `<allow-access-from>` etiketi, diğer etki alanlarına etki alanınızdan veri alma izni verirken, `allow-http-request-headers-from` etiketi de diğer etki alanlarına sizin veri alanınıza başlık şeklinde veri verme izni sağlar. Aşağıdaki örnekte, tüm etki alanlarının geçerli etki alanına SOAPAction başlığı göndermesine izin verilir:

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

`allow-http-request-headers-from` deyimi ana ilke dosyasındaysa, ana bilgisayardaki tüm dizinler için geçerli olur. Aksi takdirde, yalnızca deyimi içeren ilke dosyasının dizini ve alt dizinleri için geçerli olur.

İlke dosyalarını önceden yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir sunucudan veri yüklenmesi veya bir sokete bağlanılması eşzamansız bir işlemdir. Flash Player, ana işlemi başlatmadan önce ilke dosyasının indirilmesinin bitmesini bekler. Ancak görüntülerden piksel verilerinin ayıklanması veya seslerden örnek verilerin ayıklanması eşzamanlı bir işlemdir. Verileri ayıklayabilmeniz için öncelikle ilke dosyasının yüklenmesi gerekir. Ortamı yüklediğinizde, ortamın bir ilke dosyasını denetleyeceğini belirtin:

- `Loader.load()` yöntemini kullanırken, LoaderContext nesnesi olan `context` parametresinin `checkPolicyFile` özelliğini ayarlayın.
- `<img>` etiketini kullanarak bir metin alanına görüntü gömerken, aşağıdaki gibi, `<img>` etiketinin `checkPolicyFile` niteliğini `"true"` değerine ayarlayın:

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- `Sound.load()` yöntemini kullanırken, SoundLoaderContext nesnesi olan `context` parametresinin `checkPolicyFile` özelliğini ayarlayın.
- NetStream sınıfını kullanırken, NetStream nesnesinin `checkPolicyFile` özelliğini ayarlayın.

Bu parametrelerden birini ayarladığınızda, Flash Player uygulaması ilk önce bu etki alanı için önceden indirdiği ilke dosyalarını kontrol eder. Daha sonra hem `<allow-access-from>` deyimlerini hem de meta-ilke olup olmadığını kontrol ederek sunucudaki varsayılan konumda ilke dosyasını arar. Son olarak, `Security.loadPolicyFile()` yöntemine yapılan bekleyen çağrıların kapsamda olup olmadığını görmek için bu çağrıları inceler.

Yazar (geliştirici) denetimleri

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Güvenlik ayrıcalıkları vermek için kullanılan ana ActionScript API'si, `Security.allowDomain()` yöntemi olup bu yöntem, belirttiğiniz etki alanlarındaki SWF dosyalarına ayrıcalık verir. Aşağıdaki örnekte, SWF dosyası, www.example.com etki alanından sunulan SWF dosyalarına erişim sağlar:

```
Security.allowDomain("www.example.com")
```

Bu yöntem, şu eylemler için izin verir:

- SWF dosyaları arasında çapraz komut dosyası oluşturma (bkz. "Çapraz komut dosyası oluşturma" sayfa 1074)
- Görüntüleme listesi erişimi (bkz. "Görüntüleme listesinde geçiş yapma" sayfa 1076)
- Olay algılama (bkz. "Olay güvenliği" sayfa 1077)
- Stage nesnesinin özellik ve yöntemlerine tam erişim (bkz. "Sahne Alanı güvenliği" sayfa 1075)

`Security.allowDomain()` yönteminin çağrılmasının birincil amacı, dış bir etki alanında bulunan SWF dosyaları için `Security.allowDomain()` yöntemini çağırarak SWF dosyasının komut dosyasını yazma izni vermektir. Daha fazla bilgi için, bkz. "Çapraz komut dosyası oluşturma" sayfa 1074.

`Security.allowDomain()` yöntemine parametre olarak bir IP adresinin belirtilmesi, belirtilen IP adresinden ortaya çıkan tüm tarafların erişimine izin verilmesini sağlamaz. Bunun yerine, o IP adresine eşlenen bir etki alanının değil, yalnızca belirtilen IP adresini URL olarak içeren bir tarafın erişimine izin verilir. Örneğin, www.example.com etki alanı 192.0.34.166 IP adresiyle eşlenirse, `Security.allowDomain("192.0.34.166")` yöntemine yapılan bir çağrı, www.example.com adresine erişim sağlamaz.

Tüm etki alanlarına erişilmesine izin vermek için, `"*"` joker karakterini `Security.allowDomain()` yöntemine iletebilirsiniz. Bu *tüm* etki alanlarındaki SWF dosyalarına, çağıran SWF dosyasının komut dosyasını yazma izni verdiğinden, `"*"` joker karakterini dikkatli kullanın.

ActionScript, `Security.allowInsecureDomain()` adı verilen ikinci bir izin API'si içerir. Bu yöntem `Security.allowDomain()` yöntemiyle aynı şeyi yapar, tek fark, bu yöntem güvenli bir HTTPS bağlantısı tarafından sunulan bir SWF dosyasından çağrıldığında, ayrıca HTTP gibi güvenli olmayan bir protokolden sunulan diğer SWF dosyalarının, çağıran SWF dosyasına erişmesine de izin vermesidir. Ancak, güvenli bir protokolden (HTTPS) gelen dosyalar ile güvenli olmayan protokollerden (örn. HTTP) gelen dosyalar arasında komut dosyası oluşturulmasına izin vermek iyi bir güvenlik uygulaması değildir; bu durumda güvenli içerik, gözetleme ve sahtekarlık amaçlı saldırılara açık duruma getirilir. Bu saldırılar şu şekilde uygulanır: `Security.allowInsecureDomain()` yöntemi, HTTP bağlantıları üzerinden sunulan SWF dosyalarının, güvenli HTTPS verilerinize erişmesine izin verir, HTTP sunucunuz ile kullanıcılarınız arasında bulunan bir saldırgan, HTTP SWF dosyanızı kendi dosyalarından biriyle değiştirip böylece sizin HTTPS verilerinize erişebilir.

Önemli: *AIR uygulamasının sanal alanında kod yürütülmesi Güvenlik sınıfının `allowDomain()` veya `allowInsecureDomain()` yöntemlerinden birinin çağırılmasına izin vermez.*

Güvenlikle ilgili başka bir önemli yöntem de, Flash Player uygulamasının standart olmayan bir konumda ilke dosyasını kontrol etmesini sağlayan `Security.loadPolicyFile()` yöntemidir. Daha fazla bilgi için, bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062.

Ağ iletişim API'lerini kısıtlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Ağ iletişim API'leri iki şekilde kısıtlanabilir. Kötü amaçlı etkinliği önlemek için, yaygın olarak kısıtlandırılmış bağlantı noktalarına erişim engellenir; kodunuzda bu engelleri geçersiz kılamazsınız. Bir SWF dosyasının, diğer bağlantı noktalarıyla ilgili ağ işlevlerine erişimini denetlemek için, `allowNetworking` ayarını kullanabilirsiniz.

Engellenen bağlantı noktaları

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve Adobe AIR uygulamaları, tıpkı tarayıcılar gibi belirli bağlantı noktalarına HTTP erişiminde kısıtlamalar uygular. HTTP türünde olmayan sunucular için geleneksel olarak kullanılan belirli standart bağlantı noktalarına HTTP isteklerine izin verilmez.

Bir ağ URL'sine erişen tüm API'ler bu bağlantı noktası engelleme kısıtlamalarına tabidir. Tek istisna, `Socket.connect()` ve `XMLSocket.connect()` gibi doğrudan soketleri çağıran API'ler veya soket ilke dosyasının yüklendiği `Security.loadPolicyFile()` öğesine yapılan çağrılardır. Hedef sunucuda soket ilke dosyaları kullanılarak soket bağlantılarına izin verilir veya soket bağlantıları reddedilir.

Aşağıdaki listede, bağlantı noktası engellemesinin geçerli olduğu ActionScript 3.0 API'leri gösterilmektedir:

`FileReference.download()`, `FileReference.upload()`, `Loader.load()`, `Loader.loadBytes()`, `navigateToURL()`, `NetConnection.call()`, `NetConnection.connect()`, `NetStream.play()`, `Security.loadPolicyFile()`, `sendToURL()`, `Sound.load()`, `URLLoader.load()`, `URLStream.load()`

Ayrıca Shared Library içe aktarılması, metin alanlarında `<img>` etiketinin kullanılması ve `<object>` ve `<embed>` etiketleri kullanılarak HTML sayfasında SWF dosyalarının yüklenmesi işlemlerinde de bağlantı noktası engelleme geçerlidir.

Ayrıca metin alanlarında `<img>` etiketinin kullanılması ve `<object>` ve `<embed>` etiketleri kullanılarak HTML sayfasında SWF dosyalarının yüklenmesi işlemlerinde de bağlantı noktası engelleme geçerlidir.

Aşağıdaki listelerde, hangi bağlantı noktalarının engellendiği gösterilmektedir:

HTTP: 20 (ftp verileri), 21 (ftp denetimi)

HTTP ve 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)

allowNetworking parametresini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SWF içeriğini barındıran HTML sayfasında `<object>` ve `<embed>` etiketlerinde `allowNetworking` parametresini ayarlayarak bir SWF dosyasının ağ işlevlerine erişimini denetleyebilirsiniz.

Olası `allowNetworking` değerleri şunlardır:

- `"all"` (varsayılan)—SWF dosyasında tüm ağ iletişimi API'lerine izin verilir.
- `"internal"`—SWF dosyası, tarayıcı gezinmesini veya bu bölümün ilerleyen kısımlarında listelenen tarayıcı etkileşimi API'lerini çağıramaz ancak diğer ağ iletişimi API'lerini çağırabilir.
- `"none"`—SWF dosyası, tarayıcı gezinmesini veya bu bölümün ilerleyen kısımlarında listelenen tarayıcı etkileşimi API'lerini çağıramaz ve yine ilerleyen bölümlerde listelenen herhangi bir SWF'den SWF'ye iletişim API'sini kullanamaz.

`allowNetworking` parametresi birincil olarak SWF dosyası ve kapsayan HTML sayfası farklı etki alanlarından olduğunda kullanılmak üzere tasarlanmıştır. SWF dosyasının her zaman planladığınız HTML sayfasıyla yüklenmesini sağlayamadığınızdan, SWF dosyası, kapsayan HTML sayfalarıyla aynı etki alanından geldiğinde `"internal"` veya `"none"` değerinin kullanılması önerilmez. Güvenilmeyen taraflar, kapsayan bir HTML olmadan etki alanınızda bir SWF'yi yükleyebilir ve bu durumda da `allowNetworking` kısıtlaması tasarladığınız şekilde çalışmaz.

Önlenmiş bir API çağrıldığında SecurityError istisnası atılır.

Aşağıdaki örnekte gösterildiği gibi, `allowNetworking` parametresini ekleyin ve SWF dosyasının başvurusunu içeren HTML sayfasında `<object>` ve `<embed>` etiketlerinde bu parametrenin değerini ayarlayın:

```
<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>
```

HTML sayfası aynı zamanda SWF gömme etiketlerini oluşturmak için bir komut dosyası da kullanabilir. Komut dosyasını uygun `allowNetworking` ayarlarını ekleyecek şekilde değiştirmeniz gerekir. Adobe Flash Professional ve Adobe Flash Builder tarafından oluşturulan HTML sayfaları, SWF dosyalarının başvurularını gömmek için `AC_FL_RunContent()` işlevini kullanır. Aşağıdaki gibi, komut dosyasına `allowNetworking` parametresi ayarlarını ekleyin:

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

`allowNetworking` öğesi `"internal"` olarak ayarlandığında aşağıdaki API'ler önlenir:

`navigateToURL()`, `fscommand()`, `ExternalInterface.call()`

`allowNetworking` öğesi `"none"` olarak ayarlandığında, önceki listede bulunan API'lere ek olarak şu API'ler de önlenir:

`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()`

Seçilen `allowNetworking` ayarı, bir SWF dosyasının ağ iletişimi API'sini kullanmasına izin verse de, güvenlik sanal alanı sınırlamalarına bağlı olarak başka kısıtlamalar da olabilir (bkz. "Güvenlik sanal alanları" sayfa 1055).

`allowNetworking` öğesi `"none"` olarak ayarlandığında, bir TextField nesnesinin `htmlText` özelliğinde `<img>` etiketindeki harici ortama başvuramazsınız. (SecurityError istisnası atılır.)

`allowNetworking` öğesi `"none"` olarak ayarlandığında, Flash Professional'da (ActionScript'te değil) eklenmiş içe aktarılan bir paylaşılan kütüphanedeki sembol, çalışma zamanında engellenir.

Tam ekran modu güvenliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player 9.0.27.0 ve sonrası, Flash Player'da çalışan içeriğin ekranın tamamını doldurabildiği tam ekran modunu destekler. Tam ekran moduna girmek için, Sahne Alanı'nın `displayState` özelliği `StageDisplayState.FULL_SCREEN` sabitine ayarlanır. Daha fazla bilgi için, bkz. "Tam ekran modunda çalışma" sayfa 169.

Uzak bir sanal alan içerisinde çalıştırılan SWF dosyaları için güvenlikle ilgili dikkate alınması gereken birkaç önemli nokta vardır.

Tam ekran modunu etkinleştirmek için, aşağıdaki örnekte gösterildiği gibi, SWF dosyasının başvurusunu içeren HTML sayfasındaki `<object>` ve `<embed>` etiketlerine `"true"` değerine ayarlanmış şekilde (varsayılan değer `"false"` şeklindedir) `allowFullScreen` parametresini ekleyin:

```
<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>
```

HTML sayfası aynı zamanda SWF gömme etiketlerini oluşturmak için bir komut dosyası da kullanabilir. Komut dosyasını uygun `allowFullScreen` ayarlarını ekleyecek şekilde değiştirmeniz gerekir. Flash Professional ve Flash Builder tarafından oluşturulan HTML sayfaları, SWF dosyalarına başvuru gömmek için `AC_FL_RunContent()` işlevini kullanır ve aşağıdaki örnekte de görüldüğü gibi `allowFullScreen` parametre ayarlarını eklemeniz gerekir:

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

Tam ekran modunu başlatan ActionScript uygulaması yalnızca bir fare olayına veya klavye olayına yanıt olarak çağrılabilir. Başka durumlarda çağrılırsa, Flash Player bir istisna atar.

İçerik tam ekran moduna girdiğinde bir mesaj görüntülenerek kullanıcıya nasıl tam ekran modundan çıkıp normal moda geri döneceğini bildirir. Mesaj birkaç saniye görüntülenir ve sonra uzaklaşarak kaybolur.

Tarayıcıda çalışan içerik için, klavye kullanımı tam ekran modunda kısıtlandırılmıştır. Flash Player 9'da, yalnızca Escape tuşu gibi uygulamayı normal moda döndüren klavye kısayolları desteklenir. Kullanıcılar metin alanlarına metin giremez veya ekranda dolanamaz. Flash Player 10 ve sonraki sürümlerinde bazı basılamaz tuşlar (ok tuşları, boşluk ve Sekme tuşu) desteklenmektedir. Ancak metin girdisi engellenmektedir.

Bağımsız oynatıcıda veya bir projektör dosyasında her zaman tam ekran moduna izin verilir. Ayrıca bu ortamlarda klavye kullanımı (metin girdisi dahil) tam desteklenir.

Stage nesnesinin `displayState` özelliğinin çağrılması, Stage sahibiyle (ana SWF dosyası) aynı güvenlik sanal alanında bulunmayan çağıranlar için bir istisna atar. Daha fazla bilgi için, bkz. "Sahne Alanı güvenliği" sayfa 1075.

Yöneticiler, mms.cfg dosyasında `FullScreenDisable = 1` ayarını yaparak tarayıcılarda çalışan SWF dosyaları için tam ekran modunu devre dışı bırakabilir. Ayrıntılar için bkz. "Yönetimsel denetimler" sayfa 1060.

Tarayıcıda, tam ekran moduna izin verilmesi için HTML sayfasında bir SWF dosyasının bulunması gerekir.

Tam ekran etkileşimli modu güvenliği

Flash Player 11.3 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player 11.3 ve üstü, Flash Player'da çalışan içeriğin ekranın tamamını doldurabildiği *ve metin girişini kabul ettiği* tam ekran etkileşimli modunu destekler. Tam ekran etkileşimli moduna girmek için, Sahne Alanı'nın `displayState` özelliği `StageDisplayState.FULL_SCREEN_INTERACTIVE` sabitine ayarlanır. Daha fazla bilgi için bkz. "Tam ekran modunda çalışma" sayfa 169.

Uzak bir korumalı alan içerisinde çalıştırılan SWF dosyaları için güvenlikle ilgili dikkate alınması gereken birkaç önemli nokta vardır.

Tam ekran modunu etkinleştirmek için, aşağıdaki örnekte gösterildiği gibi, SWF dosyasının başvurusunu içeren HTML sayfasındaki `<object>` ve `<embed>` etiketlerine `"true"` değerine ayarlanmış şekilde (varsayılan değer `"false"` şeklindedir) `allowFullScreenInteractive` parametresini ekleyin:

```
<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>
```

HTML sayfası aynı zamanda SWF gömme etiketlerini oluşturmak için bir komut dosyası da kullanabilir. Komut dosyasını uygun `allowFullScreenInteractive` ayarlarını ekleyecek şekilde değiştirmeniz gerekir. Flash Professional ve Flash Builder tarafından oluşturulan HTML sayfaları, SWF dosyalarına başvuru gömmek için `AC_FL_RunContent()` işlevini kullanır ve aşağıdaki örnekte görüldüğü gibi `allowFullScreenInteractive` parametresi ayarlarını eklemeniz gerekir:

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

Tam ekran etkileşimli modunu başlatan ActionScript uygulaması yalnızca bir fare olayına veya klavye olayına yanıt olarak çağrılabilir. Başka durumlarda çağrılırsa, Flash Player bir istisna oluşturur.

İçerik, tam ekran etkileşimli moduna girdiğinde bir kaplama mesajı görünür. Mesaj; tam ekran sayfasının etki alanını, tam ekran modundan çıkmayla ilgili talimatları ve bir **İzin Ver** düğmesi görüntüler. Kullanıcı, tam ekran etkileşimli modunda olduğunu onaylayan **İzin Ver** öğesini tıklatana dek kaplama ekranda kalmaya devam eder.

Yöneticiler, mms.cfg dosyasında `FullScreenInteractiveDisable = 1` ayarını yaparak tarayıcılarda çalışan SWF dosyaları için tam ekran etkileşimli modunu devre dışı bırakabilir. Ayrıntılar için bkz. "Yönetimsel denetimler" sayfa 1060.

Tarayıcıda, tam ekran etkileşimli moduna izin verilmesi için HTML sayfasında bir SWF dosyasının bulunması gerekir.

İçerik yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve AIR içeriği aşağıdakilerin de yer aldığı birçok türde başka içerik yükleyebilir:

- SWF dosyaları
- Görüntüler
- Ses
- Video
- HTML dosyaları (yalnızca AIR)
- JavaScript (yalnızca AIR)

SWF dosyalarını ve görüntülerini Yükleyici sınıfıyla yükleyin

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

SWF dosyalarını ve görüntüleri (JPG, GIF veya PNG dosyaları) yüklemek için Loader sınıfını kullanırsınız. Dosya sistemiyle yerel sanal alanda bulunanlar dışında herhangi bir SWF dosyası, herhangi bir ağ etki alanından SWF dosyalarını ve görüntülerini yükleyebilir. Yalnızca yerel sanal alanlardaki SWF dosyaları, yerel dosya sisteminden SWF dosyalarını ve görüntüleri yükleyebilir. Ancak, ağ iletişimiyle yerel sanal alandaki dosyalar yalnızca yerel güvenilen veya ağ iletişimiyle yerel sanal alanda bulunan yerel SWF dosyalarını yükleyebilir. Ne var ki, ağ iletişimiyle yerel sanal alanda bulunan SWF dosyaları, SWF dosyaları dışındaki yerel içerikleri (örn. görüntüler) yükler ancak yüklenen içerikteki verilere erişemez.

Güven onayı verilmeyen bir kaynaktan (örn. Loader nesnesinin kök SWF dosyasının etki alanı dışındaki bir etki alanından) SWF dosyası yüklerken, aşağıdaki kodda olduğu gibi, Loader nesnesi için bir maske tanımlayarak, yüklenen içeriğin (bu, Loader nesnesinin bir alt öğesidir), Sahne Alanı'nın o maske dışında kalan kısımlarına çizim yapmasını önlemek isteyebilirsiniz:

```
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);
```

Loader nesnesinin `load()` yöntemini çağırdığınızda, LoaderContext nesnesi olan bir `context` parametresini belirtebilirsiniz. LoaderContext sınıfı, yüklenen içeriğin kullanılabileceği bağlamı tanımlamanızı sağlayan üç özellik içerir:

- `checkPolicyFile`: Yalnızca bir görüntü dosyası (SWF dosyası değil) yüklerken bu özelliği kullanın. Loader nesnesini içeren dosyanın etki alanı dışındaki bir etki alanında bulunan görüntü dosyası için bunu belirtin. Bu özelliği `true` değerine ayarlarsanız, Loader öğesi URL ilke dosyası için kaynak sunucuyu kontrol eder (bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062). Sunucu, Loader etki alanına izin verirse, Loader etki alanındaki SWF dosyalarında bulunan ActionScript, yüklenen görüntüdeki verilere erişebilir. Başka bir deyişle, yüklenen görüntüyü temsil eden Bitmap nesnesine başvuruyu almak için `Loader.content` özelliğini veya yüklenen görüntüdeki piksellere erişmek için `BitmapData.draw()` ya da `BitmapData.drawWithQuality()` yöntemini kullanabilirsiniz. `drawWithQuality` yöntemi Flash Player 11.3 ve üstü ile AIR 3.3 ve üstünde kullanılabilir.
- `securityDomain`: Yalnızca bir SWF dosyası (görüntü değil) yüklerken bu özelliği kullanın. Loader nesnesini içeren dosyanın etki alanı dışındaki bir etki alanında bulunan SWF dosyası için bunu belirtin. Şu anda `securityDomain` özelliği için yalnızca iki değer desteklenir: `null` (varsayılan) ve `SecurityDomain.currentDomain`. `SecurityDomain.currentDomain` öğesini belirtirseniz, bu, yüklenen SWF dosyasının, yükleyen SWF dosyasının sanal alanına *içe aktarılmasını* ister, başka bir deyişle, yükleyen SWF dosyasının kendi sunucusundan yüklenmiş gibi çalışır. Yalnızca yüklenen SWF dosyasının sunucusunda bir URL ilke dosyası bulunduğunda ve böylece yükleyen SWF dosyasının etki alanının erişimine izin verildiğinde buna izin verilir. Gerekli ilke dosyası bulunursa, yükleyen ve yüklenen aynı sanal alanda bulunduğundan, yükleme başladıktan sonra serbestçe birbirinin komut dosyasını oluşturabilir. Sanal alan içe aktarmasının yerine daha çok normal bir yükleme gerçekleştirildiğini ve sonra da yüklenen SWF dosyasının `Security.allowDomain()` yöntemini çağırmasının sağlandığını unutmayın. Yüklenen SWF dosyası böylece kendi doğal sanal alanında bulunup kendi gerçek sunucusundaki kaynaklara erişebileceğinden, bu ikinci yöntemin kullanımı daha kolay olabilir.
- `applicationDomain`: Yalnızca ActionScript 3.0'da yazılmış bir SWF dosyası (ActionScript 1.0 veya 2.0'da yazılmış bir görüntü ya da SWF dosyası değil) yüklenirken bu özelliği kullanın. Dosyayı yüklerken, varsayılan olarak dosyanın, yükleyen SWF dosyasının uygulama etki alanının alt öğesi olan yeni bir uygulama etki alanına yerleştirilmesi yerine belirli bir uygulama etki alanına yerleştirilmesini belirtebilirsiniz. Uygulama etki alanlarının, güvenlik etki alanlarının alt birimleri olduğunu ve bu nedenle de yalnızca yüklediğiniz SWF dosyası, kendi sunucunuzda bulunduğundan dolayı veya siz `securityDomain` özelliğini kullanarak dosyayı güvenlik etki alanınıza başarıyla içe aktardığınızdan dolayı kendi güvenlik etki alanınızdan geldiğinde, bir hedef uygulama etki alanı belirtebildiğinizi unutmayın. Bir uygulama etki alanı belirtirseniz ancak yüklenen SWF dosyası farklı bir güvenlik etki alanının parçasıysa, `applicationDomain` öğesinde belirttiğiniz etki alanı yoksayılır. Ayrıntılı bilgi için bkz. "Uygulama etki alanlarıyla çalışma" sayfa 149.

Ayrıntılar için, bkz. "Yükleme içeriği belirtme" sayfa 204.

Loader nesnesinin önemli bir özelliği, LoaderInfo nesnesi olan `contentLoaderInfo` özelliğidir. Diğer birçok nesnenin aksine LoaderInfo nesnesi, yükleyen SWF dosyası ile yüklenen içerik arasında paylaşılır ve her zaman iki taraf da nesneye erişebilir. Yüklenen içerik bir SWF dosyası olduğunda, `DisplayObject.loaderInfo` özelliği üzerinden LoaderInfo nesnesine erişebilir. LoaderInfo nesneleri, yükleme ilerlemesi, yükleyenin ve yüklenenin URL'leri, yükleyen ile yüklenen arasındaki güven onayı ilişkisi, vb. bilgileri içerir. Daha fazla bilgi için, bkz. "Yükleme ilerlemesini izleme" sayfa 202.

Ses ve video yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Dosya sistemiyle yerel sanal alanda bulunanlar dışında tüm içerik, `Sound.load()`, `NetConnection.connect()` ve `NetStream.play()` yöntemlerini kullanarak ağ kaynaklarından ses ve video yüklemesine izin verilir.

Yalnızca dosya sistemiyle yerel ve AIR uygulaması sanal alanlarındaki içerik, yerel dosya sisteminden yerel medya yükleyebilir. Bu yüklenmiş dosyalara yalnızca dosya sistemiyle yerel sanal alanı ve AIR uygulama sanal alanı ya da yerel-güvenilir sanal alanı erişebilir.

Yüklenen ortamdaki verilere erişilmesine yönelik başka kısıtlamalar da vardır. Ayrıntılar için, bkz. "Veri olarak yüklenen ortama erişme" sayfa 1077.

Metin alanında <img> etiketini kullanarak SWF dosyalarını ve görüntüleri yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki kodda olduğu gibi, `<img>` etiketini kullanarak bir metin alanına SWF dosyaları ve bitmapler yükleyebilirsiniz:

```
<img src = 'filename.jpg' id = 'instanceName' >
```

Aşağıdaki kodda olduğu gibi, TextField örneğinin `getImageReference()` yöntemini kullanarak bu şekilde yüklenen içeriğe erişebilirsiniz:

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

Ancak, bu şekilde yüklenen SWF dosyalarının ve görüntülerin, kaynaklarına karşılık gelen sanal alana yerleştirildiğini unutmayın.

Bir metin alanında `<img>` etiketini kullanarak görüntü dosyası eklediğinizde, görüntüdeki verilere erişilmesine, bir URL ilke dosyası tarafından izin verilir. Aşağıdaki kodda olduğu gibi, `<img>` etiketine bir `checkPolicyFile` niteliği ekleyerek ilke dosyasını kontrol edebilirsiniz:

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

Bir metin alanında `<img>` etiketini kullanarak SWF dosyası yüklediğinizde, `Security.allowDomain()` yöntemine yapılan bir çağrıyla, o SWF dosyasının verilerine erişilmesine izin verebilirsiniz.

Harici bir dosya yüklemek için (SWF dosyanızda gömülü bir Bitmap sınıfını kullanmanın aksine) bir metin alanında `<img>` etiketini kullandığınızda, TextField nesnesinin alt öğesi olarak otomatik şekilde bir Loader nesnesi oluşturulur ve dosya yüklemek için ActionScript'te bir Loader nesnesini kullanmışsınız gibi o Loader nesnesine harici dosya yüklenir. Bu durumda, `getImageReference()` yöntemi, otomatik olarak oluşturulan Loader nesnesini döndürür. Bu Loader nesnesi, çağıran kodla aynı güvenlik sanal alanında bulunduğundan, bu Loader nesnesine erişmek için güvenlik denetimi gerekmez.

Ancak, yüklenen ortama erişmek için Loader nesnesinin `content` özelliğine başvurduğunuzda güvenlik kuralları geçerli olur. İçerik bir görüntü olursa, bir URL ilke dosyası uygulamanız gerekir ve içerik bir SWF dosyası olursa SWF dosyasındaki kodun `allowDomain()` yöntemini çağırmasını sağlamanız gerekir.

Adobe AIR

Uygulama sanal alanında bir metin alanındaki <img> etiketleri, olta saldırılarını önlemek amacıyla göz ardı edilir. Ayrıca, uygulamanın sanal alanında yürütülen kod, Güvenlik `allowDomain()` yönteminin çağırılmasına izin vermez.

RTMP sunucuları kullanılarak teslim edilen içerik

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Media Server, veri, ses ve video sunmak için Real-Time Media Protocol (RTMP) kullanır. NetConnection sınıfının `connect()` yöntemini kullanıp bir RTMP URL'sini parametre olarak ileterek bu medyayı yükleyebilirsiniz. Flash Media Server, istekte bulunan dosyanın etki alanını esas alarak bağlantıları kısıtlayabilir ve içeriğin indirilmesini önleyebilir. Ayrıntılar için, çevrimiçi Flash Media Server belgelerine www.adobe.com/go/learn_fms_docs_tr adresinden bakın.

RTMP akışlarından çalışma zamanı grafikleri ve ses verileri ayıklamak amacıyla `BitmapData.draw()`, `BitmapData.drawWithQuality()` ve `SoundMixer.computeSpectrum()` yöntemlerini kullanmak için sunucuya erişime izin vermelisiniz. Flash Media Server'daki belli dizinlere erişime izin vermek için Sunucu-Tarafı ActionScript `Client.videoSampleAccess` ve `Client.audioSampleAccess` özelliklerini kullanın. Daha fazla bilgi için bkz. Sunucu Tarafı ActionScript Dil Başvurusu. (`drawWithQuality` yöntemi Flash Player 11.3 ve üstü ile AIR 3.3 ve üstünde kullanılabilir.)

Çapraz komut dosyası oluşturma

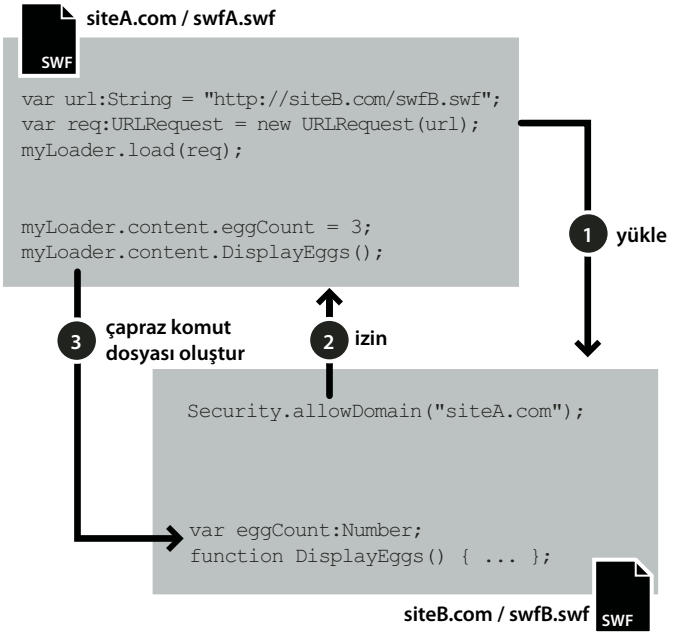
Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

ActionScript 3.0 ile yazılmış iki SWF dosyası veya AIR uygulamasında çalıştırılan iki HTML dosyası aynı alandan sunuluyorsa—örneğin, bir SWF dosyasının URL'si http://www.example.com/swfA.swf ve diğerinin URL'si http://www.example.com/swfB.swf olursa—bu durumda bir dosyada tanımlanan kod, diğerinin değişkenlerini, nesneleri, özellikleri, yöntemleri vb.'ni inceleyebilir ve değiştirebilir. Buna *çapraz komut dosyası oluşturma* denir.

Eğer iki dosya farklı alanlardan sunulursa, örneğin http://siteA.com/swfA.swf ve http://siteB.com/swfB.swf gibi, —varsayılan olarak Flash Player ve AIR ne swfA.swf dosyasının, swfB.swf komut dosyasını oluşturmasına; ne de swfB.swf dosyasının swfA.swf komut dosyasını oluşturmasına izin vermez. SWF dosyası, `Security.allowDomain()` öğesini çağırarak başka etki alanlarındaki SWF dosyalarına izin verir. swfB.swf dosyası `Security.allowDomain("siteA.com")` öğesini çağırarak, siteA.com konumundaki SWF dosyalarının swfB.swf dosyasının komut dosyasını oluşturmasına izin verir.

AVM1 SWF dosyaları ile AVM2 SWF dosyaları arasında çapraz komut dosyası oluşturma desteklenmez. AVM1 SWF dosyası, ActionScript 1.0 veya ActionScript 2.0 kullanılarak oluşturulmuş bir SWF dosyasıdır. (AVM1 ve AVM2, ActionScript Virtual Machine'i ifade eder.) Ancak AVM1 ve AVM2 arasında veri göndermek için LocalConnection sınıfını kullanabilirsiniz.

Etki alanlar arası durumlarda iki tarafın da net olması önemlidir. Bu açıklamanın amaçları doğrultusunda, çapraz komut dosyası oluşturma işlemini gerçekleştiren taraf *erişen taraf* (genellikle SWF dosyasına erişen taraf) ve diğer taraf da *erişilen taraf* (genellikle erişilen SWF) olarak adlandırılır. Aşağıdaki resimde gösterildiği gibi, siteA.swf dosyası siteB.swf dosyasının komut dosyasını oluşturduğunda, siteA.swf erişen taraf, siteB.swf ise erişilen taraf olur:



`Security.allowDomain()` yöntemiyle oluşturulan etki alanları arası izinler asimetriktir. Önceki örnekte, siteA.swf dosyası siteB.swf dosyasının komut dosyasını oluşturabilir ancak siteA.swf dosyası siteB.com konumundaki SWF dosyalarının siteA.swf dosyasının komut dosyasını oluşturmasına izin vermek için `Security.allowDomain()` yöntemini çağırmadığından, siteB.swf dosyası siteA.swf dosyasının komut dosyasını oluşturamaz Her iki SWF dosyasının `Security.allowDomain()` yöntemini çağırmasını sağlayarak simetrik izinler ayarlayabilirsiniz.

Flash Player, SWF dosyalarını, başka SWF dosyaları tarafından oluşturulan etki alanları arası komut dosyası oluşturmaya karşı korumaya ek olarak, HTML dosyaları tarafından oluşturulan etki alanları arası komut dosyası oluşturmaya karşı da korur. HTML'den SWF'ye komut dosyası oluşturma, `ExternalInterface.addCallback()` yöntemi üzerinden oluşturulan geri çağrılarla gerçekleşebilir. HTML'den SWF'ye komut dosyası oluşturma, etki alanları arasında gerçekleşir, erişilen SWF dosyasının, tıpkı erişen taraf bir SWF dosyası olduğunda olduğu gibi `Security.allowDomain()` yöntemini çağırması gerekir, aksi takdirde işlem başarısız olur. Daha fazla bilgi için bkz. "Yazar (geliştirici) denetimleri" sayfa 1066.

Ayrıca, Flash Player uygulaması, SWF'den HTML'e komut dosyası oluşturmaya yönelik güvenlik denetimleri sağlar. Daha fazla bilgi için, bkz. "Giden URL erişimini denetleme" sayfa 1084.

Sahne Alanı güvenliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Stage nesnesinin bazı özellikleri ve yöntemleri, görüntüleme listesindeki herhangi bir hareketli grafik veya film klibi için kullanılabilir durumdadır.

Ancak Stage nesnesinin bir sahibi vardır: yüklenen birinci SWF dosyası. Varsayılan olarak, Stage nesnesinin şu özellik ve yöntemleri yalnızca Stage sahibiyle aynı güvenlik sanal alanında bulunan SWF dosyaları için kullanılabilir durumdadır:

Özellikler	Yöntemler
`align`	`addChild()`
displayState	`addChildAt()`
frameRate	`addEventListener()`
`height`	`dispatchEvent()`
`mouseChildren`	`hasEventListener()`
`numChildren`	`setChildIndex()`
`quality`	`willTrigger()`
`scaleMode`	
`showDefaultContextMenu`	
`stageFocusRect`	
`stageHeight`	
`stageWidth`	
`tabChildren`	
`textSnapshot`	
`width`	

Sahne Alanı sahibinin sanal alanı dışındaki bir sanal alanda bulunan SWF dosyasının bu özellik ve yöntemlere erişmesi için, Sahne Alanı sahibi SWF dosyasının, harici sanal alanın etki alanına izin vermek için `Security.allowDomain()` yöntemini çağırması gerekir. Daha fazla bilgi için bkz. "Yazar (geliştirici) denetimleri" sayfa 1066.

`frameRate` özelliği özel bir durumdur—herhangi bir SWF dosyası `frameRate` özelliğini okuyabilir. Ancak, yalnızca Sahne Alanı sahibinin güvenlik sanal alanında bulunanlar (veya `Security.allowDomain()` yöntemine yapılan bir çağrıyla izin verilenler) özelliği değiştirebilir.

Stage nesnesinin `removeChildAt()` ve `swapChildrenAt()` yöntemlerinde de kısıtlamalar vardır ancak bunlar diğer kısıtlamalardan farklıdır. Bu yöntemleri çağırmak için, Sahne Alanı sahibiyle aynı etki alanında olma zorunluluğu yerine, kodun, etkilenen alt nesnelerin sahibiyle aynı etki alanında olması gerekir veya alt nesneler `Security.allowDomain()` yöntemini çağırabilir.

Görüntüleme listesinde geçiş yapma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Tek bir SWF dosyasının diğer sanal alanlardan yüklenmiş görüntüleme nesnelerine erişme yeteneği kısıtlandırılmıştır. SWF dosyasının, farklı bir sanal alanda başka bir SWF dosyası tarafından oluşturulmuş bir görüntüleme nesnesine erişmesi için, erişilen SWF dosyasının, erişen SWF dosyasının etki alanına erişim izni vermek üzere `Security.allowDomain()` yöntemini çağırması gerekir. Daha fazla bilgi için bkz. "Yazar (geliştirici) denetimleri" sayfa 1066.

Loader nesnesi tarafından yüklenen bir Bitmap nesnesine erişmek için, görüntü dosyasının kaynak sunucusunda bir URL ilke dosyasının bulunması ve bu ilke dosyasının, Bitmap nesnesine erişmeye çalışan SWF dosyasının etki alanına izin vermesi gerekir (bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062).

Yüklenen bir dosyaya (ve Loader nesnesine) karşılık gelen LoaderInfo nesnesi, yüklenen nesne ile Loader nesnesi arasındaki ilişkiyi tanımlayan şu üç özelliği içerir: `childAllowsParent`, `parentAllowsChild` ve `sameDomain`.

Olay güvenliği

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Görüntüleme listesiyle ilgili olaylar, olayı gönderen görüntüleme nesnesinin sanal alanı esas alınarak güvenlik erişimi sınırlamalarına sahiptir. Görüntüleme listesindeki bir olay köpürme ve yakalama aşamalarına sahiptir (bkz. "Olayları işleme" sayfa 126). Köpürme ve yakalama aşamalarında bir olay, kaynak görüntüleme nesnesinden, görüntüleme listesindeki üst görüntüleme nesneleri yoluyla alınır. Üst nesne, kaynak görüntüleme nesnesinden farklı bir güvenlik sanal alanındaysa, üst nesne sahibi ile kaynak nesne sahibi arasında karşılıklı güven onayı yoksa, yakalama ve köpürme aşaması bu üst nesnenin aşağısında durdurulur. Bu karşılıklı güven onayı şu şekilde elde edilebilir:

1. Üst nesneye sahip SWF dosyasının, kaynak nesneye sahip SWF dosyasının etki alanına güven onayı vermek için `Security.allowDomain()` yöntemini çağırması gerekir.
2. Kaynak nesneye sahip SWF dosyasının, üst nesneye sahip SWF dosyasının etki alanına güven onayı vermek için `Security.allowDomain()` yöntemini çağırması gerekir.

Yüklenen bir dosyaya (ve Loader nesnesine) karşılık gelen LoaderInfo nesnesi, yüklenen nesne ile Loader nesnesi arasındaki ilişkiyi tanımlayan şu iki özelliği içerir: `childAllowsParent` and `parentAllowsChild`.

Görüntüleme nesneleri dışındaki nesnelerden gönderilen olaylar için, güvenlik kontrolleri veya güvenlikle ilgili bir uygulama yoktur.

Veri olarak yüklenen ortama erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yükleme verilerine erişmek için, `BitmapData.draw()`, `BitmapData.drawWithQuality()` ve `SoundMixer.computeSpectrum()` yöntemlerini kullanın. Varsayılan olarak, farklı bir sanal alana yüklü medya tarafından görüntülenen veya oynatılan grafik veya ses nesnelerinden piksel verileri ya da ses verileri alamazsınız. Ancak, bu tip verilere sanal alan sınırları dahilinde erişim iznini vermek için aşağıdaki yöntemleri kullanabilirsiniz:

- Erişilecek veriyi görüntüleyen veya oynatan içerikte, `Security.allowDomain()` yöntemini çağırarak diğer etki alanlarındaki içeriğe veri erişimi sağlayın.
- Yüklenmiş bir görüntü, ses veya video için, yüklenen dosyanın sunucusuna bir URL ilke dosyası ekleyin. Bu ilke dosyasının, dosyadan verileri ayıklamak üzere `BitmapData.draw()`, `BitmapData.drawWithQuality()` veya `SoundMixer.computeSpectrum()` yöntemlerini çağırmaya çalışan SWF dosyasının etki alanına erişim hakkı vermesi gerekir. `drawWithQuality` yöntemi Flash Player 11.3 ve üstü ile AIR 3.3 ve üstünde kullanılabilir.

İlerleyen bölümlerde, bitmap, ses ve video verilerine erişilmesiyle ilgili ayrıntılar sağlanmıştır.

Bitmap verilerine erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir BitmapData nesnesinin `draw()` ve `drawWithQuality()` (Flash Player 11.3; AIR 3.3) yöntemleri, herhangi bir görüntüleme nesnesinin geçerli olarak görüntülenen piksellerini BitmapData nesnesine çizmenize izin verir. Bunlar arasında, MovieClip nesnesinin, Bitmap nesnesinin veya herhangi bir görüntüleme nesnesinin pikselleri yer alabilir. Bu yöntemlerin BitmapData nesnesine pikselleri çizmesi için şu koşulların karşılanması gerekir:

- Yüklenen bitmap dışında bir kaynak nesne olması durumunda, kaynak nesnenin ve (Sprite veya MovieClip nesnesi olması durumunda) kaynak nesnenin tüm alt nesnelerinin, draw yöntemini çağıran nesneyle aynı etki alanından gelmesi ya da `Security.allowDomain()` yönteminin çağrılmasıyla çağıran için erişilebilir olan bir SWF dosyasında bulunması gerekir.
- Yüklü bitmap kaynak nesnesi olması durumunda kaynak nesnenin, draw yöntemini çağıran nesneyle aynı etki alanından gelmesi veya kaynak sunucusunun, çağıran etki alana izin veren bir URL ilke dosyası içermesi gerekir.

Bu koşullar karşılanmazsa, SecurityError istisnası atılır.

Loader sınıfının `load()` yöntemini çağırdığınızda, LoaderContext nesnesi olan bir `context` parametresini belirtebilirsiniz. LoaderContext nesnesinin `checkPolicyFile` özelliğini `true` değerine ayarlarsanız, Flash Player uygulaması, görüntünün yükleneceği sunucuda bir URL ilke dosyasını kontrol eder. İlke dosyası varsa ve dosya, yükleyen SWF dosyasının etki alanına izin veriyorsa, dosyanın Bitmap nesnesindeki verilere erişmesine izin verilir; aksi takdirde erişim reddedilir.

Bir metin alanında `<img>` etiketi aracılığıyla yüklenmiş bir görüntüde `checkPolicyFile` özelliğini de belirtebilirsiniz. Ayrıntılar için, bkz. "Metin alanında <img> etiketini kullanarak SWF dosyalarını ve görüntüleri yükleme" sayfa 1073.

Ses verilerine erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki sesle ilgili ActionScript 3.0 API'leri güvenlik kısıtlamalarına sahiptir:

- `SoundMixer.computeSpectrum()` yöntemi—Ses dosyasıyla aynı güvenlik sanal alanında çalışan kod için her zaman izin verilir. Diğer sanal alanlardaki kodlar için güvenlik kontrolleri vardır.
- `SoundMixer.stopAll()` yöntemi—Ses dosyasıyla aynı güvenlik sanal alanında çalışan kod için her zaman izin verilir. Diğer sanal alanlardaki dosyalar için güvenlik kontrolleri vardır.
- Sound sınıfının `id3` özelliği—Ses dosyasıyla aynı güvenlik sanal alanında bulunan SWF dosyaları için her zaman izin verilir. Diğer sanal alanlardaki kodlar için güvenlik kontrolleri vardır.

Her ses, kendisiyle ilişkilendirilmiş iki tür sanal alana sahiptir—içerik sanal alanı ve sahip sanal alanı:

- Sesin kaynak etki alanı, içerik sanal alanını belirler ve bu da sesin `id3` özelliği ve `SoundMixer.computeSpectrum()` yöntemi aracılığıyla sesteki verilerin ayıklanıp ayıklanamayacağını belirler.
- Ses çalınmasını başlatan nesne, sahip sanal alanını belirler ve bu da `SoundMixer.stopAll()` yöntemi kullanılarak sesin durdurulup durdurulamayacağını belirler.

Sound sınıfının `load()` yöntemini kullanarak ses yüklediğinizde, SoundLoaderContext nesnesi olan bir `context` parametresini belirtebilirsiniz. SoundLoaderContext nesnesinin `checkPolicyFile` özelliğini `true` değerine ayarlarsanız, çalışma zamanı sesin yüklendiği sunucuda URL ilke dosyasını kontrol eder. İlke dosyası varsa ve dosya, yükleyen kodun etki alanına izin veriyorsa, kodun, Sound nesnesinin `id` özelliğine erişmesine izin verilir; aksi takdirde erişim reddedilir. Ayrıca, `checkPolicyFile` özelliğinin ayarlanması, yüklenen sesler için `SoundMixer.computeSpectrum()` yöntemini etkinleştirebilir.

Bir veya birkaç ses sahibinin, sanal alanı çağıran için erişilemez durumda olması nedeniyle `SoundMixer.stopAll()` yöntemine yapılan bir çağrının tüm sesleri durdurup durdurmayacağını öğrenmek için `SoundMixer.areSoundsInaccessible()` yöntemini kullanabilirsiniz.

`SoundMixer.stopAll()` yönteminin çağrılması, sahip sanal alanı, `stopAll()` yöntemini çağıranın sanal alanıyla aynı olan sesleri durdurur. Bu ayrıca `stopAll()` yöntemini çağıran SWF dosyasının etki alanına erişim izni vermek için `Security.allowDomain()` yöntemini çağıran SWF dosyaları tarafından oynatımı başlatılan sesleri de durdurur. Diğer sesler durdurulmaz ve `SoundMixer.areSoundsInaccessible()` yöntemi çağrılarak bu seslerin varlığı ortaya çıkarılabilir.

`computeSpectrum()` yönteminin çağrılması için, çalınan her sesin, yöntemi çağıran nesneyle aynı sanal alanda veya çağıranın sanal alanına erişim izni verilmiş bir kaynakta olması gerekir; aksi takdirde SecurityError istisnası atılır. Bir SWF dosyasındaki kütüphanede bulunan gömülü seslerden yüklenmiş sesler için, yüklenen SWF dosyasında `Security.allowDomain()` yöntemine yapılan bir çağrıyla izin verilir. SWF dosyaları dışındaki kaynaklardan yüklenmiş (yüklenmiş mp3 dosyalarından veya video dosyalarından gelen) sesler için, kaynak sunucudaki bir URL ilke dosyası, yüklenen ortamdaki verilere erişim izni verir.

Daha fazla bilgi için bkz. "Yazar (geliştirici) denetimleri" sayfa 1066 ve "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062.

Ses verilerine RTMP akışlarından erişmek için, sunucuya erişime izin vermelisiniz. Flash Media Server'daki belli dizinlere erişime izin vermek için Sunucu-Tarafı ActionScript `Client.audioSampleAccess` özelliğini kullanın. Daha fazla bilgi için bkz. Sunucu Tarafı ActionScript Dil Başvurusu.

Video verilerine erişme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Geçerli video karesinin piksel verilerini yakalamak için `BitmapData.draw()` veya `BitmapData.drawWithQuality()` yöntemini kullanabilirsiniz. (`drawWithQuality` yöntemi Flash Player 11.3 ve üstü ile AIR 3.3 ve üstünde kullanılabilir.)

İki farklı video türü vardır:

- Flash Media Server'dan RTMP üzerinden akan video
- Bir FLV veya F4V dosyasından yüklenen aşamalı video

RTMP akışlarından çalışma zamanı grafiklerini ayıklamak amacıyla draw yönetmelerini kullanmak için, sunucuya erişim izni vermelisiniz. Flash Media Server'daki belli dizinlere erişime izin vermek için Sunucu-Tarafı ActionScript `Client.audioSampleAccess` özelliğini kullanın. Daha fazla bilgi için bkz. Sunucu Tarafı ActionScript Dil Başvurusu.

`source` parametresi olarak aşamalı video ile bir draw yöntemini çağırdığınızda, yöntem çağıranının FLV dosyasıyla aynı korumalı alandan olması veya FLV dosyasının sunucusunun, çağıran SWF dosyasının etki alanına izin veren bir ilke dosyasına sahip olması gerekir. NetStream nesnesinin `checkPolicyFile` özelliğini `true` değerine ayarlayarak ilke dosyasının indirilmesini isteyebilirsiniz.

Veri yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player ve AIR içeriği, sunucularla veri alışverişi yapabilir. Yüklenen bilgiler medya olarak görüntülenmek yerine program nesneleri olarak görüntülendiğinden, veri yükleme, medya yüklemeden farklı bir işlem türüdür. Genellikle, içerik oluşturulduğu etki alanının aynısından veri yükleyebilir. Ancak, içerik genellikle başka etki alanlarından veri yüklemek için ilke dosyalarını gerektirir (bkz. "Web sitesi denetimleri (ilke dosyaları)" sayfa 1062).

Not: *AIR uygulaması sanal alanında çalışan içerik hiçbir zaman uzak bir etki alanından sunulmaz (geliştirici, uygulama sanal alanına bilerek uzak içerik aktarmadığı sürece), dolayısıyla ilke dosyalarının korunduğu türde saldırılara katılamaz. Uygulama sanal alanındaki AIR içeriğinin ilke dosyalarıyla veri yüklemesi sınırlandırılmaz. Ancak, diğer sanal alanlardaki AIR içeriği burada anlatılan sınırlamalara tabidir.*

URLLoader ve URLStream öğelerini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

XML dosyası veya metin dosyası gibi verileri yükleyebilirsiniz. URLLoader ve URLStream sınıflarının `load()` yöntemleri, URL ilke dosyası izinleriyle yönetilir.

Yöntemi çağıran kodun etki alanından başka bir etki alanından içerik yüklemek için `load()` yöntemini kullanırsanız, çalışma zamanı, yüklenen varlıkların sunucusunda bir URL ilke dosyasını kontrol eder. Bir ilke dosyası varsa ve yükleyen içeriğin etki alanına erişim izni veriyorsa, verileri yükleyebilirsiniz.

Soketlere bağlanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Çalışma zamanı varsayılan olarak bağlantı noktası 843'ten sunulan bir soket ilke dosyasını arar. URL ilke dosyaları gibi, bu dosya da *ana ilke dosyası* olarak adlandırılır.

İlke dosyaları ilk defa Flash Player 6 ile sunulduğunda, soket ilke dosyaları için destek yoktu. Soket sunucularına bağlantılar, soket sunucusuyla aynı ana bilgisayarın 80 bağlantı noktasında bir HTTP sunucusundaki varsayılan konumda bulunan ilke dosyası ile yetkilendirilirdi. Flash Player 9 halen bu yeteneği desteklerken Flash Player 10 desteklemez. Flash Player 10'da, yalnızca soket ilke dosyaları soket bağlantılarını yetkilendirebilir.

URL ilke dosyaları gibi, soket ilke dosyaları da hangi bağlantı noktalarının ilke dosyası olarak hizmet verebileceğini belirten meta-ilke ifadesini destekler. Ancak soket ilke dosyaları için varsayılan meta-ilke "yalnızca ana" yerine "tümü"dür. Başka bir deyişle, ana ilke dosyası daha kısıtlayıcı bir ayar belirtmediği sürece, Flash Player uygulaması ana bilgisayardaki herhangi bir soketin soket ilke dosyası sunabildiğini varsayar.

Bağlandığınız soket, SWF dosyasıyla aynı etki alanında olsa da, soket ve XML soket bağlantılarına erişim varsayılan olarak devre dışı bırakılmıştır. Şu konumlardan herhangi birinden soket ilke dosyası sunarak soket düzeyinde erişime izin verebilirsiniz:

- Bağlantı noktası 843 (ana ilke dosyasının konumu)
- Ana soket bağlantısıyla aynı bağlantı noktası
- Ana soket bağlantısından farklı bir bağlantı noktası

Flash Player varsayılan olarak, bağlantı noktası 843'te ve ana soket bağlantısıyla aynı bağlantı noktasında bir soket ilke dosyasını arar. Farklı bir bağlantı noktasından soket ilke dosyası sunmak istiyorsanız, SWF dosyasının `Security.loadPolicyFile()` öğesini çağırması gerekir.

Soket ilke dosyası, URL ilke dosyasıyla aynı sözdizimine sahiptir, tek farkı, erişim izni verdiği bağlantı noktalarını da belirtmek zorunda olmasıdır. Soket ilke dosyası 1024'ten düşük bir bağlantı noktası numarasından sunulduğunda, herhangi bir bağlantı noktasına erişim izni verebilir; ilke dosyası bağlantı noktası 1024 veya daha yukarısından geldiğinde, yalnızca bağlantı noktası 1024 ve yukarısına erişim izni verebilir. İzin verilen bağlantı noktaları, `<allow-access-from>` etiketinde `to-ports` niteliğinde belirtilir. Tek bağlantı noktası numaraları, bağlantı noktası aralıkları ve joker karakterler kabul edilen değerlerdir.

Aşağıda, soket ilke dosyasına bir örnek verilmiştir:

```
<?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>
```

Bağlantı noktası 843'ten veya ana soket bağlantısıyla aynı bağlantı noktasından bir soket ilke dosyasını almak için, `Socket.connect()` veya `XMLSocket.connect()` yöntemini çağırın. Flash Player öncelikle bağlantı noktası 843'te bir ana ilke dosyasını kontrol eder. Bir ilke dosyası bulursa, dosyanın, hedef bağlantı noktasında soket ilke dosyalarını yasaklayan bir meta-ilke deyimi içerip içermediğini kontrol eder. Erişim yasaklanmamışsa, Flash Player ilk olarak ana ilke dosyasında uygun `allow-access-from` deyimini arar. Bu deyimi bulamazsa, daha sonra ana soket bağlantısıyla aynı bağlantı noktasında bir soket ilke dosyasını arar.

Farklı bir konumdan soket ilke dosyasını almak için, aşağıdaki gibi, ilk olarak özel `"xmlsocket"` sözdizimiyle `Security.loadPolicyFile()` yöntemini çağırın:

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

`Socket.connect()` veya `XMLSocket.connect()` yöntemini çağırmadan önce `Security.loadPolicyFile()` yöntemini çağırın. Flash Player daha sonra ana bağlantınıza izin verip vermemeye karar vermeden önce ilke dosyası isteğinizi yerine getirinceye kadar bekler. Ancak ana ilke dosyası, hedef konumun ilke dosyaları sunamayacağını belirtirse, söz konusu konumda bir ilke dosyası olsa da `loadPolicyFile()` öğesine yapılan çağrı herhangi bir etki yaratmaz.

Bir soket sunucusu uyguluyorsanız ve bir soket ilke dosyası sağlamanız gerekiyorsa, ana bağlantıları kabul eden aynı bağlantı noktasını mı yoksa farklı bir bağlantı noktasını mı kullanarak ilke dosyasını sağlayacağınıza karar verin. Her iki durumda da, bir yanıt göndermeden önce sunucunuzun istemcinizden birinci iletimi beklemesi gerekir.

Flash Player bir ilke dosyası istediğinde, bağlantı kurulur kurulmaz her zaman şu dizeyi iletir:

```
<policy-file-request/>
```

Sunucu bu dizeyi aldıktan sonra ilke dosyasını iletebilir. Flash Player uygulamasından gelen istek her zaman bir null bayt tarafından sonlandırılır ve sunucudan gelen yanıtın da bir null bayt tarafından sonlandırılması gerekir.

Hem ilke dosyası isteği hem de ana bağlantı için aynı bağlantıyı yeniden kullanmayı beklemeyin; ilke dosyasını ilettikten sonra bağlantıyı kapatın. Aksi takdirde, Flash Player uygulaması, ana bağlantıyı kurmak üzere yeniden bağlanmadan önce ilke dosyası bağlantısını kapatır.

Verileri koruma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Verileri internette dolaşırken başkalarının görmesinden ve değişiklik yapılmasından korumak için, verilerin çıkış yeri olan sunucuda Aktarım Katmanı Güvenliği (TLS) veya Soket Katmanı Güvenliği'ni (SSL) kullanabilirsiniz. Ardından HTTPS protokolünü kullanarak sunucuya bağlanabilirsiniz.

AIR 2 veya üstü için oluşturulan uygulamalarda, TCP soket iletişimlerini de koruyabilirsiniz. SecureSocket sınıfı, TLS sürüm 1 veya SSL sürüm 4 kullanan bir soket sunucusuyla soket bağlantısı başlatmanıza izin verir.

Veri gönderme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Kod bir sunucuya ya da kaynağa veri gönderdiğinde veri gönderimi meydana gelir. Veri göndermeye bir ağ etki alanından içerik için her zaman izin verilir. Yerel SWF dosyası ise yalnızca yerel güvenilen veya ağ iletişimiyle yerel ya da AIR uygulaması sanal alanında bulunuyorsa ağ adreslerine veri gönderebilir. Daha fazla bilgi için, bkz. "Yerel sanal alanlar" sayfa 1055.

Bir URL'ye veri göndermek için `flash.net.sendToURL()` işlevini kullanabilirsiniz. Başka yöntemler de URL'lere istekler gönderebilir. Bunlar arasında, `Loader.load()` ve `Sound.load()` gibi yükleme yöntemleri ve `URLLoader.load()` ve `URLStream.load()` veri yükleme yöntemleri yer alır.

Dosya yükleme ve indirme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

`FileReference.upload()` yöntemi, bir kullanıcı tarafından seçilen dosyanın uzak sunucuya yüklenmesini başlatır. `FileReference.upload()` yöntemini çağırmadan önce `FileReference.browse()` veya `FileReferenceList.browse()` yöntemini çağırmanız gerekir.

`FileReference.browse()` veya `FileReferenceList.browse()` yöntemini başlatan kod yalnızca bir fare olayına veya klavye olayına yanıt olarak çağrılabilir. Bu başka durumlarda çağrılırsa, Flash Player 10 ve sonrası bir istisna atar. Ancak, AIR uygulaması sanal alanından bu yöntemleri çağırmak için kullanıcı tarafından başlatılan bir olay gerekmez.

`FileReference.download()` yönteminin çağrılması, kullanıcının uzak bir sunucudan dosya indirebildiği bir iletişim kutusunu açar.

Not: *Sunucunuz kullanıcı kimlik doğrulaması gerektiriyorsa, yalnızca tarayıcıda çalıştırılan—başka bir deyişle, tarayıcı eklentisi veya ActiveX denetimi kullanan—SWF dosyaları, yalnızca indirmeler için kullanıcıdan bir kullanıcı adı ve şifre istemek üzere bir iletişim kutusu sağlar. Flash Player uygulaması, kullanıcı kimlik doğrulaması gerektiren bir sunucuya yükleme yapılmasına izin vermez.*

Çağıran SWF dosyası dosya sistemiyle yerel sanal alandaysa, yüklemelere ve indirmelere izin verilmez.

Varsayılan olarak SWF dosyası, kendi sunucusu dışındaki bir sunucuya yükleme veya o sunucudan indirme işlemini başlatamaz. Sunucu, çağıran SWF dosyasının etki alanına izin veren bir ilke dosyası sağlıyorsa, SWF dosyası farklı bir sunucuya yükleme veya farklı bir sunucudan indirme işlemi yapabilir.

Güvenlik etki alanına içe aktarılan SWF dosyalarından gömülü içerik yükleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir SWF dosyasını yüklediğinizde, dosyayı yüklemek için kullanılan Loader nesnesinin `load()` yönteminin `context` parametresini ayarlayabilirsiniz. Bu parametre bir LoaderContext nesnesini alır. Bu LoaderContext nesnesinin `securityDomain` özelliğini `Security.currentDomain` olarak ayarlarsınız Flash Player uygulaması yüklenen SWF dosyasının sunucusunda bir URL ilke dosyasını kontrol eder. Bir ilke dosyası varsa ve yükleyen SWF dosyasının etki alanına erişim izni veriyorsa, SWF dosyasını içe aktarılan ortam olarak yükleyebilirsiniz. Bu şekilde, yükleyen dosya, SWF dosyasının kütüphanesindeki nesnelere erişim elde edebilir.

SWF dosyasının, farklı bir güvenlik sanal alanından yüklenmiş SWF dosyalarındaki sınıflara erişmesinin alternatif bir yolu, çağıran SWF dosyasının etki alanına erişim izni vermek için, yüklenen SWF dosyasının `Security.allowDomain()` yöntemini çağırmasını sağlamaktır. Yüklenen SWF dosyasının ana sınıfının yapıcı yöntemine, `Security.allowDomain()` yöntemine yapılan bir çağrıyı ekleyebilir ve böylece yükleyen SWF dosyasının, Loader nesnesinin `contentLoaderInfo` özelliği tarafından gönderilen `init` olayını yanıtlamak için bir olay dinleyicisi eklemesini sağlayabilirsiniz. Bu olay gönderildiğinde, yüklenen SWF dosyası, yapıcı yönteminde `Security.allowDomain()` yöntemini çağırmıştır ve yüklenen SWF dosyasındaki sınıflar, yükleyen SWF dosyası için kullanılabilir durumdadır. Yüklenen SWF dosyası, `Loader.contentLoaderInfo.applicationDomain.getDefinition()` veya Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames() öğesini çağırarak yüklü SWF dosyasından sınıfları alabilir (Flash Player 11.3 ve üstü; AIR 3.3 ve üstü).

Eski içerikle çalışma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player 6'da, belirli Flash Player ayarları için kullanılan etki alanı, SWF dosyasının etki alanının son kısmını esas alır. Bu ayarlar arasında, kamera ve mikrofon izinleri, depolama kotaları ve kalıcı paylaşılan nesnelerin depolanmasına yönelik ayarlar yer alır.

SWF dosyasının etki alanı ikiden çok parça içeriyorsa (örn. www.example.com), etki alanının birinci parçası (www) kaldırılır ve etki alanının kalan kısmı kullanılır. Bu nedenle Flash Player 6'da, hem www.example.com hem de store.example.com bu ayarlar için etki alanı olarak example.com'u kullanır. Aynı şekilde, hem www.example.co.uk hem de store.example.co.uk bu ayarlar için etki alanı olarak example.co.uk'yi kullanır. Bu da, example1.co.uk ve example2.co.uk gibi ilgisiz etki alanlarından gelen SWF dosyalarının aynı paylaşılan nesneye erişim sahibi olması durumunda sorunlara neden olabilir.

Flash Player 7 ve sonrasında, oynatıcı ayarları varsayılan olarak SWF dosyasının gerçek etki alanına göre seçilir. Örneğin, www.example.com konumundan gelen bir SWF dosyası, www.example.com'un oynatıcı ayarlarını kullanırken, store.example.com konumundan gelen bir SWF dosyası da store.example.com'un ayrı oynatıcı ayarlarını kullanır.

ActionScript 3.0 kullanılarak yazılmış bir SWF dosyasında, `Security.exactSettingstrue` değerine (varsayılan) ayarlandığında, Flash Player uygulaması oynatıcı ayarları için etki alanının tamamını kullanır. `false` değerine ayarlandığında, Flash Player uygulaması Flash Player 6'da kullanılan etki alanı ayarlarını kullanır. `exactSettings` öğesinin varsayılan değerini değiştirirseniz, Flash Player uygulamasının oynatıcı ayarlarını seçmesini gerektiren olaylardan (örn. kamera veya mikrofon kullanma ya da bir kalıcı paylaşılan nesne alma) önce bunu yapmanız gerekir.

6 SWF sürüm dosya yayınladıysanız ve bundan kalıcı paylaşılan nesneler oluşturduysanız, ActionScript 3.0 kullanan bir SWF'den bu kalıcı paylaşılan nesneleri almak için, `SharedObject.getLocal()` öğesini çağırmadan önce `Security.exactSettings` öğesini `false` değerine ayarlamanız gerekir.

LocalConnection izinlerini ayarlama

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

LocalConnection sınıfı, bir Flash Player veya AIR uygulamasından diğerine mesajlar göndermenize izin verir. LocalConnection nesneleri yalnızca aynı bilgisayarda çalışan Flash Player veya AIR içeriği arasında iletişim kurabilir, ancak bunlar farklı uygulamalarda çalışabilir—örneğin, bir tarayıcıda çalışan SWF dosyası, bir projektörde çalışan SWF dosyası ve bir AIR uygulaması LocalConnection sınıfını kullanarak iletişim kurabilir.

Her LocalConnection iletişimi için bir gönderen, bir de dinleyici mevcuttur. Varsayılan olarak Flash Player uygulaması, aynı etki alanındaki kodlar arasında LocalConnection iletişimine olanak tanır. Farklı sanal alanlardaki kodlar için, dinleyicinin `LocalConnection.allowDomain()` yöntemini kullanarak gönderene izin vermesi gerekir. `LocalConnection.allowDomain()` yöntemine argüman olarak ilettiğiniz dize, şunlardan herhangi birini içerebilir: tam etki alanı adları, IP adresleri ve `*` joker karakteri.

`allowDomain()` yöntemi, ActionScript 1.0 ve 2.0'daki halinden değiştirilmiştir. Bu önceki sürümlerde, `allowDomain()` uyguladığınız bir geri çağrı yöntemiydi. ActionScript 3.0'da `allowDomain()` öğesi, çağırdığınız LocalConnection sınıfının yerleşik bir yöntemidir. Bu değişiklik sayesinde `allowDomain()`, `Security.allowDomain()` öğesine çok benzer şekilde çalışır.

SWF dosyası, etki alanını belirlemek için LocalConnection sınıfının `domain` özelliğini kullanabilir.

Giden URL erişimini denetleme

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Aşağıdaki API'lerin kullanımıyla giden komut dosyası oluşturma ve URL erişimi (HTTP URL'leri, mailto:, vb. kullanılarak) gerçekleştirilir:

- `flash.system.fscommand()` işlevi
- `ExternalInterface.call()` yöntemi
- `flash.net.navigateToURL()` işlevi

Yerel dosya sisteminden yüklenen içerik için, bu yöntemlere yapılan çağrılar ancak kod ve içerdiği web sayfası (varsa) yerel güvenilir veya AIR uygulaması güvenlik sanal alanları içerisindeyken başarılı olur. İçerik, ağ iletişimiyle yerel veya dosya sistemiyle yerel sanal alandaysa bu yöntemlere çağrılar başarısız olur.

Yerel olarak çalıştırılmayan kodlar için bu API'lerin tümü, aşağıda açıklanan AllowScriptAccess parametresinin değerine bağlı olarak, gömülü oldukları web sayfasıyla iletişim kurabilir. `flash.net.navigateToURL()` işlevi, yalnızca SWF dosyasının gömülü olduğu sayfayla değil, ek olarak herhangi bir açık tarayıcı penceresi veya karesiyle de iletişim kurma yeteneğine sahiptir. Bu işlevsellik hakkında daha fazla bilgi almak için, bkz. "navigateToURL() işlevini kullanma" sayfa 1085.

Bir SWF dosyasını yükleyen HTML kodundaki `AllowScriptAccess` parametresi, SWF dosyasından giden URL erişimini gerçekleştirme yeteneğini denetler. PARAM veya EMBED etiketinin içinde bu parametreyi ayarlayın. `AllowScriptAccess` için herhangi bir parametre ayarlanmazsa, SWF dosyası ve HTML sayfası yalnızca her ikisi de aynı etki alanından geliyorsa iletişim kurabilir.

`AllowScriptAccess` parametresi üç olası değerden yalnızca birine sahip olabilir: `"always"`, `"sameDomain"` veya `"never"`.

- `AllowScriptAccess"always"` olduğunda, SWF dosyası, HTML sayfasından farklı bir etki alanından gelse de, gömülü olduğu HTML sayfasıyla iletişim kurabilir.
- `AllowScriptAccess"sameDomain"` olduğunda, SWF dosyası yalnızca HTML sayfasıyla aynı etki alanından geliyorsa, gömülü olduğu HTML sayfasıyla iletişim kurabilir. Bu değer `AllowScriptAccess` için varsayılan değer değildir. Bir etki alanında barındırılan SWF dosyasının, başka bir etki alanından gelen HTML sayfasındaki bir komut dosyasına erişmesini önlemek için bu ayarı kullanın veya `AllowScriptAccess` için bir değer ayarlamayın.
- `AllowScriptAccess"never"` olduğunda, SWF dosyası herhangi bir HTML sayfasıyla iletişim kuramaz. Adobe Flash CS4 Professional'ın piyasaya sürülmesinden beri bu değerin kullanımına itiraz edildi. Kendi etki alanınızdan güvenilmeyen SWF dosyaları sunmuyorsanız bu önerilmez ve gerekli de değildir. Güvenilmeyen SWF dosyaları sunmanız gerekmiyorsa, Adobe ayrı bir alt etki alanı oluşturup tüm güvenilmeyen içeriği oraya yerleştirmenizi önerir.

Burada, giden bir URL'nin farklı bir etki alanına erişmesine izin vermek için `AllowScriptAccess` etiketinin ayarlanmasına örnek verilmektedir:

```
<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>
```

navigateToURL() işlevini kullanma

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Yukarıda açıklanan `allowScriptAccess` parametresi tarafından belirtilen güvenlik ayarına ek olarak, `navigateToURL()` işlevi ikinci bir isteğe bağlı parametreye sahiptir - `target`. `target` parametresi, URL isteğinin gönderileceği bir HTML penceresinin veya karesinin adını belirtmek için kullanılabilir. Bu istekler için ek güvenlik kısıtlamaları uygulanır ve bu kısıtlamalar, `navigateToURL()` öğesinin komut dosyası oluşturma deyimi olarak mı yoksa komut dosyası oluşturmama deyimi olarak mı kullanıldığına bağlı olarak değişiklik gösterir.

`navigateToURL("javascript: alert('Hello from Flash Player.')")` gibi, komut dosyası oluşturma deyimleri için şu kurallar geçerlidir.

- SWF dosyası yerel olarak güvenilen bir dosyaysa, istek başarılı olur.
- Hedef, SWF dosyasının gömülü olduğu HTML sayfasıysa, yukarıda açıklanan `allowScriptAccess` kuralları geçerli olur.
- Hedef, SWF dosyasıyla aynı etki alanından yüklenmiş içeriği barındırıyorsa, istek başarılı olur.
- Hedef, SWF dosyasından farklı bir etki alanından yüklenmiş içeriği barındırıyorsa ve önceki iki koşul da karşılanmamışsa, istek başarısız olur.

Komut dosyası oluşturmama deyimleri için (örn. HTTP, HTTPS ve `mailto:`), şu koşulların tümü geçerli olursa istek başarısız olur:

- Hedef, `"_top"` veya `"_parent"` özel anahtar sözcüklerinden biriyse ve
- SWF dosyası, farklı bir etki alanından barındırılan bir web sitesindeyse ve
- SWF dosyası, `"always"` dışındaki bir `allowScriptAccess` değeriyle gömülüyse.

Daha fazla bilgi için

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Giden URL erişimi hakkında daha fazla bilgi için, *Adobe Flash Platformu için ActionScript 3.0 Başvurusu*'ndaki aşağıdaki girişlere bakın:

- flash.system.fscommand() işlevi
- ExternalInterface sınıfının call() yöntemi
- `flash.net.navigateToURL()` işlevi

Paylaşılan nesneler

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Flash Player, kullanıcının dosya sisteminde yerel olarak veya bir RTMP sunucusunda uzaktan SWF dosyası dışında olmayı sürdüren ActionScript nesneleri olan *paylaşılan nesneleri* kullanma yeteneği sağlar. Flash Player'daki diğer ortamlar gibi, paylaşılan nesneler de güvenlik sanal alanlarında bölümlenir. Ancak paylaşılan nesneler, etki alanı sınırları üzerinden erişilebilen kaynaklar olmadığından, paylaşılan nesneler için sanal alan modeli biraz daha farklıdır. Paylaşılan nesneler her zaman, SharedObject sınıfının yöntemlerini çağıran her SWF dosyasının etki alanına özgü olan paylaşılan bir nesne deposundan alınır. Paylaşılan nesne deposu genellikle SWF dosyasının etki alanından çok daha özeldir: varsayılan olarak her SWF dosyası, kaynak URL'sinin tamamına özel olan bir paylaşılan nesne deposu kullanır. Paylaşılan nesnelerle ilgili daha fazla bilgi için, bkz. “Paylaşılan nesneler” sayfa 709.

SWF dosyası, yalnızca URL'sinin bir kısmıyla ilişkilendirilmiş olan paylaşılan nesne deposunu kullanmak için `SharedObject.getLocal()` ve `SharedObject.getRemote()` yöntemlerinin `localPath` parametresini kullanabilir. Bu şekilde, SWF dosyası, başka URL'lerden gelen SWF dosyalarıyla paylaşıma izin verebilir. `localPath` parametresi olarak `'/'` öğesini iletirseniz, bu, kendi etki alanına özgü bir paylaşılan nesne deposunu belirtir.

Kullanıcılar Flash Player Ayarları iletişim kutusunu veya Ayarlar Yöneticisi'ni kullanarak paylaşılan nesne erişimini kısıtlayabilir. Varsayılan olarak, paylaşılan nesneler her etki alanı için maksimum 100 KB veriye kadar oluşturulabilir. Yönetici kullanıcılar ve kullanıcılar da dosya sistemine yazma yeteneğine kısıtlamalar uygulayabilir. Daha fazla bilgi için bkz. “Yönetimsel denetimler” sayfa 1060 ve “Kullanıcı denetimleri” sayfa 1061.

`SharedObject.getLocal()` yönteminin veya `SharedObject.getRemote()` yönteminin `secure` parametresi için `true` değerini belirterek paylaşılan nesnenin güvenli olduğunu belirtebilirsiniz. `secure` parametresiyle ilgili olarak şunları unutmayın

- Bu parametre `true` olarak ayarlanırsa, Flash Player yeni bir güvenli paylaşılan nesne oluşturur ve varolan bir güvenli paylaşılan nesneye yapılan başvuruyu alır. Bu güvenli paylaşılan nesne, yalnızca `secure` parametresi `true` olarak ayarlanmış `SharedObject.getLocal()` öğesini çağıran HTTPS üzerinden teslim edilen SWF dosyaları tarafından okunabilir veya yazılabilir.

- Bu parametre `false` olarak ayarlanmışsa, Flash Player yeni bir paylaşılan nesne oluşturur veya HTTPS olmayan bağlantılar üzerinden teslim edilen SWF dosyaları tarafından okunabilen veya yazılabilen mevcut paylaşılan nesne için bir başvuru alır.

Çağıran SWF bir HTTPS URL'sinden gelmiyorsa, `SharedObject.getLocal()` yönteminin veya `SharedObject.getRemote()` yönteminin `secure` parametresi için `true` değerinin belirtilmesi SecurityError istisnasının atılmasına neden olur.

Paylaşılan nesne deposu tercihi, SWF dosyasının kaynak URL'sini esas alır. SWF dosyasının basit bir URL'den gelmediği şu iki koşulda da bu durum geçerlidir: içe aktarma yükleme ve dinamik yükleme İçe aktarma yükleme, `LoaderContext.securityDomain` özelliği `SecurityDomain.currentDomain` olarak ayarlıyken bir SWF dosyası yüklemeniz durumunu ifade eder. Bu durumda, yüklenen SWF dosyası, yükleyen SWF dosyasının etki alanıyla başlayan ve sonra da gerçek kaynak URL'sini belirten sahte bir URL'ye sahiptir. Dinamik yükleme, `Loader.loadBytes()` yöntemi kullanılarak SWF dosyasının yüklenmesini ifade eder. Bu durumda, yüklenen SWF dosyası, yükleyen SWF dosyasının tam URL'sini ve ardından bir tam sayı kimliğini içeren sahte bir URL'ye sahiptir. Hem içe aktarma yükleme hem de dinamik yükleme durumunda da, `LoaderInfo.url` özelliği kullanılarak SWF dosyasının sahte URL'si incelenebilir. Paylaşılan nesne deposu seçilmesi amacıyla, sahte URL, tıpkı gerçek bir URL gibi değerlendirilir. Sahte URL'nin bir kısmını veya tamamını kullanan bir paylaşılan nesne `localPath` parametresini belirtebilirsiniz.

Kullanıcılar ve yöneticiler, *üçüncü taraf paylaşılan nesnelerin* kullanımını devre dışı bırakmayı seçebilir. Bu, bir SWF dosyasının kaynak URL'si, tarayıcının adres çubuğunda gösterilen URL'den farklı bir etki alanından geldiğinde, web tarayıcısında çalıştırılan SWF dosyası tarafından, paylaşılan nesnelerin kullanılmasıdır. Kullanıcılar ve yöneticiler, gizlilik nedenleriyle etki alanları arası izlemeyi önlemek amacıyla üçüncü taraf paylaşılan nesne kullanımını devre dışı bırakmayı seçebilir. Bu kısıtlamayı önlemek için, paylaşılan nesne kullanan SWF dosyalarının, yalnızca tarayıcının adres çubuğunda gösterilen aynı etki alanından SWF dosyasının gelmesini sağlayan HTML sayfası yapıları içinde yüklendiğinden emin olmak isteyebilirsiniz. Üçüncü taraf bir SWF dosyasından gelen paylaşılan nesneleri kullanmayı denediğinizde üçüncü taraf paylaşılan nesne kullanımı devre dışı bırakılmışsa, `SharedObject.getLocal()` ve `SharedObject.getRemote()` yöntemleri `null` değerini döndürür. Daha fazla bilgi için, bkz. www.adobe.com/products/flashplayer/articles/thirdpartylso.

Kamera, mikrofon, pano, fare ve klavye erişimi

Flash Player 9 ve üstü, Adobe AIR 1.0 ve üstü

Bir SWF dosyası, `Camera.get()` veya `Microphone.get()` yöntemlerini kullanarak kullanıcının kamerasına ya da mikrofonuna erişmeyi denediğinde, Flash Player uygulaması, kullanıcının kamerasına ve mikrofonuna erişilmesine izin verebildiği veya erişilmesini reddedebildiği bir Gizlilik iletişim kutusunu görüntüler. Kullanıcı ve yönetici kullanıcı, mms.cfg dosyasındaki denetimler, Ayarlar UI'si ve Ayarlar Yöneticisi aracılığıyla site çapında veya genel olarak kamera erişimini devre dışı bırakabilir (bkz. "Yönetimsel denetimler" sayfa 1060 ve "Kullanıcı denetimleri" sayfa 1061). Kullanıcı kısıtlamaları olduğunda, `Camera.get()` ve `Microphone.get()` yöntemlerinin her biri `null` değerini döndürür. Kamera veya mikrofonun yönetici tarafından yasaklanmış mı (`true`) yoksa kamera veya mikrofona izin mi verilmiş olduğunu (`false`) belirlemek için `Capabilities.avHardwareDisable` özelliğini kullanabilirsiniz.

`System.setClipboard()` yöntemi, bir SWF dosyasının panodaki içerikleri, karakterlerden oluşan bir düz metin dizesiyle değiştirmesine olanak sağlar. Bu herhangi bir güvenlik riski oluşturmaz. Şifre ve diğer hassas bilgilerin kesilip panolara kopyalanmasıyla ortaya çıkan riskten korunmak için ilgili bir `getClipboard()` yöntemi yoktur.

Flash Player'da çalışan bir uygulama yalnızca odağı içinde gerçekleşen klavye ve fare olaylarını izleyebilir. Flash Player'da çalışan içerik, başka bir uygulamada klavye veya fare olaylarını algılayamaz.

AIR güvenliği

Adobe AIR 1.0 ve üstü

AIR güvenlik temelleri

Adobe AIR 1.0 ve üstü

AIR uygulamaları yerel uygulamalarla aynı güvenlik kısıtlamaları ile çalışır. Genellikle, yerel uygulamalar gibi AIR uygulamalarının da dosya okuma ve yazma, uygulamalara başlama, ekrana çizim yapma ve ağ ile iletişime geçme gibi işletim sistemi özelliklerine genel erişimi vardır. Yerel uygulamalarda geçerli olan kullanıcıya özgü ayrıcalıklar gibi işletim sistemi kısıtlamaları, aynı şekilde AIR uygulamalarında da geçerlidir.

Adobe® AIR® güvenlik modeli, Adobe® Flash® Player güvenlik modelinin evrilmiş bir hali olsa da, güvenlik sözleşmesi tarayıcıdaki içeriğe uygulanan güvenlik sözleşmesinden farklıdır. Bu sözleşme, geliştiricilere tarayıcı tabanlı uygulamalar için alışılmadık özgürlükler ile zengin deneyimler sağlayan kapsamlı işlevlerin güvenli yollarını sunar.

Çalışma zamanının bellek yönetimi sağlayabilmesi için, AIR uygulamaları, derlenmiş bayt kodu (SWF içeriği) veya yorumlanmış komut dosyası (JavaScript, HTML) kullanılarak yazılır. Bu, AIR uygulamalarının arabellek taşması ve bellek bozulması gibi bellek yönetimiyle ilişkili güvenlik açıklarından etkilenme olasılığını en aza indirir. Bunlar, yerel kodda yazılan masaüstü uygulamalarını etkileyen en yaygın güvenlik açıklarından bazılarıdır.

Yükleme ve güncellemeler

Adobe AIR 1.0 ve üstü

AIR uygulamaları `air` uzantısını kullanan AIR yükleyici dosyaları veya yerel platformun dosya biçimini ve uzantısını kullanan yerel yükleyiciler aracılığıyla dağıtılır. Örneğin, Windows'un yerel yükleyici biçimi EXE dosyasıdır ve Android'in yerel biçimi APK dosyasıdır.

Adobe AIR yüklendiğinde ve bir AIR yükleyicisi açıldığında yükleme işlemini AIR çalışma zamanı yönetir. Yerel bir yükleyici kullanıldığında yükleme işlemini işletim sistemi yönetir.

Not: *Geliştiriciler AIR dosyası biçimi kullanırken bir sürüm, uygulama adı ve yayıncı kaynağı belirtebilir, ancak ilk uygulama yükleme iş akışının kendisi değiştirilemez. Tüm AIR uygulamaları çalışma zamanı tarafından yönetilen güvenli, düzene koyulmuş ve tutarlı bir yükleme yordamını paylaştığından, bu kısıtlama kullanıcılar için avantajlıdır. Uygulama özelleştirmesi gerekliyse, uygulama ilk yürütüldüğünde sağlanabilir.*

Çalışma zamanı yükleme konumu

Adobe AIR 1.0 ve üstü

SWF dosyalarının Flash Player tarayıcı eklentisinin yüklenmesini gerektirdiği gibi, AIR dosyası biçimini kullanan AIR uygulamaları da kullanıcının bilgisayarında önce çalışma zamanının yüklenmesini gerektirir.

Masaüstü bilgisayarlarda çalışma zamanı şu konuma yüklenir:

- Mac OS: `/Library/Frameworks/`
- Windows: `C:\Program Files\Common Files\Adobe AIR`
- Linux: `/opt/Adobe AIR/`

Mac OS'de uygulamanın güncellenmiş bir sürümünü yüklemek için, kullanıcının uygulama dizinine yüklemek için yeterli sistem ayrıcalıklarına sahip olması gerekir. Windows ve Linux'ta, kullanıcı yönetici ayrıcalıklarına sahip olmalıdır.

Not: *iOS'de AIR çalışma zamanı ayrı olarak yüklenmez, her AIR uygulaması bağımsız bir uygulamadır.*

Çalışma zamanı iki şekilde yüklenebilir: kesintisiz yükleme özelliğini kullanarak (doğrudan bir web tarayıcısından yükleme) veya elle yükleme yoluyla. Yerel yükleyiciler olarak paketlenen AIR uygulamaları da AIR çalışma zamanını kendi normal uygulama yükleme işlemlerinin bir parçası olarak yükleyebilir. (AIR çalışma zamanının bu şekilde dağıtılması Adobe ile yeniden dağıtım sözleşmesi gerektirir.)

Kesintisiz yükleme (çalışma zamanı ve uygulama)

Adobe AIR 1.0 ve üstü

Kesintisiz yükleme özelliği, geliştiricilere Adobe AIR'i henüz yüklememiş kullanıcılar için düzenlenmiş bir yükleme deneyimi sağlar. Kesintisiz yükleme yönteminde geliştirici, yüklemeye ilişkin uygulamayı sunan bir SWF dosyası oluşturur. Kullanıcı uygulamayı yüklemek için SWF dosyasını tıklattığında, SWF dosyası çalışma zamanını saptamaya çalışır. Çalışma zamanı saptanamazsa, yüklenir ve hemen geliştirici uygulamasına ilişkin yükleme işlemiyle etkinleştirilir.

Elle yükleme

Adobe AIR 1.0 ve üstü

Alternatif olarak, kullanıcı AIR dosyasını açmadan önce çalışma zamanını elle indirip yükleyebilir. Daha sonra geliştirici AIR dosyasını farklı şekillerde dağıtabilir (örneğin, e-posta veya bir web sitesindeki HTML linki yoluyla). AIR dosyası açıldığında, çalışma zamanı uygulama yüklemesini işlemeye başlar.

Uygulama yükleme akışı

Adobe AIR 1.0 ve üstü

AIR güvenlik modeli, kullanıcıların AIR uygulamasının yüklenip yüklenmeyeceğine karar vermelerine olanak tanır. AIR yükleme deneyimi, kullanıcıların güvenle ilgili bu kararı daha kolay vermelerini sağlayan yerel uygulama yükleme teknolojilerinin üzerinde çeşitli iyileştirmeler sağlar.

- Çalışma zamanı, AIR uygulaması bir web tarayıcısındaki linkten yüklendiğinde bile, tüm işletim sistemlerinde tutarlı bir yükleme deneyimi sağlar. Çoğu yerel uygulama yükleme deneyimi, güvenlik bilgisi sağlıyorsa bile, bunu tarayıcıya veya diğer uygulamalara bağlı olarak sağlar.
- AIR uygulaması yükleme deneyimi, uygulamanın kaynağını ve uygulamanın hangi ayrıcalıkları kullanabileceğine ilişkin bilgiyi tanımlar (kullanıcı yüklemenin ilerlemesine izin verirse).
- Çalışma zamanı, AIR uygulamasının yükleme işlemini yönetir. AIR uygulaması, çalışma zamanının kullandığı yükleme işlemini değiştiremez.

Genellikle, kullanıcılar güvenmedikleri veya doğrulanamayan bir kaynaktan gelen masaüstü uygulamalarını yüklememelidir. Yerel uygulamalara ilişkin güvenlik kanıtının zorluğu, diğer yüklenebilir uygulamalar için olduğu gibi, AIR uygulamaları için de aynı şekilde geçerlidir.

Uygulama hedefi

Adobe AIR 1.0 ve üstü

Yükleme dizini, aşağıdaki seçeneklerden biri kullanılarak ayarlanabilir:

1. Kullanıcı, yükleme sırasında hedefi özelleştirebilir. Uygulama, kullanıcının belirttiği yere yüklenir.
2. Kullanıcı yükleme hedefini değiştirmezse, uygulama çalışma zamanı tarafından belirlenen varsayılan yola yüklenir:
   - Mac OS: `~/Applications/`
   - Windows XP ve öncesi: `C:\Program Files\`
   - Windows Vista: `~/Apps/`
   - Linux: /opt/

   Geliştirici uygulama tanımlayıcı dosyasında bir `installFolder` ayarı belirlerse, uygulama bu dizinin bir alt yoluna yüklenir.

AIR dosya sistemi

Adobe AIR 1.0 ve üstü

AIR uygulamaları için yükleme işlemi, geliştiricinin AIR yükleyici dosyasına dahil ettiği tüm dosyaları kullanıcının yerel bilgisayarına kopyalar. Yüklenen uygulama şunlardan oluşur:

- Windows: AIR yükleyici dosyasında bulunan tüm dosyaları içeren bir dizin. Çalışma zamanı AIR uygulamasının yüklenmesi sırasında bir exe dosyası da oluşturur.
- Linux: AIR yükleyici dosyasında bulunan tüm dosyaları içeren bir dizin. Çalışma zamanı AIR uygulamasının yüklenmesi sırasında bir bin dosyası da oluşturur.
- Mac OS: AIR yükleyici dosyasının tüm içeriğini içeren bir `app` dosyası. Bu, Finder'daki "Paket İçeriğini Göster" seçeneği kullanılarak denetlenebilir. Çalışma zamanı, bu app dosyasını AIR uygulaması yüklemesinin bir parçası olarak oluşturur.

AIR uygulaması şu şekilde çalıştırılır:

- Windows: Yükleme klasöründeki .exe dosyasını veya bu dosyaya karşılık gelen bir kısayolu çalıştırarak (Başlangıç Menüsü veya masaüstündeki bir kısayol gibi).
- Linux: Yükleme klasöründeki .bin dosyasını başlatarak, uygulamayı Uygulamalar menüsünden seçerek veya bir takma addan ya da masaüstü kısayolundan çalıştırarak.
- Mac OS: .app dosyasını veya bunu işaret eden bir takma adı çalıştırarak.

Uygulama dosya sistemi, uygulama işleviyle ilişkili alt dizinler de içerir. Örneğin, şifrelenmiş yerel depoya yazılmış bilgi, uygulamanın uygulama kimliğine göre adlandırılan dizindeki bir alt dizine kaydedilir.

AIR uygulama depolaması

Adobe AIR 1.0 ve üstü

AIR uygulamaları, kullanıcının sabit sürücüsündeki herhangi bir konuma yazma ayrıcalığına sahiptir; ancak, geliştiriciler uygulamalarına ilişkin yerel depo için `app-storage:/` yolunu kullanmaya teşvik edilir. Uygulamadan `app-storage:/` konumuna yazılan dosyalar, kullanıcının işletim sistemine bağlı standart bir konumda konumlandırılır.

- Mac OS'de: Bir uygulamanın depolama dizini AIR sürümüne göre değişiklik gösterir:
  - **AIR 3.2 ve öncesi** - `<appData>/<appId>/Local Store/`: Burada `<appData>` kullanıcının "tercihler klasörüdür" ve genelde şu şekildedir: `/Users/<user>/Library/Preferences`
  - **AIR 3.3 ve üstü** - `<path>/Library/Application Support/<appID>/Local Store`: Burada `<path>/Users/<user>/Library/Containers/<bundle-id>/Data` (korumalı ortam) veya `/Users/<user>` (korumalı bir ortamın dışında çalıştırırken) şeklindedir.
- Windows'ta: `<appData>` kullanıcının CSIDL_APPDATA "Özel Klasör"ü, tipik olarak `C:\Documents and Settings\<user>\Application Data` olduğunda, uygulamanın depolama dizini `<appData>\<appId>\Local Store\` olur.
- Linux'ta: `<appData>`, `/home/<user>/.appdata` olduğunda `<appData>/<appID>/Local Store/` olur.

Uygulama depolama dizinine `air.File.applicationStorageDirectory` özelliği yoluyla erişebilirsiniz. İçindekilere File sınıfının `resolvePath()` yöntemini kullanarak erişebilirsiniz. Ayrıntılar için bkz. "Dosya sistemiyle çalışma" sayfa 659.

Adobe AIR'i güncelleme

Adobe AIR 1.0 ve üstü

Kullanıcı, çalışma zamanının güncellenmiş bir versiyonunu gerektiren bir AIR uygulaması yüklediğinde, çalışma zamanı otomatik olarak gerekli çalışma zamanı güncellemesini yükler.

Kullanıcı, çalışma zamanını güncellemek için bilgisayara ilişkin yönetici ayrıcalıklarına sahip olmalıdır.

AIR uygulamalarını güncelleme

Adobe AIR 1.0 ve üstü

Yazılım güncellemelerinin geliştirilmesi ve konuşlandırılması, yerel kod uygulamalarının karşılaştığı en büyük güvenlik sorunlarından biridir. AIR API, bu sorunu gidermek için bir mekanizma sağlar: başlatmadan sonra bir AIR dosyası için uzak konumu kontrol etmek üzere `Updater.update()` yöntemi çağrılabilir. Güncelleme gerekliyse, AIR dosyası indirilir, yüklenir ve uygulama yeniden başlatılır. Geliştiriciler bu sınıfı yalnızca yeni işlevler sağlamak için değil, olası güvenlik açıklarına yanıt vermek için de kullanabilirler.

Updater sınıfı yalnızca AIR dosyası olarak dağıtılmış uygulamaları güncellemek için kullanılabilir. Yerel uygulama olarak dağıtılan uygulamalar varsa yerel işletim sisteminin güncelleme araçlarını kullanmalıdır.

Not: *Geliştiriciler uygulama açıklayıcı dosyasının versionNumber özelliğini ayarlayarak uygulamanın sürümünü belirtebilirler.*

AIR uygulamasını kaldırma

Adobe AIR 1.0 ve üstü

AIR uygulamasının kaldırılması, uygulama dizinindeki tüm dosyaları kaldırır. Ancak, uygulamanın uygulama dizini dışına yazmış olabileceği tüm dosyaları kaldırmaz. AIR uygulamalarının kaldırılması, AIR uygulamasının uygulama dizininin dışındaki dosyalarda yaptığı değişiklikleri geri almaz.

Yöneticiler için Windows kayıt defteri ayarları

Adobe AIR 1.0 ve üstü

Windows'ta, yöneticiler bilgisayarı AIR uygulamasını ve çalışma zamanı güncellemelerini engelleyecek (veya bunlara izin verecek) şekilde yapılandırabilir. Bu ayarlar, Windows kayıt defterinde aşağıdaki anahtarın altında bulunur: HKLM\Yazılım\Politikalar\Adobe\AIR. Bunlar, aşağıdakileri içerir:

Kayıt defteri ayarı	Açıklama
AppInstallDisabled	AIR uygulamasının yüklenebileceğini ve kaldırılabileceğini belirtir. “İzinli” için 0, “izinli değil” için 1 olarak ayarlanır.
UntrustedAppInstallDisabled	Güvenli olmayan AIR uygulamalarının (güvenli bir sertifikası olmayan uygulamalar) kurulumuna izin verildiğini belirtir “İzinli” için 0, “izinli değil” için 1 olarak ayarlanır.
UpdateDisabled	Çalışma zamanının arka plan görevi veya açık yüklemenin bir parçası olarak güncellenebileceğini belirtir. “İzinli” için 0, “izinli değil” için 1 olarak ayarlanır.

Adobe AIR'de HTML güvenliği

Adobe AIR 1.0 ve üstü

Bu konu AIR HTML güvenlik yapısını ve HTML tabanlı uygulamalar oluşturmak ve HTML içeriğini SWF tabanlı uygulamalara güvenli bir şekilde entegre etmek için iframe'lerin, frame'lerin ve sanal alan köprüsünün nasıl kullanılacağını açıklar.

Çalışma zamanı, HTML ve JavaScript'teki olası güvenlik açıklarının üstesinden gelmek için kurallar uygular ve mekanizmalar sağlar. Uygulamanız temel olarak JavaScript'te yazılmış olduğunda veya HTML ve JavaScript içeriğini SWF tabanlı bir uygulamaya yüklediğinizde aynı kurallar uygulanır. Uygulama sanal alanında ve uygulama dışı güvenlik sanal alanındaki içeriğin farklı ayrıcalıkları vardır. Iframe'e veya frame'e içerik yüklenirken, çalışma zamanı, frame'deki veya iframe'deki içeriğin uygulama güvenlik sanal alanındaki içerikle güvenli biçimde iletişim kurmasını sağlayan güvenli bir *sanal alan köprüsü* mekanizması sağlar.

AIR SDK HTML içeriği oluşturmak için üç sınıf sağlar.

HTMLLoader sınıfı JavaScript kodu ve AIR API'leri arasında yakın entegrasyon sağlar.

StageWebView sınıfı HTML oluşturma sınıfıdır ve ana bilgisayar AIR uygulamasıyla sınırlı entegrasyonu vardır. StageWebView sınıfı tarafından yüklenen içerik asla uygulama güvenlik sanal alanına yerleştirilmez ve ana bilgisayar AIR uygulamasında veri veya çağrı işlevlerine erişemez. Masaüstü platformlarda StageWebView sınıfı Webkit'e dayalı olan ve HTMLLoader sınıfı tarafından kullanılan yerleşik AIR HTML motorunu kullanır. Mobil platformlarda StageWebView sınıfı işletim sistemi tarafından sağlanan HTML kontrolünü kullanır. Bu nedenle mobil platformlarda StageWebView sınıfı sistem web tarayıcısıyla aynı güvenlik kaygılarına ve açıklarına sahiptir.

TextField sınıfı HTML metninin dizelerini görüntüleyebilir. JavaScript yürütülemez ancak metin bağlantılar ve dışarıdan yüklenmiş görüntüler içerebilir.

Daha fazla bilgi için bkz. “Güvenlikle ilişkili JavaScript hatalarını önleme” sayfa 993.

HTML tabanlı uygulamanızın yapılandırılmasına genel bakış

Adobe AIR 1.0 ve üstü

Çerçeveler ve iframe'ler, AIR'de HTML içeriğinin düzenlenmesi için uygun bir yapı sağlar. Çerçeveler, hem veri kalıcılığını korumak hem de uzak içerikle güvenli bir şekilde çalışmak için yollar sağlar.

AIR'de HTML, normal, sayfa tabanlı düzenini koruduğundan, HTML içeriğinizin üst karesi farklı bir sayfaya "geçtiğinde" HTML ortamı tamamen yenilenir. Kareleri ve iframe'leri, tarayıcıda çalışan bir web uygulaması için yapacağınız gibi veri kalıcılığını korumak amacıyla kullanabilirsiniz. Üst karede ana uygulama nesnelerinizi tanımladığınızda, bu nesneler karenin yeni bir sayfaya geçmesine izin vermediğiniz sürece korunur. Uygulamanın geçici bölümlerini yüklemek ve görüntülemek için alt kareleri ve iframe'leri kullanın. (Veri kalıcılığını korumak için karelerin yanı sıra veya karelerin yerine kullanılabilecek çeşitli yollar mevcuttur. Bunlar tanımlama bilgilerini, yerel paylaşılan nesneleri, yerel dosya deposunu, şifrelenmiş dosya deposunu ve yerel veritabanı deposunu içerir.)

AIR'de HTML yürütülebilir kod ve veriler arasındaki normal, bulanık çizgisini koruduğundan, AIR HTML ortamının üst karesindeki içeriği uygulama sanal alanına yerleştirir. Sayfa `load` olayından sonra AIR, bir metin dizesini yürütülebilir bir nesneye çevirebilecek `eval()` gibi işlemleri sınırlar. Bu kısıtlama, uygulama uzak içerik yüklemediğinde bile uygulanır. HTML içeriğinin bu sınırlanmış işlemleri yürütebilmesi için, içeriği uygulama dışı bir sanal alana yerleştirmek üzere kareleri ve iframe'leri kullanmalısınız. (`eval()` işlevine dayanan bazı JavaScript uygulama çerçevelerinin kullanılması sırasında, içeriği sanal alanlı bir alt karede çalıştırmak gerekebilir.) Uygulama sanal alanındaki JavaScript kısıtlamalarının tam listesi için bkz. "Farklı sanal alanlardaki içerik için kod kısıtlamaları" sayfa 1094.

AIR'de HTML uzak, büyük olasılıkla güvenli olmayan içerik yükleme özelliğini koruduğundan, AIR, bir etki alanındaki içeriğin farklı bir etki alanındaki içerikle etkileşime girmesini engelleyen aynı kaynak politikasını uygular. Uygulama içeriği ve farklı etki alanındaki içerik arasında etkileşime izin vermek için, üst ve alt kare arasında arabirim olarak görev yapacak bir köprü kurabilirsiniz.

Üst-alt sanal alan ilişkisini kurma

Adobe AIR 1.0 ve üstü

AIR, HTML kare ve iframe öğelerine `sandboxRoot` ve `documentRoot` niteliklerini ekler. Bu nitelikler, uygulama içeriğini başka bir etki alanından gelmiş gibi ele almanızı sağlar:

Nitelik	Açıklama
sandboxRoot	Kare içeriğinin yerleştirileceği sanal alan ve etki alanını belirlemek için kullanılacak URL. URL şemasının kullanması gereken `file:`, `http:` veya `https:`.
documentRoot	Kare içeriğinin yükleneceği kaynak URL. URL şemasının kullanması gereken `file:`, `app:` veya `app-storage:`.

Aşağıdaki örnek, uygulamanın sanal alan alt dizinine yüklenen içeriği uzak sanal alanda ve www.example.com etki alanında çalışacak biçimde eşler:

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

Farklı sanal alanlarda veya etki alanlarında üst ve alt kareler arasında köprü kurma

Adobe AIR 1.0 ve üstü

AIR, her alt karenin `window` nesnesine `childSandboxBridge` ve `parentSandboxBridge` özelliklerini ekler. Bu özellikler, üst ve alt kare arasında arabirim olarak görev yapacak köprüleri tanımlamanızı sağlar. Her köprü bir yöne gider:

`childSandboxBridge` — `childSandboxBridge` özelliği alt karenin üst karedeki içeriğe bir arabirim göstermesini sağlar. Arabirim göstermek için `childSandbox` özelliğini alt karedeki bir işleme veya nesneye ayarlarsınız. Böylece nesneye veya işleve üst karedeki içerikten erişebilirsiniz. Aşağıdaki örnek, alt karede çalışan bir komut dosyasının bir işlev ve bir özellik içeren nesneyi üst öğesine nasıl gösterdiğini açıklar:

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

Bu alt içerik, `"child"` öğenin `id` öğesi atanmış bir iframe içindeyse, karenin `childSandboxBridge` özelliğini okuyarak üst içerikten arabirime erişebilirsiniz:

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

`parentSandboxBridge` — `parentSandboxBridge` özelliği üst karenin alt karedeki içeriğe bir arabirim göstermesini sağlar. Arabirim göstermek için alt karenin `parentSandbox` özelliğini üst karedeki bir işleve veya bir nesneye ayarlarsınız. Böylece nesneye veya işleve alt karedeki içerikten erişebilirsiniz. Aşağıdaki örnek, üst karede çalışan bir komut dosyasının saklama işlevi içeren bir nesneyi alt öğeye nasıl gösterdiğini açıklar:

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

Alt karedeki içerik, bu arabirimi kullanarak save.txt adlı dosyaya metin kaydedebilir. Ancak, dosya sistemine başka bir erişimi olmayacaktır. Genellikle, uygulama içeriği diğer sanal alanlara olanaklı en dar arabirimi göstermelidir. Alt içerik saklama işlevini aşağıdaki şekilde çağırabilir:

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

Alt içerik `parentSandboxBridge` nesnesinin bir özelliğini ayarlamaya çalıştığında, çalışma zamanı bir SecurityError istisnası atar. Üst içerik `childSandboxBridge` nesnesinin bir özelliğini ayarlamaya çalıştığında, çalışma zamanı bir SecurityError istisnası atar.

Farklı sanal alanlardaki içerik için kod kısıtlamaları

Adobe AIR 1.0 ve üstü

Bu “Adobe AIR'de HTML güvenliği” sayfa 1092 başlığında tartışıldığı gibi, çalışma zamanı HTML ve JavaScript'teki olası güvenlik açıklarının üstesinden gelebilmek için kurallar uygular ve mekanizmalar sağlar. Bu başlık, bu kısıtlamaları listeler. Kod, bu kısıtlanmış API'leri çağırmaya çalıştığında, çalışma zamanı “Uygulama güvenlik sanal alanındaki JavaScript kodu için Adobe AIR çalışma zamanı güvenlik ihlali” mesajını içeren bir hata verir.

Daha fazla bilgi için bkz. "Güvenlikle ilişkili JavaScript hatalarını önleme" sayfa 993.

JavaScript eval() işlevi ve benzer tekniklerin kullanımındaki kısıtlamalar

Adobe AIR 1.0 ve üstü

Uygulama güvenlik sanal alanındaki HTML içeriği için, kod yüklendiken sonra (`body` öğesinin `onload` olayı gönderildikten ve `onload` işleyici işlevinin yürütülmesi tamamlandıktan sonra) dizeleri dinamik olarak yürütülebilir koda dönüştürebilen API'lerin kullanımı için sınırlamalar mevcuttur. Bunun amacı, uygulamanın yanlışlıkla uygulama dışı kaynaklardan (güvenli olmayabilecek ağ etki alanları gibi) kod yerleştirmesini (ve yürütmesini) engellemektir.

Örneğin, uygulamanız DOM öğesinin innerHTML özelliğine yazmak için uzak bir kaynaktan dize verileri kullanıyorsa, dize güvenli olmayan işlemler gerçekleştirebilecek yürütülebilir (JavaScript) kod içerebilir. Ancak, içerik yüklenirken uzak dizelerin DOM'a eklenme riski yoktur.

JavaScript `eval()` işlevinin kullanımında bir kısıtlama vardır. Uygulama sanal alanındaki kod yüklendikten ve onload olay işleyicisindeki işlemeden sonra `eval()` işlevini sınırlı biçimde kullanabilirsiniz. Aşağıdaki kurallar, kodun uygulama güvenlik sanal alanından yüklenmesinden *sonra*`eval()` işlevinin kullanımı için geçerlidir:

- Değişmez içeren ifadeler izinli. Örneğin:

```
eval("null");
eval("3 + .14");
eval("'foo'");
```

- Aşağıdaki gibi nesne değişmezleri izinlidir:

```
{ prop1: val1, prop2: val2 }
```

- Aşağıdaki gibi nesne değişmezi ayarlayıcı/alıcıları *izinli değildir*:

```
{ get prop1() { ... }, set prop1(v) { ... } }
```

- Aşağıdaki gibi dizi değişmezleri izinlidir:

```
[ val1, val2, val3 ]
```

- Aşağıdaki gibi özellik okumaları içeren ifadeler *izinli değildir*:

```
a.b.c
```

- İşlev başlatma *izinli değil.*
- İşlev tanımları *izinli değil.*
- Özellik ayarlama *izinli değil.*
- İşlev değişmezleri *izinli değil.*

Ancak, kod yüklenirken, `onload` olayından önce ve `onload` olay işleyici işlevinin yürütülmesi sırasında bu kısıtlamalar uygulama güvenlik sanal alanındaki içeriğe uygulanmaz.

Örneğin, kod yüklendikten sonra istisna atan çalışma zamanı aşağıdaki kodla sonuçlanır:

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

Örneğin `eval()` işlevini çağırma sırasında yapılan gibi dinamik olarak oluşturulan kod, uygulama sanal alanında izinliyse bir güvenlik riski ortaya çıkarır. Örneğin, bir uygulama yanlışlıkla ağ etki alanından yüklenmiş bir dizeyi yürütebilir ve bu dize kötü amaçlı bir kod içerebilir. Örneğin, bu kod kullanıcının bilgisayarındaki dosyaları değiştiren veya silen bir kod olabilir. Veya yerel dosyanın içeriğini güvenilir olmayan bir ağ etki alanına bildiren bir kod olabilir.

Dinamik kod oluşturma yolları şunlardır:

- `eval()` işlevini çağırma.
- Uygulama dizininin dışındaki bir komut dosyasını yükleyen komut dosyası etiketleri eklemek için `innerHTML` özelliklerini veya DOM işlevlerini kullanma.
- Satır içi kod içeren komut dosyası etikeleri eklemek için `innerHTML` özelliklerini veya DOM işlevlerini kullanma (`src` niteliği yoluyla komut dosyası yüklemek yerine).
- Uygulama dizininin dışıda yer alan bir JavaScript dosyası yüklemek üzere `script` etiketleri için `src` niteliğini ayarlama.
- `Javascript` URL şemasını kullama (`href="javascript:alert('Test')"` için olduğu gibi).
- İlk parametre (işlevin senkronize olmayan bir şekilde çalışacağını tanımlayan) işlev adı (`setTimeout('x = 4', 1000)` gibi) yerine bir dize (değerlendirilecek) olduğunda `setInterval()` veya `setTimout()` işlevini kullanma.
- `document.write()` veya `document.writeln()` çağırma.

Uygulama güvenlik sanal alanındaki kod, içerik yüklenirken yalnızca bu yöntemleri kullanabilir.

Bu kısıtlamalar, JSON nesne değişmezleriyle `eval()` işlevinin kullanılmasını *engellemez*. Bu, uygulama içeriğinizin JSON JavaScript kitaplığıyla çalışmasını sağlar. Ancak, aşırı yüklü JSON kodu kullanımınız kısıtlanmıştır (olay işleyicileriyle).

Diğer Ajax çerçeveleri ve JavaScript kod kitaplıkları için, çerçevedeki veya kitaplıktaki kodun dinamik olarak oluşturulan kodlara ilişkin bu kısıtlamalarla çalışıp çalışmadığını kontrol edin. Çalışmıyorsa, uygulama dışı güvenlik sanal alanına çerçeveyi veya kitaplığı kullanan içerik dahil edin. Ayrıntılar için bkz. "AIR uygulamasının içindeki JavaScript'in sınırlılıkları" sayfa 1057 ve "Uygulama içeriği ve uygulama dışı içerik arasında komut dosyası oluşturma" sayfa 1102. Adobe, http://www.adobe.com/products/air/develop/ajax/features/ linklerinde uygulama güvenlik sanal alanını desteklemek için bilinen Ajax çerçevelerinin listesini tutar.

Uygulama güvenlik sanal alanındaki içerikten farklı olarak, uygulama dışı bir güvenlik sanal alanında yer alan JavaScript içeriği, dinamik olarak oluşturulan kodu yürütmek için her zaman `eval()` işlevini *çağırabilir*.

AIR API'lerine erişim sınırlamaları (uygulama dışı sanal alanlar için)

Adobe AIR 1.0 ve üstü

Uygulama dışı sanal alandaki JavaScript kodunun `window.runtime` nesnesine erişimi yoktur, ve aynı şekilde bu kod AIR API'lerini yürütemez. Uygulama dışı güvenlik sanal alanındaki içerik aşağıdaki kodu çağırdığında, uygulama bir TypeError istisnası atar:

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

Uygulama dışı sanal alandaki içerik `window.runtime` nesnesini tanımadığından, istisna TypeError (tanımlanmamış değer) türündedir, bu nedenle tanımlanmamış bir değer olarak görülür.

Uygulama dışı sanal alandaki içeriğe komut dosyası köprüsünü kullanarak çalışma zamanı işlevi gösterebilirsiniz. Ayrıntılar için bkz. "Uygulama içeriği ve uygulama dışı içerik arasında komut dosyası oluşturma" sayfa 1102.

XMLHttpRequest çağrılarının kullanımına ilişkin kısıtlamalar

Adobe AIR 1.0 ve üstü

Uygulama güvenlik sanal alanındaki HTML içeriği, HTML içeriği yüklenirken ve `onLoad` olayı sırasında uygulama sanal alanının dışından veri yüklemek için senkronize XMLHttpRequest yöntemlerini kullanamaz.

Varsayılan olarak, uygulama dışı güvenlik sanal alanlarındaki HTML içeriği, isteği çağıran etki alanı dışındaki etki alanlarından veri yüklemek üzere JavaScript XMLHttpRequest nesnesini kullanmak için izne sahip değildir. Bir `frame` veya `iframe` etiketi `allowcrosscomainxhr` niteliğini dahil edebilir. Bu niteliği herhangi bir null olmayan değere ayarlamak, frame veya iframe'deki içeriğin isteği çağıran kodun etki alanından başka etki alanlarından veri yüklemek için JavaScript XMLHttpRequest nesnesini kullanmasına izin verir.

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

Daha fazla bilgi için bkz. “Farklı etki alanlarındaki içerikler arasında komut dosyası oluşturma” sayfa 1098.

CSS, kare, iframe ve img öğelerinin yüklenmesine ilişkin kısıtlamalar (uygulama dışı sanal alanlardaki içerik için)

Adobe AIR 1.0 ve üstü

Uzak (ağ) güvenlik sanal alanlarındaki HTML içeriği uzak sanal alanlardan (ağ URL'lerinden) yalnızca `frame`, `iframe` ve `img` içeriği yükleyebilir.

Dosya sistemiyle yerel, ağ iletişimiyle yerel veya yerel-güvenilir sanal alanlardaki HTML içeriği yerel sanal alanlardan (uygulamadan veya uzak sanal alanlardan değil) yalnızca frame, iframe ve `img` içeriğini okuyabilir.

JavaScript window.open() yönteminin çağrılmasına ilişkin kısıtlamalar

Adobe AIR 1.0 ve üstü

JavaScript `window.open()` yöntemine çağrı yoluyla oluşturulan bir pencere, uygulama dışı güvenlik sanal alanından içerik görüntülediğinde, pencere başlığı ana (başlatılan) pencerenin başlığıyla başlar ve iki nokta karakteriyle devam eder. Pencere başlığının bu bölümünü ekranın dışına taşımak için kodu kullanamazsınız.

Uygulama dışı güvenlik sanal alanlarındaki içerik, kullanıcı fare veya klavye etkileşimiyle tetiklenen bir olaya yanıt olarak yalnızca JavaScript `window.open()` yöntemini başarıyla çağırabilir. Bu, uygulama dışı içeriğin aldatıcı biçimde kullanılabilecek pencereler oluşturmasını engeller (örneğin kimlik avı saldırıları). Ayrıca, fare veya klavye olayı için olay işleyici, `window.open()` yöntemini gecikmeden sonra yürütülmek üzere ayarlayamaz (örneğin `setTimeout()` işlevini çağırarak).

Uzak sanal alanlardaki (ağ) içerik, içeriği uzak ağ sanal alanlarında açmak için yalnızca `window.open()` yöntemini kullanabilir. İçeriği uygulamadan veya yerel sanal alanlardan açmak için `window.open()` yöntemini kullanamaz.

Dosya sistemiyle yerel, ağ iletişimiyle yerel veya yerel-güvenli sanal alanlardaki (bkz. “Güvenlik sanal alanları” sayfa 1055) içerik, yerel sanal alanlarda içerik açmak için yalnızca `window.open()` yöntemini kullanabilir. İçeriği uygulamadan veya uzak sanal alanlardan açmak için `window.open()` yöntemini kullanamaz.

Kısıtlanmış kodu çağırmada hata

Adobe AIR 1.0 ve üstü

Bu güvenlik kısıtlamaları nedeniyle sanal alanda kullanımı kısıtlanmış olan kodu çağırdığınızda çalıştırma zamanı bir JavaScript hatası gönderir: "Uygulama güvenlik sanal alanında JavaScript koduna ilişkin Adobe AIR çalıştırma zamanı güvenlik ihlali."

Daha fazla bilgi için bkz. “Güvenlikle ilişkili JavaScript hatalarını önleme” sayfa 993.

Dizeden HTML içeriği yükleme sırasında sanal alan koruması

Adobe AIR 1.0 ve üstü

HTMLLoader sınıfının `loadString()` yöntemi, çalışma zamanında HTML içeriği oluşturmanızı sağlar. Ancak, HTML içeriği olarak kullandığınız veriler güvenli olmayan bir Internet kaynağından yüklendiyse, bozuk olabilir. Bu nedenle, `loadString()` yöntemi kullanılarak oluşturulan HTML varsayılan olarak uygulama sanal alanına yerleştirilmez ve AIR API'lerine erişimi yoktur. Ancak, `loadString()` yöntemi kullanılarak oluşturulan HTML'yi uygulama sanal alanına yerleştirmek için HTMLLoader nesnesinin `placeLoadStringContentInApplicationSandbox` özelliğini true olarak ayarlayabilirsiniz. Daha fazla bilgi için bkz. “Bir dizeden HTML içeriğini yükleme” sayfa 992.

Farklı etki alanlarındaki içerikler arasında komut dosyası oluşturma

Adobe AIR 1.0 ve üstü

AIR uygulamaları yüklendiklerinde, özel ayrıcalıklara sahip olurlar. Aynı ayrıcalıkların uygulamanın parçası olmayan uzak dosyalar ve yerel dosyalar dahil diğer içeriğe verilmemesi önemlidir.

AIR sanal alan köprüsü hakkında

Adobe AIR 1.0 ve üstü

Etki alanlarında yer alan içerik, normalde diğer etki alanlarında yer alan komut dosyalarını çağıramaz. AIR uygulamalarını ayrıcalıklı bilgi veya kontrolün yanlışlıkla verilmesinden korumak için, `application` sanal alanındaki içeriğe (uygulamayla yüklenen içerik) aşağıdaki kısıtlamalar yerleştirilmiştir:

- Uygulama güvenlik sanal alanındaki kod, `Security.allowDomain()` yöntemini çağırarak diğer sanal alanlara izin veremez. Bu yöntemin uygulama güvenlik sanal alanından çağrılması hata oluşturur.
- Uygulama dışı içeriğin `LoaderContext.securityDomain` veya `LoaderContext.applicationDomain` özelliği ayarlanarak uygulama sanal alanına aktarılması engellenir.

Yine de AIR uygulamasının ana AIR uygulamasındaki komut dosyalarına denetimli erişime sahip olmak için uzak bir etki alanından içerik istediği, veya tam tersi durumlar olur. Çalışma zamanı, bunu gerçekleştirmek için iki sanal alan arasında ağ geçidi olarak görev yapan bir *sanal alan köprüsü* mekanizması sağlar. Bir sanal alan köprüsü, uzak ve uygulama güvenlik sanal alanları arasında açık etkileşim sağlar.

Sanal alan köprüsü, hem yüklenmiş hem de yüklenen komut dosyalarının erişebileceği iki nesne gösterir:

- `parentSandboxBridge` nesnesi, yüklenen içeriğin yüklenmiş içerikteki komut dosyalarına özellikler ve işlevler göstermesini sağlar.
- `childSandboxBridge` nesnesi, yüklenmiş içeriğin yüklenen içerikteki komut dosyalarına özellikler ve işlevler göstermesini sağlar.

Sanal alan köprüsü yoluyla gösterilen nesneler başvuruya göre değil, değere göre iletilir. Tüm veriler sıralanmıştır. Bu, köprünün bir tarafıyla gösterilen nesnelerin diğer tarafıyla ayarlanamayacağı anlamına gelir ve gösterilen tüm nesneler türsüzdür. Ayrıca, yalnızca basit nesneler ve işlevleri gösterebilirsiniz; karmaşık nesneleri gösteremezsiniz.

Alt içerik bir nesneyi parentSandboxBridge nesnesine ayarlamaya çalıştığında, çalışma zamanı bir SecurityError istisnası atar. Benzer şekilde, üst içerik bir özelliği childSandboxBridge nesnesine ayarlamaya çalıştığında, çalışma zamanı bir SecurityError istisnası atar.

Sanal alan köprüsü örneği (SWF)

Adobe AIR 1.0 ve üstü

Bir AIR müzik deposu uygulamasının albümlerin fiyatını yayınlamak için uzak SWF içeriğine izin vermek istediğini, ancak uzak SWF dosyasının fiyatın sağış fiyatı olup olmadığını açıklamasını istemediğini düşünün. Bu amaçla, StoreAPI sınıfı fiyatı elde etmek için bir yöntem sağlar, ancak satış fiyatını saklı tutar. Daha sonra StoreAPI sınıfının bir örneği, uzak SWF'yi yükleyen Loader nesnesinin LoaderInfo nesnesine ait `parentSandboxBridge` özelliğine atanır.

Aşağıdaki, AIR müzik deposuna ilişkin koddur:

```
<?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>
```

StoreAPI nesnesi, normal albüm fiyatını elde etmek için ana uygulamayı çağırır, ancak `getSaleAlbumPrice()` yöntemi çağrıldığında "Yok" mesajını döndürür. Aşağıdaki kod StoreAPI sınıfını tanımlar:

```
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();
    }
}
```

Aşağıdaki kod, depo fiyatını bildiren, ancak satış fiyatını bildiremeyen bir PriceQuoter SWF dosyası örneğini temsil eder:

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

Sanal alan köprüsü örneği (HTML)

Adobe AIR 1.0 ve üstü

HTML içeriğinde, `parentSandboxBridge` ve `childSandboxBridge` özellikleri alt belgenin JavaScript pencere nesnesine eklenir. HTML içeriğinde köprü işlevlerinin nasıl kurulacağına ilişkin bir örnek için bkz. "Sanal alan köprü arabirimini kurma" sayfa 1010.

API pozlamasını sınırlama

Adobe AIR 1.0 ve üstü

Sanal alan köprüleri pozlanırken, kötüye kullanma derecesini sınırlayan üst düzey API'ler göstermek önemlidir. Köprü uygulamanızı çağıran içeriğin ödün verilmiş bir içerik olabileceğini aklınızda bulundurun (örneğin kod yerleştirmesi yoluyla). Bu nedenle, örneğin köprü yoluyla `readFile(path:String)` yönteminin (rasgele bir dosyanın içeriğini okuyan) gösterilmesi kötüye kullanım için açık oluşturabilir. Bir yol almayan ve belirli bir dosya okumayan `readApplicationSetting()` API'sinin gösterilmesi daha uygundur. Daha anlamsal bir yaklaşım, bir bölümünden ödün verildiğinde uygulamanın vereceği hasarı sınırlar.

Daha fazla Yardım konusu

"Farklı güvenlik sanal alanlarında çapraz komut dosyası içeriği" sayfa 1008

"AIR uygulamasının sanal alanı" sayfa 1056

Diske yazma

Adobe AIR 1.0 ve üstü

Bir web tarayıcısında çalışan uygulamalar, kullanıcının yerel dosya sistemiyle sınırlı olarak etkileşime girer. Web tarayıcıları, web içeriği yüklemenin kullanıcının bilgisayarından ödün verilmesiyle sonuçlanmamasını sağlamak için güvenlik politikaları uygular. Örneğin, bir web tarayıcısında Flash Player aracılığıyla çalışan SWF dosyaları, önceden kullanıcının bilgisayarında olan dosyalarla doğrudan etkileşim kuramaz. Paylaşılan nesneler ve tanımlama bilgileri, kullanıcı tercihlerinin ve diğer verilerin korunması amacıyla kullanıcının bilgisayarına yazılabilir, ancak bu, dosya sistem etkileşiminin sınırıdır. AIR uygulamaları, yerel olarak kurulduğundan, yerel dosya sistemi içinde okuyup yazabilme özelliğini içeren farklı bir güvenlik sözleşmesine sahiptir.

Bu özgürlük, geliştiriciler için beraberinde yüksek sorumluluk getirir. Kaza eseri oluşan uygulama tehlikeleri yalnızca uygulama işlevlerini değil, kullanıcı bilgisayarının bütünlüğünü de tehdit eder. Bu nedenle, geliştiriciler "Geliştiriciler için en iyi güvenlik uygulamaları" sayfa 1103 bölümünü okumalıdır.

AIR kullanıcıları, çeşitli URL şeması kurallarını kullanarak yerel dosya sistemine erişebilir ve dosyaları yazabilirler:

URL şeması	Açıklama
app:/	Uygulama dizinine başka ad. Bu yoldan erişilen dosyalar uygulama sanal alanına atanır ve çalışma zamanı tarafından verilen tüm ayrıcalıklara sahiptir.
app-storage:/	Yerel depo dizinine başka ad, çalışma zamanı tarafından standartlaştırılmış. Bu yoldan erişilen dosyalar uygulama dışı bir sanal alana atanır.
file:///	Kullanıcının sabit diskinin kökünü temsil eden başka ad. Bu yoldan erişilen bir dosya, uygulama dizininde bulunuyorsa uygulama sanal alanına, aksi halde uygulama dışı sanal alana atanır.

Not: *AIR uygulamaları app: URL şeması kullanarak içeriği değiştiremezler. Ayrıca, uygulama dizini yönetici ayarları nedeniyle salt okunur olabilir.*

Kullanıcının bilgisayarında yönetici kısıtlamaları olmadığı sürece, AIR uygulamaları kullanıcının sabit sürücüsünde herhangi bir konuma yazma ayrıcalığına sahiptir. Geliştiricilerin uygulamalarıyla ilişkili yerel depo için `app-storage:/` yolunu kullanmaları önerilir. Uygulamadan `app-storage:/` hedefine yazılan dosyalar standart bir konuma yerleştirilir.

- Mac OS'de: kullanıcının tercihler klasörü `<appData>` olduğunda, bir uygulamanın depo dizini `<appData>/<appId>/Local Store/` olur. Bu genellikle `/Users/<user>/Library/Preferences` konumundadır.
- Windows'ta: Kullanıcının CSIDL_APPDATA Özel Klasörü `<appData>` olduğunda, uygulamanın depo dizini `<appData>\<appId>\Local Store\` olur. Bu genellikle, `C:\Documents and Settings\<userName>\Application Data` konumundadır.
- Linux'ta: `<appData>`, `/home/<user>/.appdata` olduğunda `<appData>/<appID>/Local Store/` olur.

Bir uygulama kullanıcının dosya sisteminde var olan dosyalarla etkileşim kurmak üzere tasarlandıysa, "Geliştiriciler için en iyi güvenlik uygulamaları" sayfa 1103 bölümünü okuduğunuzdan emin olun.

Güvenilmeyen içerikle güvenli biçimde çalışma

Adobe AIR 1.0 ve üstü

Uygulama sanal alanına atanmayan içerik, ancak çalışma zamanının güvenlik ölçütlerini karşılayabilirse uygulamanıza ek komut dosyası işlevleri sağlayabilir. Bu başlık, uygulama dışı içerikle AIR güvenlik sözleşmesini açıklar.

Security.allowDomain()

Adobe AIR 1.0 ve üstü

AIR uygulamaları, uygulama dışı içerik için komut dosyası erişimini, Flash Player tarayıcı eklentisinin güvenilmeyen içerik için komut dosyası erişimini kısıtladığından daha sıkı biçimde kısıtlar. Örneğin, `local-trusted` bir sanal alana atanmış SWF dosyası `System.allowDomain()` yöntemini çağırdığında, tarayıcıdaki Flash Player'da belirtilen etki alanından yüklenen herhangi bir SWF'ye komut dosyası oluşturma erişimi verilir. Kullanıcının bilgisayarındaki uygulama dışı dosyalara akılcı olmayan bir biçimde erişim sağlayacağından, benzer yaklaşıma AIR uygulamalarındaki `application` içeriğinden izin verilmez. `Security.allowDomain()` yöntemine yapılan çağrılar dikkate alınmaksızın, uzak dosyalar uygulama sanal alanına doğrudan erişemez.

Uygulama içeriği ve uygulama dışı içerik arasında komut dosyası oluşturma

Adobe AIR 1.0 ve üstü

Uygulama içeriği ve uygulama dışı içerik arasında komut dosyası oluşturan AIR uygulamaları, daha karmaşık güvenlik düzenlemelerine sahiptir. Uygulama sanal alanında yer almayan dosyalar, sanal alan köprüsü kullanımı yoluyla yalnızca uygulama sanal alanındaki dosyaların özelliklerine ve yöntemlerine erişebilir. Bir sanal alan köprüsü, uygulama içeriği ve uygulama dışı içerik arasında, iki dosya arasındaki açık etkileşimi sağlayan bir ağ geçidi olarak görev yapar. Sanal alan köprüleri, doğru kullanıldığında uygulama dışı içeriğin uygulama içeriğinin parçası olan nesne başvurularına erişimini kısıtlayarak fazladan bir güvenlik katmanı sağlar.

Sanal alan köprülerinin faydaları, örnekle en iyi şekilde gösterilebilir. AIR müzik deposu uygulamasının, depo uygulamasının sonradan iletişim kurabileceği kendi SWF dosyalarını oluşturmak isteyen reklamcılara bir API sağlamak istediğini düşünün. Depo, reklamcılara depodan sanatçıları ve CD'leri arayabilecekleri yöntemler sağlamak, ancak aynı zamanda güvenlik nedenleriyle bazı yöntemleri ve özellikleri üçüncü taraf SWF dosyasının kullanımına kapatmak istiyor.

Bu işlevleri bir sanal alan köprüsü sağlayabilir. Varsayılan olarak, bir AIR uygulamasına çalışma zamanında harici olarak yüklenen içerik, ana uygulamadaki yöntemler ve özelliklere erişemez. Geliştirici, özel bir sanal alan köprüsü uygulamasıyla bu yöntemleri ve özellikleri göstermeden uzak içeriğe servis sağlayabilir. Sanal alan köprüsünü, yükleyen ve yüklenen içerik arasındaki iletişimi nesne başvurularını göstermeden sağlayan, güvenilen ve güvenilmeyen içerik arasında yer alan bir patika olarak düşünün.

Sanal alan köprülerinin nasıl güvenli biçimde kullanılacağına ilişkin daha fazla bilgi için bkz. "Farklı etki alanlarındaki içerikler arasında komut dosyası oluşturma" sayfa 1098.

Dinamik olarak güvenilir olmayan SWF içeriği oluşturmaya karşı koruma

Adobe AIR 1.0 ve üstü

`Loader.loadBytes()` yöntemi, uygulamanın bir bayt dizisinden SWF içeriği oluşturması için bir yol sağlar. Ancak, uzak kaynaktan yüklenen verilerdeki yerleştirme saldırıları, içerik yükleme sırasında ciddi hasara neden olabilir. Bu, özellikle oluşturulan SWF içeriği tüm AIR API'leri kümesine erişebiliyorsa, uygulama sanal alanına veri yükleme sırasında geçerlidir.

`loadBytes()` yöntemini yürütülebilir SWF kodu oluşturmadan kullanabilen geçerli kullanıcılar vardır. Örneğin, görüntü zamanlamasını denetlemek için görüntü verileri oluşturmak amacıyla `loadBytes()` yöntemini kullanabilirsiniz. Ses oynatma için dinamik SWF oluşturma gibi, kod yürütmeyi *kullanan* geçerli kullanıcılar da vardır. AIR'de, `loadBytes()` yöntemi varsayılan olarak SWF içeriği yüklemenize izin *vermez*; yalnızca görüntü içeriği yüklemenize izin verir. AIR'de `loadBytes()` yönteminin `loaderContext` özelliği, uygulamanın yürütülebilir SWF içeriğini yüklemek üzere `loadBytes()` kullanmasına izin vermek için `true` olarak ayarlayabileceğiniz `allowLoadBytesCodeExecution` özelliğini içerir. Aşağıdaki kod, bu özelliğin nasıl kullanılacağını gösterir:

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

SWF içeriği yüklemek için `loadBytes()` çağırdığınızda ve LoaderContext nesnesinin `allowLoadBytesCodeExecution` özelliği `false` (varsayılan) olarak ayarlandığında, the Loader nesnesi bir SecurityError istisnası atar.

Not: *Adobe AIR'in gelecekteki bir sürümünde bu API değişebilir. Bu gerçekleştiğinde, LoaderContext sınıfının `allowLoadBytesCodeExecution` özelliğini kullanan içeriği yeniden derlemeniz gerekebilir.*

Geliştiriciler için en iyi güvenlik uygulamaları

Adobe AIR 1.0 ve üstü

AIR uygulamaları web teknolojileri kullanılarak oluşturulmuş olsa da, geliştiricilerin tarayıcı güvenlik sanal alanı içinde çalışmadıklarını bilmeleri önemlidir. Bu, yerel sisteme istemli veya istemsiz zarar verebilecek AIR uygulamaları oluşturmanın mümkün olduğu anlamına gelir. AIR bu riski en aza indirmeye çalışmaktadır, ancak yine de güvenlik açıklarının kullanılabileceği yollar mevcuttur. Bu başlık, önemli olası tehlikeleri konu alır.

Dosyaları uygulama güvenlik sanal alanına aktarmakdan doğan risk

Adobe AIR 1.0 ve üstü

Uygulama dizininde bulunan dosyalar, uygulama sanal alanına atanır ve tüm çalışma zamanı ayrıcalıklarına sahiptir. Yerel dosya sistemine yazan uygulamaların `app-storage:/` hedefine yazmaları önerilir. Dosyalar uygulama sanal alanına atanmadığından ve daha düşük bir güvenlik riski sunduğundan, bu dizin kullanıcının bilgisayarındaki uygulama dosyalarından ayrı olarak bulunur. Geliştiricilerin şunları dikkate alması önerilir:

- Bir dosyayı, yalnızca gerekli olduğunda bir AIR dosyasına dahil edin (yüklenen uygulamada).
- Bir komut dosyasını yalnızca davranışı tam olarak anlaşılır ve güvenilir olduğunda bir AIR dosyasına dahil edin (yüklenen uygulamada).
- Uygulama dizinindeki içeriğe yazmayın veya bu içeriği değiştirmeyin. Çalışma zamanı, SecurityError istisnasını atarak, uygulamaların `app:/` URL şemasını kullanarak dosyaları ve dizinleri yazmasını veya değiştirmesini engeller.
- Kod yürütmesine neden olabilecek AIR API yöntemlerine ilişkin parametreler olan ağ kaynaklarından gelen verileri kullanmayın. Bu, `Loader.loadBytes()` yöntemini ve JavaScript `eval()` işlevini içerir.

Yolları belirlemek için harici bir kaynak kullanmaktan doğan risk

Adobe AIR 1.0 ve üstü

Harici veri veya içerik kullanımı sırasında, AIR uygulamasından ödün verilebilir. Bu nedenle, ağdan veya dosya sisteminden veri kullanırken özellikle dikkat edin. Güven sorumluluğu nihayetinde geliştiriciye ve kurduğu ağ bağlantılarına bağlıdır, ancak yabancı veriler yapıları gereği risklidir ve hassas işlemlerde girdi olarak kullanılmamalıdır. Geliştiriciler, aşağıdakilere karşı uyarılır:

- Dosya adını belirlemek için bir ağ kaynağından verileri kullanma
- Uygulamanın özel bilgi göndermek için kullandığı bir URL yapmak için bir ağ kaynağından verileri kullanma

Güvenli olmayan kimlik bilgilerinin kullanılması, saklanması veya iletilmesinden doğan risk

Adobe AIR 1.0 ve üstü

Kullanıcı kimlik bilgilerinin kullanıcının yerel dosya sisteminde saklanması, yapısı gereği bu kimlik bilgilerinden ödün verilebileceği riskini doğurur. Geliştiricilerin şunları dikkate alması önerilir:

- Kimlik bilgilerinin yerel olarak saklanması gerekiyorsa, yerel dosya sistemine yazma sırasında kimlik bilgilerini şifreleyin. Çalışma zamanı, EncryptedLocalStore sınıfı aracılığıyla, yüklenen her uygulamaya benzersiz şifrelenmiş bir depo sağlar. Ayrıntılar için, bkz. "Şifrelenmiş yerel depolama" sayfa 718.
- Kodlanmamış kullanıcı kimlik bilgilerini, kaynak güvenilir olana ve iletim HTTPS: ya da Aktarım Katmanı Güvenliği (TLS) protokollerini kullanana dek bir ağ kaynağına iletmeyin.
- Kimlik bilgileri oluşturmada hiçbir zaman varsayılan bir parola belirlemeyin - kullanıcıların kendi parolalarını oluşturmalarına izin verin. Varsayılanı değişiklik olmadan kullanan kullanıcılar, kimlik bilgilerini varsayılan parolayı önceden bilen bir saldırgana göstermiş olur.

İndirgeme saldırısından doğan risk

Adobe AIR 1.0 ve üstü

Uygulama yükleme sırasında, çalışma zamanı uygulamanın bir sürümünün yüklü olup olmadığını kontrol eder. Bir uygulama önceden yüklenmişse, çalışma zamanı sürüm dizesini yüklenmekte olan uygulamanın sürümüyle karşılaştırır. Bu dize farklıysa, kullanıcı yüklemeyi yükseltmeyi seçebilir. Çalışma zamanı, yeni yüklenen sürümün eski sürümden daha yeni olacağını garanti etmez, yalnızca değişik olduğunu garanti eder. Saldırgan, bir güvenlik açığını atlatmak için kullanıcıya daha eski bir sürüm dağıtabilir. Bu nedenle, uygulama çalışırken geliştiricinin sürüm kontrolleri yapması önerilir. Uygulamaların gerekli güncellemeler için ağı kontrol etmesi, iyi bir fikirdir. Bu şekilde, saldırgan kullanıcının eski sürümü çalıştırmasını sağlasa bile, bu eski sürüm güncellenmesi gerektiğini anlayacaktır. Ayrıca, uygulamanız için açık bir sürüm belirleme şeması kullanmak, düşük bir sürüm yüklemeleri için kullanıcıların kandırılmasını zorlaştıracaktır.

Kod imzalama

Adobe AIR 1.0 ve üstü

Tüm AIR yükleyici dosyaları kodlarının imzalanmış olması gerekir. Kod imzalama, belirtilen yazılım kaynağının doğru olduğunu teyit etmeye ilişkin bir şifreleme işlemidir. AIR uygulamaları, harici sertifika yetkilileri (CA) tarafından yayınlanan bir sertifika veya kendinizin oluşturduğu sertifika kullanılarak imzalanır. İyi bilinen bir CA'dan alınan ticari bir sertifika kesinlikle önerilir ve kullanıcılarınıza sahte bir ürünü değil, sizin uygulamanızı yüklediklerine ilişkin güvence sağlar. Ancak, kendinden imzalı sertifikalar SDK'den `adt` kullanılarak veya Flash, Flash Builder ya da sertifika oluşturma için `adt` kullanan herhangi bir uygulama kullanılarak oluşturulabilir. Kendinizin oluşturduğu sertifika, kurulan uygulamanın orijinal olup olmadığının garantisini vermez ve yalnızca, piyasaya çıkmasından önce test amaçlı kullanılması gerekir.

Android aygıtlarında güvenlik

Adobe AIR 2.5 ve üstü

Tüm bilgisayar aygıtlarında olduğu gibi Android'de AIR yerel güvenlik modeline uyum sağlar. Aynı zamanda, AIR kendi güvenlik kurallarını da korur. Bunların amacı geliştiricilerin güvenli, İnternete bağlı uygulamalar yazmasını kolaylaştırmaktır.

Android'deki AIR uygulamaları Android paketi biçimini kullandığından yükleme Android güvenlik modeli altındadır. AIR uygulaması yükleyicisi kullanılmaz.

Android güvenlik modelinin üç ana yönü vardır:

- İzinler
- Uygulama imzaları
- Uygulama kullanıcısı kimlikleri

Android izinleri

Android'in birçok özelliği işletim sistemi izin mekanizması tarafından korunur. Korunan bir özelliği kullanmak için AIR uygulama açıklayıcısı uygulamanın gerekli izni gerektirdiğini belirtmelidir. Bir kullanıcı uygulamayı yüklemeyi denediğinde, Android işletim sistemi yüklemeye devam etmeden önce tüm istenen izinleri kullanıcıya gösterir.

Çoğu AIR uygulamasının uygulama açıklayıcısında Android izinleri belirtmesi gerekir. Varsayılan olarak izinler dahil edilmez. AIR çalışma zamanı ile kullanılan korunan Android özellikleri için aşağıdaki izinler gereklidir:

ACCESS_COARSE_LOCATION Geolocation sınıfı aracılığıyla uygulamanın WIFI ve hücresel ağ konumu verilerine erişmesine izin verir.

ACCESS_FINE_LOCATION Uygulamanın Geolocation sınıfı aracılığıyla GPS verilerine erişmesine izin verir.

ACCESS_NETWORK_STATE ve ACCESS_WIFI_STATE Uygulamanın NetworkInfo sınıfının ağ bilgilerine erişmesine izin verir.

CAMERA Uygulamanın kameraya erişmesine izin verir.

INTERNET Uygulamanın ağ isteği yapmasına izin verir. Ayrıca uzaktan hata ayıklamaya izin verir.

READ_PHONE_STATE AIR çalışma zamanının gelen çağrı olması durumunda sesi kapatmasına izin verir.

RECORD_AUDIO Uygulamanın mikrofona erişmesine izin verir.

WAKE_LOCK ve DISABLE_KEYGUARD Uygulamanın SystemIdleMode sınıfı ayarlarını kullanarak aygıtın uyku moduna girmesini engellemesine izin verir.

WRITE_EXTERNAL_STORAGE Uygulamanın aygıttaki harici bellek kartına yazmasına izin verir.

Uygulama imzaları

Android platformu için oluşturulan tüm uygulama paketleri imzalanmalıdır. Android'deki AIR uygulamaları yerel Android APK biçiminde paketlendiğinden, AIR kuralları yerine Android kurallarına göre imzalanır. Android ve AIR kod imzalamayı benzer şekilde kullansa da önemli farklılıklar vardır:

- Android'de imza özel anahtara geliştiricinin sahip olduğunu doğrular ancak geliştiricinin kimliğini doğrulamak için kullanılmaz.
- Android pazarına gönderilmiş uygulamalar için sertifika en az 25 yıl geçerli olmalıdır.
- Android paket imzasının başka bir sertifikaya taşınmasını desteklemez. Bir güncelleme farklı bir sertifika tarafından imzalıysa, kullanıcıların güncellenmiş uygulamayı yükleyebilmeleri için orijinal uygulamayı kaldırmaları gerekir.
- Aynı sertifikaya sahip iki uygulama birbirlerinin önbelleğine ve veri dosyalarına erişmelerine izin veren ortak bir kimlik belirtebilirler. (AIR bu tarz bir paylaşım sunmaz.)

Uygulama kullanıcısı kimlikleri

Android Linux çekirdeği kullanır. Yüklü her uygulamaya dosya erişimi gibi işlemler için izinlerini belirleyen Linux türü bir kullanıcı kimliği atanır. Uygulamadaki, uygulama deposundaki ve geçici dizinlerdeki dosyalara erişim dosya sistemi izinleri tarafından korunur. Harici depoya yazılan dosyalar (diğer bir deyişle SD karta) SD kartı bilgisayara yığın depolama aygıtı olarak takıldığında diğer uygulamalar veya kullanıcı tarafından okunabilir, değiştirilebilir ve silinebilir.

İnternet istekleriyle alınan tanımlama bilgileri AIR uygulamaları arasında paylaşılmaz.

Arka plan görüntüsü gizliliği

Kullanıcı, bir uygulamayı arka plana geçirdiğinde bazı Android sürümleri, son uygulamalar listesinde minik resim olarak kullanan bir ekran görüntüsü alır. Bu ekran görüntüsü cihazın belleğinde saklanır ve cihazın fiziksel kontrolüne sahip bir saldırgan bu görüntüye erişebilir.

Uygulamanız hassas bilgiler görüntülüyorsa, bu tür bilgilerin arka plan ekran görüntüsü tarafından yakalanmasına karşı korunmalısınız. NativeApplication nesnesi tarafından gönderilen `deactivate` olayı bir uygulamanın arka plana geçmek üzere olduğunu gösterir. Hassas bilgileri temizlemek veya gizlemek için bu olayı kullanın.

Daha fazla Yardım konusu

Android: Güvenlik ve İzinler

Android'de şifrelenmiş veriler

Android'deki AIR uygulamaları şifrelenmiş verileri kaydetmek için yerleşik SQL veri tabanında bulunan şifreleme seçeneklerini kullanabilir. En uygun güvenlik için, şifreleme anahtarını, uygulamanın her çalıştırılışında girilen kullanıcı girişli bir şifre üzerine temellendirin. Yerel olarak depolanmış şifreleme anahtarının ya da şifrenin uygulama dosyalarına erişimi olan bir saldırgandan "saklanması" zor veya imkansızdır. Saldırgan anahtarı alabilirse, bu durumda verileri şifrelemek, Android sistemi tarafından sağlanan kullanıcı kimliği bazlı dosya sistemi güvenliği dışında hiçbir ek koruma sağlayamaz.

EncryptedLocalStore sınıfı veri kaydetmek için kullanılabilir ancak bu veri Android cihazlarında şifreli değildir. Bunun yerine, Android güvenlik modeli, verileri diğer uygulamalardan korumak için uygulama kullanıcı kimliğine güvenir. Paylaşılan bir kullanıcı kimliği kullanan ve aynı kod imzalama dosyası ile imzalanan uygulamalar aynı şifrelenmiş yerel depoyu kullanır.

Önemli: *Root edilmiş bir telefonda, kök kullanıcı yetkileri ile çalışan her uygulama, diğer her uygulamanın dosyalarına erişebilir. Bu nedenle, şifreli yerel depo kullanılarak depolanan veriler root edilmiş bir cihazda güvende değildir.*

iOS aygıtlarında güvenlik

iOS'de, AIR yerel güvenlik modeline uyum sağlar. Aynı zamanda, AIR kendi güvenlik kurallarını da korur. Bunların amacı geliştiricilerin güvenli, İnternete bağlı uygulamalar yazmasını kolaylaştırmaktır.

iOS'deki AIR uygulamaları iOS paketi biçimini kullandığından yükleme iOS güvenlik modeli altındadır. AIR uygulaması yükleyicisi kullanılmaz. Ayrıca, iOS aygıtlarında ayrı bir AIR çalışma zamanı kullanılmaz. Her AIR uygulaması çalışabilmesi için gereken tüm kodu içerir.

Uygulama imzaları

iOS platformu için oluşturulan tüm uygulama paketleri imzalanmalıdır. iOS'deki AIR uygulamaları yerel iOS IPA biçiminde paketlendiğinden, AIR gereksinimleri yerine iOS gereksinimlerine göre imzalanırlar. iOS ve AIR kod imzalamayı benzer şekilde kullansa da önemli farklılıklar vardır:

- iOS'de, uygulamayı imzalamak için kullanılan sertifika Apple tarafından verilmelidir; diğer sertifika yetkilileri tarafından verilen sertifikalar kullanılamaz.
- iOS'de, Apple'ın verdiği dağıtım sertifikaları genelde bir yıl süreyle geçerlidir.

Arka plan görüntüsü gizliliği

iOS'de bir kullanıcı uygulamayı arka plana geçirdiğinde, işletim sistemi geçişe hareket vermek için kullandığı bir ekran görüntüsü yakalar. Bu ekran görüntüsü cihazın belleğinde saklanır ve cihazın fiziksel kontrolüne sahip bir saldırgan bu görüntüye erişebilir.

Uygulamanız hassas bilgiler görüntülüyorsa, bu tür bilgilerin arka plan ekran görüntüsü tarafından yakalanmasına karşı korunmalısınız. NativeApplication nesnesi tarafından gönderilen `deactivate` olayı bir uygulamanın arka plana geçmek üzere olduğunu gösterir. Hassas bilgileri temizlemek veya gizlemek için bu olayı kullanın.

Bölüm 65: ActionScript Örnekleri Nasıl Kullanılır?

ActionScript 3.0 kodu örneği çalıştırmak, belirli sınıfların ve yöntemlerin nasıl çalıştığını öğrenmek için en iyi yollardan biridir. Kullandığınız veya hedeflediğiniz aygıtlara bağlı olarak örnekleri farklı şekillerde kullanabilirsiniz.

Flash Professional veya Flash Builder çalıştıran bilgisayarlar ActionScript 3.0 örneklerini çalıştırma amacıyla bu geliştirme ortamlarının nasıl kullanıldığı hakkında bilgi için "ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması" sayfa 1110 veya "ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması" sayfa 1111 bölümlerine bakın. Kod örneğinin nasıl çalıştığı ile ilgili bilgilerinizi artırmak için izleme ifadelerini ve diğer hata ayıklama araçlarını kullanın.

Mobil aygıtlar ActionScript 3.0 kod örneklerini, Flash Player 10.1 ve daha sonraki sürümleri destekleyen mobil aygıtlarda çalıştırabilirsiniz. Bkz. "ActionScript 3.0 örneklerinin taşınabilir cihazlarda çalıştırılması" sayfa 1112. Bu örnekleri, Flash Professional veya Flash Builder kullanarak da bilgisayarınızda çalıştırabilirsiniz.

TV aygıtları Bu örnekleri TV aygıtlarında çalıştıramasanız da, örnekleri bilgisayarınızda çalıştırarak bunlardan bilgi edinebilirsiniz. TV aygıtları uygulamalarını geliştirme hakkında bilgi için Adobe Developer Connection web sitesinde TV için Flash Platformu bölümüne bakın.

Örnek türleri

ActionScript 3.0 kod örneği türleri şu şekildedir:

- Kod parçacığı örnekleri (ActionScript 3.0 belge kümesinde bulunan)
- Sınıf bazlı örnekler (öncelikle ActionScript 3.0 Dil Referansı içinde bulunan)
- Çoklu kaynak dosyaları içeren pratik örnekler (Kaynak ZIP dosyalarını www.adobe.com/go/learn_programmingAS3samples_flash_tr adresinden indirin)

Kod parçacığı örnekleri:
Bir kod parçacığı örneğinin görünümü:

```
var x:int = 5;
trace(x); // 5
```

Kod parçacıkları tek bir düşünce ortaya koymaya yetecek ölçüde kod içerir. Paket ya da sınıf ifadeleri içermezler.

Sınıf bazlı örnekler
Eksiksiz bir ActionScript sınıfının kaynak kodunu gösteren birçok örnek vardır. Sınıf bazlı bir örnek görünümü:

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

Sınıf bazlı bir örneğin kodunda bir paket ifadesi, sınıf bildirimi ve yapıcı işlevi bulunur.

Çoklu kaynak dosyaları içeren pratik örnekler

ActionScript 3.0 Yazılım Geliştirici Kılavuzu'nun çoğu başlığında belli ActionScript özelliklerini pratik ve gerçek dünya koşullarında kullanmayı gösteren pratik örnekler bulunur. Bu örneklerde genellikle çoklu dosyalar vardır:

- Bir ya da birden fazla ActionScript kaynak dosyası
- Flash Professional ile birlikte kullanılacak .FLA dosyası
- Flash Builder ile birlikte kullanılacak bir veya daha fazla MXML dosyaları
- Veri dosyaları, görüntü dosyaları, ses dosyaları ve örnek uygulamada kullanılan (isteğe bağlı) diğer varlıklar.

Pratik örnekler genellikle ZIP arşiv dosyaları olarak verilir.

ZIP dosyasındaki Geliştirici Kılavuzı örneklerinin listesi

Flash Professional ve Flex 4 için ZIP dosyası (www.adobe.com/go/learn_programmingAS3samples_flash_tr adresinden indirin) şu örnekleri içerir:

- AlarmClock (“Olay işleme örneği: Çalar Saat” sayfa 142)
- AlgorithmicVisualGenerator (“API çizme örneği: Algoritmik Görsel Oluşturucu” sayfa 236)
- ASCIIArt (“Dize örneği: ASCII sanatı” sayfa 19)
- CapabilitiesExplorer (“Yetenekler örneği: Sistem yeteneklerini algılama” sayfa 883)
- CustomErrors (“Hataları işleme örneği: CustomErrors uygulaması” sayfa 70)
- DisplayObjectTransformer (“Geometri örneği: Görüntüleme nesnesine matris dönüştürmesi uygulama” sayfa 220)
- FilterWorkbench (“Görüntüleme nesnelerine filtre uygulama örneği: Filter Workbench” sayfa 294)
- GlobalStockTicker (“Örnek: Bir stok onaylayıcı uygulamasını uluslararasılaştırma” sayfa 964)
- IntrovertIM_HTML (“Harici API örneği: Bir web tarayıcısında ActionScript ve JavaScript arasındaki iletişim” sayfa 858)
- NewsLayout (“TextField Örneği: Gazete stili metin biçimlendirme” sayfa 390)
- PlayList (“Diziler örneği: PlayList” sayfa 48)
- PodcastPlayer (“Ses örneği: Podcast Oynatıcısı” sayfa 471)
- ProjectionDragger (“Örnek: Perspektif projeksiyonu” sayfa 359)
- ReorderByZ (“Görüntünün yeniden sıralanması için Matrix3D nesnelerini kullanma” sayfa 364)
- RSSViewer (“ActionScript'te XML örneği: internetten RSS verilerini yükleme” sayfa 114)
- RuntimeAssetsExplorer (“Film klibi örneği: RuntimeAssetsExplorer” sayfa 333)
- SimpleClock (“Tarih ve saat örneği: Basit analog saat” sayfa 6)
- SpinningMoon (“Bitmap örneği: Animasyonlu dönen ay” sayfa 257)
- SpriteArranger (“Görüntüleme nesnesi örneği: SpriteArranger” sayfa 207)
- TelnetSocket (“TCP socket örneği: Telnet istemcisi oluşturma” sayfa 807)
- VideoJukebox (“Video örneği: Video Jukebox” sayfa 511)
- WikiEditor (“Normal ifade örneği: Wiki ayrıştırıcısı” sayfa 91)
- WordSearch (“Fare girdisi örneği: WordSearch” sayfa 583)

Pratik örnekler, Flash Geliştirici Merkezi ve Flex Geliştirici Merkezi'ndeki çoğu Hızlı Başlatma makalelerinde de bulunabilir.

ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması

Flash Professional'ı kullanarak bir örnek çalıştırmak için aşağıdaki işlemlerden birini (örnek türüne bağlı olarak) kullanın.

Kod parçacığının örnek Flash Professional'da çalıştırılması

Kod parçacığını örnek Flash Professional'da çalıştırma:

1. Dosya > Yeni'yi seçin.
2. Yeni Belge iletişim kutusunda Flash Belgesi seçeneğini belirleyip Tamam'ı tıklatın.

   Yeni bir Flash penceresi görüntülenir.
3. Zaman çizelgesi panelinin ilk katmanının ilk karesini tıklatın.
4. Eylemler panelinde kod parçacığı örneğini yazın veya yapıştırın.
5. Dosya > Kaydet'i seçin. Dosyaya bir ad verin ve Tamam düğmesini tıklatın.
6. Örneği test etmek için Kontrol > Filmi Test Et seçeneğini belirleyin.

Sınıf bazlı örneğin Flash Professional'da çalıştırılması

Sınıf bazlı örneği Flash Professional'da çalıştırma:

1. Dosya > Yeni seçimini belirleyin.
2. Yeni Belge iletişim kutusunda ActionScript Dosyası'nı seçin ve Tamam'ı tıklatın. Yeni bir düzenleme penceresi görüntülenir.
3. Sınıf bazlı örnek kodunu kopyalayın ve düzenleme penceresine yapıştırın.

   Sınıf programın ana belge sınıfıysa, MovieClip sınıfını genişletmelidir:

   ```
   import flash.display.MovieClip;
   public class Example1 extends MovieClip{
   //...
   }
   ```

   Ayrıca örnekte adı geçen tüm sınıfların `import` ifadeleri kullanılarak bildirildiğinden emin olun.
4. Dosya > Kaydet seçimini belirleyin. Dosya adını örnekteki sınıf adının aynısını verin (ör. ContextMenuExample.as).

   Not: *flashx.textLayout.container.ContainerController sınıfı örneği gibi bazı sınıf temelli örnekler, paket bildiriminde birden fazla düzey içerir (`package flashx.textLayout.container.examples {`). Bu örnekler için, dosyayı paket bildirimiyle eşleşen bir alt dosyaya kaydedin (flashx/textLayout/container/examples) veya paket adını kaldırın (böylece ActionScript yalnızca `package {` ile başlar) ve dosyayı herhangi bir konumdan test edebilirsiniz.*
5. Dosya > Yeni seçimini belirleyin.
6. Yeni Belge iletişim kutusunda Flash Belgesi(ActionScript 3.0) öğesini seçip Tamam'ı tıklatın. Yeni bir Flash penceresi görüntülenir.

7 Az önce kaydettiğiniz ActionScript kaynak dosyasının adıyla eşleşmesi gereken örnek sınıfın adını Özellikler panelinde, Belge Sınıfı alanında yazın (ör. ContextMenuExample).

8 Dosya > Kaydet seçimini belirleyin. FLA dosya adına, örnekteki sınıf adının aynısını verin (ör. ContextMenuExample.fla).

9 Örneği test etmek için Kontrol > Filmi Test Et seçeneğini belirleyin.

Flash Professional'da pratik bir örnek çalıştırma

Pratik örnekler genellikle ZIP arşiv dosyaları olarak verilir. Flash Professional kullanarak pratik bir örnek çalıştırma:

1 Zip arşiv dosyasını seçtiğiniz bir klasöre ayıklayın.

2 Flash Professional'da, Dosya > Aç seçeneklerini belirleyin.

3 Arşiv dosyasını ayıkladığınız klasöre göz atın. O klasördeki FLA dosyasını seçin ve Aç seçeneğine tıklatın.

4 Örneği test etmek için Kontrol > Filmi Test Et seçeneğini belirleyin.

ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması

Flash Builder'ı kullanarak bir örnek çalıştırmak için aşağıdaki işlemlerden birini (örnek türüne bağlı olarak) kullanın.

Kod parçacığı örneğinin Flash Professional'da çalıştırılması

Kod parçacığı örneğini Flash Professional'da çalıştırma:

1 Yeni bir Flex Projesi oluşturun (Dosya > Yeni > Flex Projesi seçeneklerini belirleyin) ya da mevcut bir Flex projesinden yeni bir MXML Uygulaması (Dosya > Yeni > MXML Uygulaması seçeneklerini belirleyin) oluşturun. Projeye veya uygulamaya açıklayıcı bir isim verin (ContextMenuExample gibi).

2 Oluşturulmuş MXML dosyasının içinde `<mx:Script>` etiketi ekleyin.

3 Kod parçacığı örneğinin içeriklerini `<mx:Script>` ve `</mx:Script>` etiketler arasına yapıştırın. MXML dosyasını kaydedin.

4 Örneği çalıştırmak üzere ana MXML dosyası için Çalıştır > Çalıştır menü seçeneği Ana Dosya (Çalıştır > Çalıştır ContextMenuExample gibi) seçeneklerini belirleyin.

Sınıf bazlı örneğin Flash Builder'da çalıştırılması

Sınıf bazlı örneği Flash Builder'da çalıştırma:

1 Dosya Seçin > Yeni > ActionScript Projesi seçimlerini belirleyin.

2 Birincil sınıfın (ContextMenuExample gibi) adını Proje Adı alanına yazın. Diğer alanlar için varsayılan değerleri kullanın (veya kendi özel ortamınıza göre değiştirin). Projeyi ve ana ActionScript dosyasını oluşturmak için Son'u tıklatın.

3 Oluşturulmuş herhangi bir içeriği ActionScript dosyasından silin. Paket ve içe aktarma ifadelerini içeren örnek kodu ActionScript dosyasına yapıştırın ve dosyayı kaydedin.

Not: *flashx.textLayout.container.ContainerController sınıfı örneği gibi bazı sınıf temelli örnekler, paket bildiriminde birden fazla düzey içerir (`package flashx.textLayout.container.examples {`). Bu örnekler için, dosyayı paket bildirimiyle eşleşen bir alt dosyaya kaydedin (flashx/textLayout/container/examples) veya paket adını kaldırın (böylece ActionScript yalnızca `package {` ile başlar) ve dosyayı herhangi bir konumdan test edebilirsiniz.*

4 Örneği çalıştırmak üzere, ana ActionScript sınıf adı için (Çalıştır > Çalıştır menüsü ContextMenuExample) Çalıştır > Çalıştır menüsü seçeneğini belirleyin.

Flash Builder'da pratik bir örnek çalıştırma

Pratik örnekler genellikle ZIP arşiv dosyaları olarak verilir. Flash Builder kullanarak pratik bir örnek çalıştırma:

1 Zip arşiv dosyasını seçtiğiniz bir klasöre ayıklayın. Klasöre açıklayıcı bir isim verin (ContextMenuExample gibi).

2 Flash Builder'da, Dosya > Yeni Flex Projesi seçeneklerini belirleyin. Proje Konumu bölümünde Gözat'ı tıklatın ve örnek dosyaların bulunduğu klasörü seçin. Proje Adı alanına dosya adı yazın (ContextMenuExample gibi). Diğer alanlar için varsayılan değerleri kullanın (veya kendi özel ortamınıza göre değiştirin). Devam etmek için İleri'yi tıklatın.

3 Çıktı panelinde varsayılan değeri kabul etmek için İleri'yi tıklatın.

4 Kaynak Yolları'nda Ana Uygulama Dosya alanına gitmek için Gözat düğmesine tıklatın. Örnek klasöründen ana MXML örneğini seçin. Projeyi dosyalarını oluşturmak için Son'u tıklatın.

5 Örneği çalıştırmak üzere ana MXML dosyası için Çalıştır > Çalıştır menü seçeneği Ana Dosya (Çalıştır > Çalıştır ContextMenuExample gibi) seçeneklerini belirleyin.

ActionScript 3.0 örneklerinin taşınabilir cihazlarda çalıştırılması

ActionScript 3.0 kod örneklerini Flash Player 10.1'i destekleyen taşınabilir cihazlarda çalıştırabilirsiniz. Ancak belirli sınıfların ve yöntemlerin nasıl çalıştığını öğrenmek için genellikle bir kod örneği çalıştırmanız gerekir. Bu durumda örneği, masaüstü bilgisayar gibi taşınamaz cihazlarda çalıştırın. Bir kod örneğini daha iyi anlamak için, Masaüstü bilgisayarda Flash Professional veya Flash Builder'da izleme ifadelerini ve diğer hata ayıklama araçlarını kullanabilirsiniz.

Örneği taşınabilir bir cihazda çalıştırmak istiyorsanız, dosyaları cihaza veya web sunucusuna kopyalayabilirsiniz. Dosyaları cihaza kopyalamak ve örneği tarayıcıda çalıştırmak için aşağıdakileri uygulayın:

1 SWF dosyasını oluşturmak için“ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması” sayfa 1110 veya “ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması” sayfa 1111'ndaki talimatları uygulayın. Flash Professional’da SWF dosyası oluşturmak için Kontrol Et > Filmi Test Et seçeneklerini belirleyin. Flash Builder'da SWF oluşturmak için çalıştır, ayıkla işlemlerini yapmanız veya Flash Builder projenizi oluşturmanız gerekir.

2 SWF dosyanızı taşınabilir cihazdaki dizine kopyalayın. Dosyayı kopyalamak için cihazla birlikte verilen yazılımı kullanın.

3 Taşınabilir cihazınızdaki tarayıcının adres çubuğuna SWF dosyası için file:// URL yazın. Örneğin, `file:://applications/myExample.swf` yazın.

Dosyaları cihaza kopyalamak ve örneği tarayıcıda çalıştırmak için aşağıdakini uygulayın:

1 SWF dosyası ve HTML dosyası oluşturun. İlk önce, “ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması” sayfa 1110 veya “ActionScript 3.0 örneklerinin Flash Professional'da Çalıştırılması” sayfa 1111'ndaki talimatları uygulayın. Flash Professional’da, Kontrol Et > Filmi Test Et seçeneklerinin belirlenmesi sadece SWF dosyasını oluşturur. Her iki dosyayı oluşturmak için Yayınlama Ayarları iletişim kutusundaki Biçimler sekmesinde hem Flash'ı hem de HTML'yi seçin. Sonra HTML ve SWF dosyalarını oluşturmak için Dosya> Yayınla seçeneklerini belirleyin. Flash Builder'da SWF oluşturmak için çalıştır, ayıkla işlemlerini yapmanız veya Flash Builder projenizi oluşturmanız gerekir.

2 SWF ve HTML dosyasını oluşturmak için web sunucusundaki dizine kopyalayın.

3 Taşınabilir cihazdaki tarayıcının adres çubuğuna HTML dosyası için HTTP adresi yazın. Örneğin `http://www.myWebServer/examples/myExample.html` yazın.

Örneği taşınabilir bir cihazda çalıştırmadan önce aşağıdaki noktaları göz önünde bulundurun.

Sahne alanı boyutu

Bir örneği mobil bir cihazda çalıştırırken kullandığınız sahne alanı, taşıanamayan bir cihaz kullandığınızdakinden daha küçüktür. Örneklerin çoğu için belirli bir Sahne alanı gerekmez. SWF dosyasını oluştururken cihazınız için uygun bir Sahne alanı boyutu belirleyin. Örneğin 176 x 208 piksel olarak belirleyin.

ActionScript 3.0 Geliştirme Kılavuzundaki pratik örnekler, farklı ActionScript 3.0 kavramları ve sınıflarını göstermek amacıyla verilmiştir. Kullanıcı arabirimleri, iyi bir görünüm sergilemek ve masaüstü veya laptop bilgisayarda rahat çalışabilmek amacıyla geliştirilmiştir. Örneklerin taşınabilir cihazlarda kullanılabiliyor olmasına rağmen Sahne alanı ve kullanıcı arabirim tasarımı küçük ekran için uygun değildir. Adobe, ActionScript'i öğrenmek için pratik örnekleri bilgisayarınızda çalıştırmanızı ve daha sonra ilgili kod parçacıklarını mobil uygulamalarda kullanmanızı tavsiye eder.

İzleme ifadeleri yerine metin alanları

Taşınabilir bir cihazda bir örnek çalıştırırken çıktıyı, örneğin izleme ifadesinden göremezsiniz. Çıktıyı görmek için TextField sınıfı örneği oluşturun. Sonra metni, izleme ifadelerinden metin alanının `text` özelliğine iliştirin.

İzleme için kullanmak amacıyla metin alanı kurmak için aşağıdaki işlevi kullanın:

```
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;
}
```

Örneğin bu fonksiyonu belge sınıfına özel işlev olarak ekleyin. Daha sonra belge sınıfı yöntemlerinde verileri kodla birlikte aşağıdaki gibi izleyin.

```
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("");
```

`appendText()` yöntemi parametre olarak yalnızca bir değer kabul eder. Bu değerin bir dize (bir String örneği veya değişmez bir dize) olması gerekir. Dize olmayan bir değişkenin değerini yazdırmak için öncelikle değeri bir String öğesine dönüştürün. Bunu yapmanın en kolay yolu, nesnenin `toString()` yönteminin çağrılmasıdır:

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

Metin boyutu

Çoğu örnek, bir kavramın gösterilmesi için metin alanları kullanır. Bazen metin boyutunun metin alanında ayarlanması, taşınabilir bir cihazda daha iyi okunabilirlik sağlar. Mesela bir örnek `myTextField` isimli bir metin alanı örneği kullanırsa, metnin boyutunu aşağıdaki koda göre değiştirin.

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

Kullanıcı girdisini yakalama

Mobil kullanım sistemi ve tarayıcı, SWF içeriğinin kabul etmediği bazı kullanıcı girdi olaylarını yakalar. İşletim sistemi ve tarayıcıya dayanan belirli davranışlar vardır ama örnekleri taşınabilir bir cihazda çalıştırdığınızda beklenmeyen bir davranışla sonuçlanabilir. Ayrıntılı bilgi için bkz. “KeyboardEvent önceliği” sayfa 567.

Ayrıca, masaüstü veya dizüstü bilgisayar için birçok örneğin kullanıcı arabirimi tasarlanmıştır. Örneğin, ActionScript 3.0 Geliştirici Kılavuzu'ndaki pratik örneklerin çoğu masaüstü izleme için oldukça uygundur. Bu nedenle Sahne alanının tamamı taşınabilir cihazın ekranından görünmeyebilir. Tarayıcınızın içindekilerin yatay kaydırma yapabilmesi, tarayıcınıza bağlıdır. Ayrıca örnekler, dikey ve yatay kaydırma olaylarını yakalamak ve işlemek amacıyla geliştirilmemiştir. Bu nedenle bazı örneklerin kullanıcı arabirimleri, küçük ekranlarda çalıştırmak için uygun değildir. Adobe, ActionScript'i öğrenmek için pratik örnekleri bilgisayarınızda çalıştırmanızı ve daha sonra ilgili kod parçacıklarını mobil uygulamalarda kullanmanızı tavsiye eder.

Ayrıntılı bilgi için bkz. “Görüntüleme nesnelerini yatay ve dikey kaydırma” sayfa 181.

Odak kullanma

Bazı örneklerde alana odak vermeniz gerekir. Alana odak vererek, metin girme veya bir düğme seçme gibi işlemler yapabilirsiniz. Alan odağı vermek için taşınabilir cihazın işaretleyicisini ekran kalemi ya da parmağınız gibi kullanın. Veya, alan odağı vermek için taşınabilir cihazınızın gezinme tuşlarını kullanın. Odak olan bir düğme seçmek için bilgisayarda Enter kullandığınız gibi taşınabilir cihazın Seç tuşunu kullanın. Bazı cihazlarda bir düğmeye hafifçe dokunarak da seçim yapabilirsiniz.

Ayrıntılı bilgi için bkz. “Odak yönetme” sayfa 563.

Fare olaylarını kullanma

Birçok örnek fare olaylarını dinler. Bu fare olayları bir bilgisayarda; bir kullanıcının fareyi, görüntü nesnesinin üzerine getirmesi veya fare düğmesini görüntü nesnesinin üzerine tıklatması örneklerindeki gibi gerçekleşebilir. Taşınabilir cihazlarda ekran kalemi veya parmak gibi işaretleyici araçları kullanma olaylarına "dokunma olayları" denir. Flash Player 10.1 dokunma olaylarını fare olaylarına eşlemler. Bu eşleme, SWF içeriğinin Flash Player 10.1 çalışmaya devam etmeden önce geliştirilmesini sağlar. Bu nedenle örnekler, bir görüntü nesnesi seçmek veya sürüklemek için işaretleyici aracını kullanırken çalışır.

Performans

Taşınabilir cihazların işleme gücü masaüstü cihazlarınkinden daha azdır. CPU yoğunluğu olan bazı örnekler taşınabilir cihazlarda daha yavaş çalışabilir. Mesela, "API çizme örneği: Algoritmik Görsel Oluşturucu" sayfa 236 öğesindeki örnek, her karede geniş çaplı hesaplamalar ve çizimler yapar. Bu örneği bir bilgisayarda çalıştırılması, çeşitli API çizimleri gösterir. Ancak, performanstaki sınırlılıklar nedeniyle örnek, bazı taşınabilir cihazlara uygun değildir.

Mobil aygıtlardaki performansla ilgili daha fazla bilgi için bkz. *Flash Platform için Performansı En iyileştirme*.

En iyi uygulamalar

Örnekler, taşınabilir cihazlarda uygulama geliştirmedeki en iyi uygulamaları göz önünde bulundurmaz. Taşınabilir cihazlarda bellek ve işlem gücüyle ilgili sınırlılıklar için özel bir değerlendirme gerekir. Aynı şekilde, küçük ekranının kullanıcı arabirimi gereksinimleri, masaüstü ekranınkinden daha farklıdır. Mobil aygıtlar için uygulama geliştirmeyle ilgili daha fazla bilgi için *bkz. Flash Platform için Performansı En iyileştirme*.

Bölüm 66: yerel veritabanlarında SQL desteği

Adobe AIR uygulamasına, açık kaynaklı SQLite veritabanı sistemini kullanan, birçok standart SQL özelliğine sahip yerel SQL veritabanları için desteğin bulunduğu bir SQL veritabanı motoru dahildir. Çalışma zamanı, veritabanı verilerinin dosya sisteminde nerede ve nasıl saklandığını belirtmez. Her veritabanının tamamı tek bir dosyada saklanır. Geliştirici, dosya sisteminin içinde veritabanı dosyasının saklanacağı konumu belirtebilir ve tek bir AIR uygulaması, bir veya birden fazla ayrı veritabanına erişebilir (örn. ayrı veritabanı dosyaları). Bu belge, Adobe AIR yerel SQL veri tabanları için SQL söz dizimi ve veri türü desteğinin genel bir açıklamasını sunar. Bu belge, kapsamlı bir SQL başvuru kaynağı görevi yapacak şekidle tasarlanmamıştır. Daha çok, Adobe AIR'nin desteklediği SQL lehçesine ilişkin belirli ayrıntıları açıklar. Çalışma zamanı, SQL-92 standart SQL lehçesinin çoğunu destekler. SQL öğrenmek için çok fazla sayıda başvuru, web sitesi, kitap ve eğitim malzemesi bulunduğundan, bu belge kapsamlı bir SQL başvuru kaynağı veya eğitim malzemesi olacak şekilde tasarlanmamıştır. Bunun yerine, bu belge, özellikle AIR destekli SQL söz dizimi ve desteklenen SQL lehçesiyle SQL-92 arasındaki farklılıklar üzerinde durmaktadır.

SQL ifadesi tanımı kuralları

Bu belgedeki ifade tanımlarının içinde şu kurallar kullanılmaktadır:

- Metin büyük/küçük harf durumu
  - BÜYÜK HARF - değişmez SQL anahtar sözcüklerinin tümü büyük harfle yazılır.
  - küçük harf - yer tutucu terimleri veya deyim adlarının tümü küçük harfle yazılır.
- Tanım karakterleri
  - ::= Bir deyim veya ifade tanımını gösterir.
- Gruplama ve değiştirme karakterleri
  - | Alternatif seçenekler arasında çubuk karakteri kullanılır ve bu işaret "veya" olarak okunabilir.
  - [] Köşeli ayraç içindeki öğeler isteğe bağlı öğelerdir; köşeli ayraçlar tek bir öğe veya bir alternatif öğe kümesi içerebilir.
  - () Alternatif öğe kümesini çevreleyen parantezler (çubuk karakterleri ile ayrılan öğeler kümesi), gerekli bir öğe grubunu; diğer bir deyişle, gerekli tek bir öğe için olası değerlerin olduğu öğeler kümesini atar.
- Nicelik belirteçleri
  - + Parantez içindeki bir öğeden sonra gelen bir artı karakteri, önceki öğenin 1 veya daha fazla kez görülebileceğini gösterir.
  - * Köşeli ayraç içindeki bir öğeden sonra gelen bir yıldız karakteri, önceki öğenin (köşeli ayraç içindeki) 0 veya daha fazla kez görülebileceğini gösterir.
- Değişmez karakterler
  - * Bir sütun adında veya işlev adını izleyen parantezler içinde kullanılan yıldız karakteri "0 veya daha fazla" nicelik belirteci yerine değişmez bir yıldız karakterini gösterir.
  - . Nokta karakteri değişmez bir noktayı temsil eder.
  - , Virgül karakteri değişmez bir virgülü temsil eder.
  - () Tek bir deyimi veya öğeyi çevreleyen parantez çifti, gerekli parantezlerin değişmez parantez karakterleri olduğunu gösterir.

- Diğer karakterler, aksi belirtilmediği takdirde bu değişmez karakterleri temsil eder.

Desteklenen SQL söz dizimi

Aşağıda listelenen SQL söz dizimleri Adobe AIR SQL veritabanı motoru tarafından desteklenmektedir. Listeler, farklı ifade ve deyim türlerinin, ifadelerin, yerleşik işlevlerin ve operatörlerin açıklamalarına bölünmüştür. Şu başlıklar işlenmektedir:

- Genel SQL söz dizimi
- Veri işleme ifadeleri (SELECT, INSERT, UPDATE, and DELETE)
- Veri tanımı ifadeleri (tablolar, görünümler, dizinler ve tetikleyiciler için CREATE, ALTER ve DROP ifadeleri)
- Özel ifadeler ve deyimler
- Yerleşik işlevler (Toplama, ölçekleme ve tarih/saat biçimlendirme işlevleri)
- Operatörler
- Parametreler
- Desteklenmeyen SQL özellikleri
- Ek SQL özellikleri

Genel SQL söz dizimi

Çeşitli deyimler ve ifadeler için belirli sözdizimine ek olarak, aşağıda SQL sözdizimi için genel kuralları bulabilirsiniz:

Büyük/küçük harf duyarlılığı Nesne adları da dahil olmak üzere SQL ifadeleri, büyük küçük harf duyarlı değildir. Yine de, SQL ifadeleri genellikle, büyük harfle yazılan SQL anahtar sözcükleri ile yazılır ve bu belge bu kuralı kullanır. SQL söz dizimi büyük/küçük harf duyarlı değilken, SQL'deki değişmez metin değerleri büyük/küçük harf duyarlıdır; bir sütun veya işlem için tanımlanan harmanlama sırası tarafından belirlendiği şekilde karşılaştırma ve sıralama işlemleri de büyük/küçük harf duyarlı olabilir. Daha fazla bilgi için COLLATE ifadesine bakın.

Beyaz boşluk SQL ifadesinde ayrı sözcükleri birbirinden ayırmak için bir beyaz boşluk karakterinin (boşluk, sekme, yeni satır vb.) kullanılması gerekir. Ancak, sözcükler ve semboller arasındaki beyaz boşluk isteğe bağlıdır. SQL ifadesindeki beyaz boşluk karakterlerinin türü ve sayısı bir öneme sahip değildir. İfadenin anlamını etkilemeden, daha kolay okunabilmeleri için SQL ifadelerinizi biçimlendirmede girinti ve satır kesmeleri gibi beyaz boşluklar kullanabilirsiniz.

Veri işleme ifadeleri

Veri işleme ifadeleri en yaygın kullanılan SQL ifadeleridir. Bu ifadeler, veritabanı tablolarına veri eklemek, tablolardan veri almak, değiştirmek ve kaldırmak için kullanılır. Şu veri işleme ifadeleri desteklenmektedir: SELECT, INSERT, UPDATE ve DELETE.

SELECT

SELECT ifadesi veritabanında sorgu yürütmek için kullanılır. SELECT ifadesinin sonucu sıfır veya her satırın sabit sayıda sütuna sahip olduğu daha fazla veri satırıdır. Sonuçtaki sütun sayısı, sonuç sütun adı veya SELECT ve isteğe bağlı FROM anahtar sözcükleri arasındaki ifade listesi tarafından belirlenir.

```
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
```

Sonuç olarak rastgele bir ifade kullanılabilir. Sonuç ifadesi * ise, tüm tabloların tüm sütunları bu tek ifade için değiştirilir. İfade, ardından* gelen bir tablo adıysa, sonuç bu tek tablodaki tüm sütunlar olur.

DISTINCT anahtar sözcüğü, her sonuç satırının farklı olduğu sonuç satırlarının bir alt kümesinin döndürülmesine neden olur. NULL değerleri birbirinden farklı olarak işlenmez. Varsayılan davranış şekli, tüm sonuç satırlarının döndürülmesidir ve bu da ALL anahtar sözcüğüyle belirgin hale getirilebilir.

Sorgu, FROM anahtar sözcüğünden sonra belirtilen bir veya birden fazla tabloda çalıştırılır. Birden fazla tablo adı virgülle ayrılırsa, sorgu birden çok tablonun çapraz birleştirmesini kullanır. JOIN sözdizimi, tabloların nasıl birleştirileceğini belirlemek için de kullanılabilir. Desteklenen tek dış birleştirme tipi LEFT OUTER JOIN'dir. join-args içindeki ON deyim ifadesi bir boolean değerine çözümlenmelidir. Parantez içindeki bir alt sorgu FROM deyiminde bir tablo olarak kullanılabilir. FROM deyiminin tamamı atılabilir; bu durumda sonuç, sonuçta elde edilen ifade listesinin değerlerinden oluşan tek bir satır olur.

WHERE deyimi, sorgunun getirdiği satır sayısını sınırlamak için kullanılır. WHERE deyimi ifadeleri bir boolean değerine çözümlenmelidir. WHERE deyimi filtreleme işlemi, herhangi bir gruplama işleminden önce gerçekleştirilir; böylece, WHERE deyimi ifadeleri toplu işlevleri içermeyebilir.

GROUP BY deyimi, sonucun bir veya birden fazla satırının tek bir çıktı satırında birleştirilmesini sağlar. GROUP BY deyimi özellikle, sonuç toplu işlevler içerdiğinde faydalıdır. GROUP BY deyimindeki ifadelerin, SELECT ifade listesinde görülen ifadeler olmasına gerek yoktur.

HAVING deyimi, ifade tarafından döndürülen satırları sınırlandırmada WHERE deyimine benzer. Ancak, HAVING deyimi, GROUP BY deyimi tarafından belirtilen bir gruplamadan sonra uygulanır. Sonuç olarak, HAVING ifadesi toplu işlevleri içeren değerlere başvuru yapabilir. HAVING deyimi ifadesinin, SELECT listesinde görülmesine gerek yoktur. WHERE ifadesi gibi, HAVING ifadesi de bir boolean değerine çözümlenmelidir.

ORDER BY deyimi, çıktı sıralarının sıralanmasını sağlar. ORDER BY deyimindeki sort-expr-list argümanı sıralama için anahtar olarak kullanılan ifadelerin bir listesidir. İfadelerin, basit SELECT deyiminin bir parçası olmasına gerek yoktur; ancak, bileşik bir SELECT deyiminde (compound-op operatörlerinden birini kullanan SELECT deyimi), her sıralama ifadesinin sonuç sütunlarından biriyle tam olarak eşleşmesi gerekir. Her sıralama ifadesinden sonra isteğe bağlı olarak, COLLATE anahtar sözcüğü ve sıralama metni için kullanılan harmanlama işlevinin adını ve/veya sıralama düzenini (artan veya azalan) belirlemek için ASC veya DESC anahtar sözcüklerini içeren sort-order deyimi gelebilir. sort-order deyimi atılabilir ve varsayılan (artan düzen) kullanılabilir. COLLATE deyiminin ve harmanlama işlevlerinin tanımı için bkz. COLLATE.

LIMIT deyimi, sonuçta döndürülen satır sayısına bir üst sınır koyar. Eksi LIMIT üst sınır olmadığını gösterir. LIMIT deyiminden sonra gelen isteğe bağlı OFFSET anahtar sözcüğü, sonuç kümesinin başında kaç satırın atlanacağını belirler. Bileşik SELECT sorgusunda, LIMIT deyimi yalnızca son SELECT ifadesinden sonra gelebilir ve sınırlama tüm sorguya uygulanır. LIMIT deyiminde OFFSET anahtar sözcüğü kullanılırsa, sınırın ilk tam sayı olduğuna ve uzaklığın ikinci tam sayı olduğuna dikkat edin. OFFSET anahtar sözcüğü yerine virgül kullanılırsa, uzaklık ilk sayı ve sınır ise ikinci sayı olur. Ortaya çıkan bu çelişki kasıtlıdır. Eski SQL veritabanı sistemleriyle uyumluluğu artırır.

Bileşik SELECT deyimi, bir veya birden fazla UNION, UNION ALL, INTERSECT veya EXCEPT operatörü tarafından birbirine bağlanan bir veya birden fazla SELECT ifadesinden oluşur. Bileşik SELECT deyiminde, deyimi oluşturan tüm SELECT ifadelerinin aynı sayıda sonuç sütununu belirtmesi gerekir. Son SELECT ifadesinden sonra (bir adet belirtilmişse, tek LIMIT deyiminden önce) yalnızca bir ORDER BY deyimi olabilir. UNION ve UNION ALL operatörleri, önceki ve sonraki SELECT ifadelerinin sonuçlarını tek bir tabloda birleştirir. Aradaki fark, UNION operatöründe, tüm sonuç satırları birbirinden farklıdır ancak, UNION ALL operatöründe yinelemeler olabilir. INTERSECT operatörü, önceki ve sonraki SELECT ifadelerinin sonuçlarının kesişimini alır. EXCEPT, önceki SELECT ifadesinin sonucunu sonraki SELECT ifadesinin sonuçlarını kaldırarak alır. Üç veya daha fazla SELECT ifadesi bir bileşik ifadede bağlanırsa, ilkinden sonuncuya doğru gruplanırlar.

İzin verilen ifadelerin tanımı için bkz. İfadeler.

AIR 2.5'ten itibaren, BLOB verisini ActionScript ByteArray nesnelerine dönüştürmek için okuma işlemi gerçekleştirilirken SQL CAST operatörü desteklenir. Örneğin, aşağıdaki AMF biçiminde saklanmayan işlenmemiş veri okur ve bunu bir ByteArray nesnesinde saklar:

```
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

INSERT ifadesi, iki temel biçimde gelir ve tabloları veriyle doldurmak için kullanılır.

```
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
```

İlk biçim, (VALUES anahtar sözcüğüyle) varolan tabloda tek bir yeni satır oluşturur. Hiçbir sütun listesi belirtilmezse, değer sayısının tablodaki sütun sayısıyla aynı olması gerekir. Bir sütun listesi belirtilirse, değer sayısının belirtilen sütun sayısıyla eşleşmesi gerekir. Sütun listesinde görülmeyen tablo sütunları tablo oluşturulduğunda tanımlanan varsayılan değerle veya hiçbir değer tanımlanmamışsa NULL değeriyle doldurulur.

İkinci INSERT ifadesi biçimi, verilerini SELECT ifadesinden alır. SELECT ifadesinin sonucundaki sütun sayısının column-list belirtilmemişse tablodaki sütun sayısıyla tam olarak eşleşmesi veya column-list altında belirtilen sütun sayısıyla eşleşmesi gerekir. SELECT sonucunun her satırı için tabloda yeni bir giriş oluşturulur. SELECT ifadesi basit veya bileşik olabilir. İzin verilen SELECT ifadelerinin tanımı için bkz. SELECT.

İsteğe bağlı çakışma algoritması, bu tek komut sırasında kullanılacak alternatif sınırlama çakışması çözümü algoritmasının belirtilmesine olanak tanır. Çakışma algoritmalarının açıklaması ve tanımlaması için bkz. "Özel ifadeler ve deyimler" sayfa 1126.

İfadenin iki REPLACE INTO biçimi, standart INSERT [OR çakışma algoritması] biçiminin REPLACE çakışma algoritmasıyla (örneğin, INSERT OR REPLACE... biçimi) kullanımına benzer.

İfadenin iki REPLACE INTO biçimi, standart INSERT [OR çakışma algoritması] biçiminin REPLACE çakışma algoritmasıyla (örneğin, INSERT OR REPLACE... biçimi) kullanımına benzer.

UPDATE

Güncelle komutu bir tabloda bulunan kayıtları değiştirir.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,...
[WHERE expr]
```

Komut UPDATE anahtar sözcüğünden ve bunu takip eden kayıtları güncellemek istediğiniz tablonun adından oluşur. SET anahtar sözcüğünden sonra, sütunun adını ve virgülle ayrılmış liste olarak sütunun değiştirileceği değeri sağlayın. WHERE deyimi ifadesi kayıtların güncellendiği satırı veya satırları sağlar.

DELETE

Sil komutu bir tablodan kayıtları kaldırmak için kullanılır.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

Komut, DELETE FROM anahtar sözcükleri ile bunları takip eden kayıtların kaldırılacağı tablonun adından oluşur.

WHERE deyimi olmadan, tablonun tüm satırları kaldırılır. WHERE deyimi varsa, yalnızca ifadeyle eşleşen satırlar kaldırılır. WHERE deyimi ifadesi bir boolean değerine çözümlenmelidir. İzin verilen ifadelerin tanımı için bkz. İfadeler.

Veri tanımı ifadeleri

Veri tanımı ifadeleri; tablolar, görünümler, dizinler ve tetikleyiciler gibi veritabanı nesnelerini oluşturmak, değiştirmek ve kaldırmak için kullanılır. Aşağıdaki veri tanımı ifadeleri desteklenir:

- Tablolar:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
- Dizinler:
  - CREATE INDEX
  - DROP INDEX
- Görünümler:
  - CREATE VIEWS
  - DROP VIEWS
- Tetikleyiciler:
  - CREATE TRIGGERS
  - DROP TRIGGERS

CREATE TABLE

CREATE TABLE ifadesi, CREATE TABLE anahtar sözcüklerini izleyen yeni tablonun adı ile (parantez içinde) sütun tanımları ve sınırlamaları listesinden oluşur. Tablo adı bir tanımlayıcı veya bir dize olabilir.

```
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]*
```

Her sütun tanımı, sütun adını izleyen bu sütunun veri türü ve bir veya birden fazla isteğe bağlı sütun sınırlamasından oluşur. Sütunun veri türü bu sütunda hangi verilerin saklanabileceğini sınırlandırır. Bir sütunda, farklı veri türünde bir değer saklanmaya çalışılması durumunda, çalışma zamanı, mümkünse değeri uygun türe dönüştürür veya bir hata döndürür. Ek bilgi için Veri türü desteği bölümüne bakın.

NOT NULL sütun sınırlaması, sütunun NULL değerleri içeremeyeceğini gösterir.

UNIQUE sınırlaması, belirtilen sütun veya sütunlarda bir dizinin oluşturulmasını sağlar. Bu dizin benzersiz anahtarlar içermelidir. İki satır, yinelenen değerleri veya belirtilen sütun veya sütunların değerler birleşimini içeremez. CREATE TABLE ifadesi, sütunun tanımındaki UNIQUE sınırlaması da dahil olmak üzere birden fazla UNIQUE sınırlaması ve/veya çoklu tablo düzeyindeki UNIQUE sınırlamaları içerebilir.

CHECK sınırlaması, değerlendirilen bir ifadeyi tanımlar ve bir satırın verilerinin eklenmesi veya güncellenmesi için true değerinde olmalıdır. CHECK ifadesi bir boolean değerine çözümlenmelidir.

Sütun tanımındaki COLLATE deyimi, sütun için metin girişlerini karşılaştırırken hangi harmanlama işlevinin kullanılacağını tanımlar. Varsayılan olarak BINARY harmanlama işlevi kullanılır. COLLATE deyiminin ve harmanlama işlevlerinin ayrıntıları için bkz. COLLATE.

DEFAULT sınırlaması INSERT ifadesi çalıştırılırken kullanılan varsayılan değeri belirtir. Değer NULL, bir dize sabiti veya bir sayı olabilir. Varsayılan değer CURRENT_TIME, CURRENT_DATE veya CURRENT_TIMESTAMP gibi özel büyük/küçük harften bağımsız anahtar sözcükler de olabilir. Değer NULL, bir dize sabiri veya bir sayı ise INSERT ifadesi sütun için bir değer belirtmediğinde sütuna eklenir. Değer CURRENT_TIME, CURRENT_DATE veya CURRENT_TIMESTAMP ise, geçerli UTC tarihi ve/veya saati sütuna eklenir. CURRENT_TIME için biçim şu şekildedir: SS:DD:SS. CURRENT_DATE için biçim şu şekildedir: YYYY-AA-GG. CURRENT_TIMESTAMP için biçim şu şekildedir: YYYY-AA-GG SS:DD:SS.

Bir PRIMARY KEY belirlenmesi genelde yalnızca ilgili sütun veya sütunlarda bir UNIQUE dizini oluşturur. Ancak, PRIMARY KEY sınırlaması INTEGER veri türündeki (veya int gibi bir eşanlamlı ifade) tek bir sütun üzerindeyse, bu sütun tablo için gerçek ana anahtar olarak dahili şekilde kullanılır. Bu durum, sütunun yalnızca benzersiz tam sayı değerlerini tutabileceği anlamına gelir. (Çoğu SQLite uygulamasında yalnızca sütun türü INTEGER sütunun dahili birincil anahtar olarak işlev yapmasına neden olur, ancak Adobe AIR uygulamasında int gibi INTEGER ile eşanlamlı ifadeler de bu davranışı belirler.)

Bir tabloda INTEGER PRIMARY KEY sütunu yoksa, satır eklendiğinde otomatik olarak bir tam sayı anahtarı oluşturulur. Satır için ana anahtara her zaman, ROWID, OID veya _ROWID_ özel adlarından biri kullanılarak erişilebilir. Bu adlar, açıkça belirtilen bir INTEGER PRIMARY KEY veya dahili şekilde oluşturulan bir değer olup olmadığına bakılmaksızın kullanılabilir. Ancak tabloda açık bir INTEGER PRIMARY KEY varsa, sonuç verilerindeki sütunun adı özel ad yerine gerçek sütun adı olur.

INTEGER PRIMARY KEY sütunu, AUTOINCREMENT anahtar sözcüğünü de içerebilir. AUTOINCREMENT anahtar sözcüğü kullanıldığında, veritabanı ardışık olarak artırılan bir tam sayı anahtarını otomatik olarak oluşturur ve sütun için tam değer belirtmeyen bir INSERT ifadesi çalıştırılırken INTEGER PRIMARY KEY sütununa bu anahtarı ekler.

Bir CREATE TABLE ifadesinde yalnızca bir PRIMARY KEY sınırlaması olabilir. Bu ifade, bir sütun tanımının veya tek tablo düzeyindeki PRIMARY KEY sınırlamasının bir parçası olabilir. Bir ana anahtar sütunu üstü kapalı bir şekilsw NOT NULL değerine sahiptir.

Birçok sınırlamayı izleyen isteğe bağlı çakışma deyimi, bu sınırlama için alternatif bir varsayılan sınırlama çakışma çözümleme algoritmasının belirlenmesine olanak tanır. Varsayılan, ABORT şeklindedir. Aynı tablodaki farklı sınırlamalar farklı varsayılan çakışma çözümleme algoritmalarına sahip olabilir. INSERT veya UPDATE ifadesi farklı bir çakışma çözümleme algoritması belirtirse, bu algoritma, CREATE TABLE ifadesinde belirtilen algoritmanın yerine kullanılır. Daha fazla bilgi için "Özel ifadeler ve deyimler" sayfa 1126 kısmının ON CONFLICT bölümüne bakın.

FOREIGN KEY sınırlamaları gibi ek sınırlamalar, bir hataya yol açmaz, ancak çalışma zamanı bunları yok sayar.

TEMP veya TEMPORARY anahtar sözcüğü CREATE ve TABLE deyimlerinin arasına gelirse, oluşturulan tablo yalnızca aynı veritabanı bağlantısı içinde görünür (SQLConnection örneği). Veritabanı bağlantısı kapatıldığında otomatik olarak silinir. Geçici tabloda oluşturulan dizinler de geçicidir. Geçici tablolar ve dizinler ana veritabanı dosyasından farklı ayrı bir dosyada saklanır.

İsteğe bağlı database-name öneki belirtilirse, tablo adlı veritabanında oluşturulur (attach() yöntemini belirtilen veritabanı adıyla çağırarak SQLConnection örneğine bağlanan bir veritabanı). database-name öneki temp değilse, hem veritabanı adı önekinin hem de TEMP anahtar sözcüğünün belirtilmesi bir hatadır. Hiçbir veritabanı adı belirtilmezse ve TEMP anahtar sözcüğü yoksa, tablo ana veritabanında oluşturulur (SQLConnection örneğine open() veya openAsync() yöntemini kullanarak bağlanmış olan veritabanı).

Sütun sayısı veya tablodaki sınırlama sayısı üzerinde hiçbir rastgele sınır yoktur. Bir satırdaki veri miktarı üzerinde de hiçbir rastgele sınır yoktur.

CREATE TABLE AS biçimi, sonuç kümesinin sonucu olarak tabloyu tanımlar. Tablo sütunlarının adı, sonuçtaki sütunların adlarıdır.

İsteğe bağlı IF NOT EXISTS deyimi varsa ve aynı adı taşıyan başka bir tablo önceden varsa, veritabanı CREATE TABLE komutunu yok sayar.

Bir tablo DROP TABLE ifadesi kullanılarak kaldırılabilir ve ALTER TABLE ifadesi kullanılarak sınırlı değişiklikler yapılabilir.

ALTER TABLE

ALTER TABLE komutu kullanıcının varolan bir tabloyu yeniden adlandırmasını veya yeni bir sütun eklemesini sağlar. Bir tablodan sütun kaldırılamaz.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

RENAME TO sözdizimi, [database-name.] table-name tarafından tanımlanan tabloyu new-table-name olarak yeniden adlandırmak için kullanılır. Bu komut, bağlı veritabanları arasında bir tabloyu taşımak için kullanılmaz; yalnızca aynı veritabanındaki bir tabloyu yeniden adlandırmak için kullanılır.

Yeniden adlandırılan tabloda tetikleyiciler ve dizinler varsa, yeniden adlandırıldıktan sonra tabloya ekli kalmaya devam ederler. Ancak, yeniden adlandırılan tabloya başvuruda bulunan görünüm tanımları veya tetikleyiciler tarafından çalıştırılan ifadeler varsa, yeni tablo adını kullanmak üzere otomatik olarak değiştirilmezler. Yeniden adlandırılan tabloda ilişkilendirilmiş görünümler veya tetikleyiciler varsa, yeni tablo adını kullanarak tetikleyicileri veya görünüm tanımlarını el ile bırakıp yeniden oluşturmanız gerekir.

ADD [COLUMN] sözdizimi, varolan bir tabloya yeni bir sütun eklemek için kullanılır. Yeni sütun her zaman varolan sütunlar listesinin sonuna eklenir. column-def deyimi aşağıdaki sınırlamalarla birlikte bir CREATE TABLE ifadesinde her biçimi alabilir:

- Sütun PRIMARY KEY veya UNIQUE sınırlamasına sahip olamaz.
- Sütun, CURRENT_TIME, CURRENT_DATE veya CURRENT_TIMESTAMP varsayılan değerine sahip olamaz.
- NOT NULL sınırlaması belirtilirse, sütunun NULL değeri dışında bir varsayılan değere sahip olması gerekir.

ALTER TABLE ifadesinin çalışma zamanı tablodaki veri miktarından etkilenmez.

DROP TABLE

DROP TABLE ifadesi, CREATE TABLE ifadesiyle eklenen bir tabloyu kaldırır. Belirtilen table-name deyimine sahip bir tablo, bırakılan tablodur. Veritabanından ve disk dosyasından tamamıyla kaldırılır. Tablo kurtarılamaz. Tablo ile ilişkilendirilen tüm dizinler de silinir.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

Varsayılan olarak, DROP TABLE ifadesi veritabanı dosyasının boyutunu azaltmaz. Veritabanındaki boş alan tutulur ve sonraki INSERT işlemlerinde kullanılır. Veritabanındaki boş alanı temizlemek için SQLConnection.clean() yöntemini kullanın. Veritabanı ilk oluşturulduğunda, autoClean parametresi true olarak ayarlanırsa, alan otomatik olarak boşaltılır.

İsteğe bağlı IF EXISTS deyimi, tablo yoksa ortaya çıkacak olan hatayı bastırır.

CREATE INDEX

CREATE INDEX komutu, CREATE INDEX anahtar sözcüklerinden sonra gelen yeni dizinin adı, ON anahtar sözcüğü, dizine alınacak daha önce oluşturumuş tablonun adı, dizin anahtarı için değerleri kullanılacak tablodaki sütun adlarının parantezli listesinden oluşur.

```
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]
```

Her sütun adından sonra, sıralama düzenini göstermek için ASC veya DESC anahtar sözcükleri gelebilir; ancak, sıralama düzeni ataması çalışma zamanı tarafından yok sayılır. Sıralama her zaman artan düzende yapılır.

Her sütun adından sonra gelen COLLATE deyimi, bu sütundaki metin değerleri için kullanılan harmanlama dizisini tanımlar. Varsayılan harmanlama sırası, CREATE TABLE ifadesinde bu sütun için tanımlanan harmanlama sırasıdır. Hiçbir harmanlama sırası belirtilmezse, BINARY harmanlama sırası kullanılır. COLLATE deyiminin ve harmanlama işlevlerinin tanımı için bkz. COLLATE.

Tek bir tabloya eklenebilecek dizin sayısı üzerinde hiçbir rastgele sınır yoktur. Bir dizindeki sütun sayısı üzerinde de hiçbir sınır yoktur.

DROP INDEX

DROP INDEX ifadesi, CREATE INDEX ifadesi ile eklenen bir dizini kaldırır. Belirtilen dizin, veritabanı dosyasından tamamıyla kaldırılır. Dizini kurtarmanın tek yolu, uygun CREATE INDEX komutunun yeniden girilmesidir.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

Varsayılan olarak, DROP INDEX ifadesi veritabanı dosyasının boyutunu azaltmaz. Veritabanındaki boş alan tutulur ve sonraki INSERT işlemlerinde kullanılır. Veritabanındaki boş alanı temizlemek için SQLConnection.clean() yöntemini kullanın. Veritabanı ilk oluşturulduğunda, autoClean parametresi true olarak ayarlanırsa, alan otomatik olarak boşaltılır.

CREATE VIEW

CREATE VIEW komutu önceden tanımlı SELECT ifadesine bir ad atar. Bu yeni ad daha sonra, bir tablo adı yerine başka bir SELECT ifadesinin FROM deyiminde kullanılabilir. Görünümler genellikle, karmaşık bir (ve sık kullanılan) veri kümesini diğer işlemlerde kullanılabilecek bir yapıda birleştirerek sorguları kolaylaştırmak için kullanılır.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS
select-statement
```

TEMP veya TEMPORARY anahtar sözcüğü CREATE ve VIEW deyimi arasında bulunursa, oluşturulan görünüm yalnızca, veritabanını açan SQLConnection örneğinde görülür ve veritabanı kapatıldığında otomatik olarak silinir.

Bir [database-name] belirtilirse, görünüm, adlandırılmış veritabanında (Belirtilen name argümanıyla SQLConnection örneğine attach() yöntemi kullanılarak bağlanmış olan veritabanı) oluşturulur. [database-name] öneki temp değilse, hem [database-name] önekinin hem de TEMP anahtar sözcüğünün belirtilmesi bir hatadır. Hiçbir veritabanı adı belirtilmezse ve TEMP anahtar sözcüğü yoksa, görünüm ana veritabanında (SQLConnection örneğine open() veya openAsync() yöntemini kullanarak bağlanmış olan veritabanı) oluşturulur.

Görünümler salt okunurdur. İlişkilendirilmiş türde en az bir tetikleyici (INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE) tanımlanmamışsa bir görünümde, DELETE, INSERT veya UPDATE ifadesi kullanılamaz. Bir görünüm için tetikleyici oluşturmayle ilgili bilgiler için bkz. CREATE TRIGGER.

Bir görünüm, veritabanından DROP VIEW ifadesi kullanılarak kaldırılır.

DROP VIEW

DROP VIEW ifadesi CREATE VIEW ifadesi tarafından oluşturulmuş görünümü kaldırır.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

Belirtilen view-name, bırakılacak görünümün adıdır. Veritabanından kaldırılır; ancak, alttaki tablolardaki hiçbir veri değiştirilmez.

CREATE TRIGGER

CREATE TRIGGER ifadesi, veritabanı şemasına tetikleyiciler eklemek için kullanılır. Tetikleyici, belirtilen veritabanı olayı (database-event) meydana geldiğinde otomatik olarak gerçekleştirilen bir veritabanı işlemidir (trigger-action).

```
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]*
```

Belirli bir veritabanı tablosunda bir DELETE, INSERT veya UPDATE ifadesi çalıştırıldığında veya bir tablonun bir veya birden fazla sütununda bir UPDATE ifadesi güncellendiğinde gerçekleşecek bir tetikleyici belirlenir. TEMP veya TEMPORARY anahtar sözcüğü kullanılmazsa, tetikleyiciler kalıcıdır. Bu durumda tetikleyici SQLConnection örneğinin ana veritabanı bağlantısı kesildiğinde kaldırılır. Hiçbir zamanlama belirtilmezse (BEFORE veya AFTER), tetikleyici varsayılan olarak BEFORE değerine ayarlanır.

Yalnızca FOR EACH ROW tetikleyicileri desteklenir, bu nedenle FOR EACH ROW metni isteğe bağlıdır. WHEN deyimi ifadesi true olarak değerlendirilirse, trigger-step ifadeleri FOR EACH ROW tetikleyicisi ile, tetikleyiciyi çalıştıran ifade tarafından eklenen, güncellenen veya silinen her veritabanı satırı için çalıştırılır.

WHEN deyimi girilirse, trigger-step olarak belirtilen SQL ifadeleri yalnızca WHEN deyiminin true değerine sahip olduğu satırlar için çalıştırılır. WHEN deyimi girilmezse, SQL ifadeleri tüm satırlarda çalıştırılır.

Tetikleyicinin gövdesinde, (trigger-action deyimi) etkilenen tablonun değişiklik öncesi ve değişiklik sonrası değerleri, OLD ve NEW özel tablo adları kullanılarak kullanılabilir. OLD ve NEW tablolarının yapısı, tetikleyicinin oluşturulduğu tablonun yapısıyla eşleşir. OLD tablosu, tetikleyici ifadesi tarafından değiştirilen veya silinen satırları tetikleyici ifadesinin işlemlerinden önceki durumlarında olacak şekilde içerir. NEW tablosu, tetikleyici ifadesi tarafından değiştirilen veya oluşturulan satırları tetikleyici ifadesinin işlemlerinden sonraki durumlarında olacak şekilde içerir. Hem WHEN deyimi hem de trigger-step ifadeleri NEW.column-name ve OLD.column-name biçiminin başvuruları kullanılarak eklenen, silinen veya güncellenen satırdan değerlere erişebilir; burada, column-name, tetikleyicinin ilişkilendirildiği tablodaki sütunun adıdır. OLD ve NEW tablo başvurularının kullanılabilirliği, tetikleyicinin işlediği database-event türüne bağlıdır:

- INSERT - NEW başvuruları geçerlidir
- UPDATE - NEW ve OLD başvuruları geçerlidir
- DELETE - OLD başvuruları geçerlidir

Belirtilen zamanlama (BEFORE, AFTER veya INSTEAD OF), trigger-step ifadelerinin ilişkilendirilmiş ekleme, değiştirme veya kaldırmaya göre ne zaman çalıştırılacağını belirler. ON CONFLICT deyimi, trigger-step içindeki UPDATE veya INSERT ifadesinin bir parçası olarak belirlenebilir. Ancak, ON CONFLICT deyimi, tetikleyicinin çalışmasına neden olan ifadenin bir parçası olarak belirtilirse, bunun yerine bu çakışma işleme ilkesi kullanılır.

Tablo tetikleyicilerine ek olarak, görünümde bir INSTEAD OF tetikleyicisi oluşturulabilir. Bir görünümde bir veya birden fazla INSTEAD OF INSERT, INSTEAD OF DELETE veya INSTEAD OF UPDATE tetikleyicisi tanımlanırsa, ilişkilendirilmiş tipte ifadenin (INSERT, DELETE veya UPDATE) görünümde çalıştırılması bir hata olarak kabul edilmez. Bu durumda, görünümde bir INSERT, DELETE veya UPDATE ifadesinin çalıştırılması tetikleyicilerin çalışmasına yol açar. Tetikleyici bir INSTEAD OF tetikleyicisi olduğu için, görünümün temelini oluşturan tablolar, tetikleyicinin çalışmasına yol açan ifade tarafından değiştirilmez. Ancak, tetikleyiciler temel tablolarda değiştirme işlemleri yapmak için kullanılabilir.

INTEGER PRIMARY KEY sütunu bulunan bir tabloda bir tetikleyici oluştururken göz önünde bulundurulması gereken önemli bir nokta vardır. Bir BEFORE tetikleyicisi, tetikleyiciyi çalıştıran ifade tarafından güncellenecek satırın INTEGER PRIMARY KEY sütununu değiştirirse, güncelleme gerçekleştirilmez. Buna dolaylı bir çözüm olarak, INTEGER PRIMARY KEY sütunu yerine PRIMARY KEY sütunu olan bir tablo oluşturabilirsiniz.

Tetikleyici DROP TRIGGER ifadesi kullanılarak kaldırılabilir. Bir tablo veya görünüm kaldırıldığında, bu tablo veya görünümle ilişkilendirilmiş tüm tetikleyiciler de otomatik olarak kaldırılır.

RAISE () işlevi

Özel bir RAISE() SQL işlevi, bir tetikleyicinin trigger-step ifadesinde kullanılabilir. Bu işlev aşağıdaki sözdizimine sahiptir:

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

Tetikleyicinin çalıştırması sırasında ilk üç biçimden biri çağrıldığında, belirtilen ON CONFLICT işleme eylemi (ABORT, FAIL veya ROLLBACK) gerçekleştirilir ve geçerli ifadenin çalıştırılması sonlandırılır. ROLLBACK bir ifade çalıştırma hatası olarak göz önünde bulundurulur; dolayısıyla execute() yöntemi yürütülen SQLStatement örneği bir error (SQLErrorEvent.ERROR) olayı gönderir. Gönderilen olay nesnesinin error özelliğindeki SQLError nesnesinin details özelliği, RAISE() işlevinde belirtilen error-message değerine ayarlanır.

RAISE(IGNORE) çağrıldığında, geçerli tetikleyicinin geriye kalanı, tetikleyicinin çalıştırılmasına neden olan ifade ve çalıştırılmış olması gereken sonraki tetikleyiciler iptal edilir. Hiçbir veritabanı değişikliği geri alınmaz. Tetikleyicinin çalıştırılmasına neden olan ifadenin kendisi bir tetikleyicinin parçasıysa, bu tetikleyici programı sonraki adımın başından çalışmaya devam eder. Çakışma çözümleme algoritmaları hakkında daha fazla bilgi için ON CONFLICT (çakışma algoritmaları) bölümüne bakın.

DROP TRIGGER

DROP TRIGGER ifadesi, CREATE TRIGGER ifadesi tarafından oluşturulan bir tetikleyiciyi kaldırır.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

Tetikleyici, veritabanından silinir. Tetikleyicilerin, ilişkilendirilmiş tablosu kaldırıldığında otomatik olarak kaldırıldığını göz önünde bulundurun.

Özel ifadeler ve deyimler

Bu bölümde, çalışma zamanı tarafından sunulan SQL'in uzantısı olan çeşitli deyimlerin yanı sıra, birçok ifade ve yorumda kullanılabilecek iki dil öğesine ilişkin açıklamalar bulunur.

COLLATE

COLLATE deyimi, değerleri karşılaştırırken veya sıralarken kullanılan karşılaştırma algoritmasını belirtmek için SELECT, CREATE TABLE ve CREATE INDEX ifadelerinde kullanılır.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

Sütunlar için varsayılan harmanlama türü: BINARY. BINARY harmanlama TEXT saklama sınıfının değerlerinde kullanıldığında, ikili harmanlama, metin kodlamasına bakılmaksızın değeri gösteren bellekteki baytlar karşılaştırılarak gerçekleştirilir.

NOCASE harmanlama sırası yalnızca TEXT saklama sınıfının değerlerine uygulanır. Kullanıldığında, NOCASE harmanlama sırası, büyük/küçük harf duyarsız bir karşılaştırma yapar.

NULL, BLOB, INTEGER veya REAL türündeki saklama sınıfları için hiçbir harmanlama sırası kullanılmaz.

Bir sütunla BINARY dışında bir harmanlama türü kullanmak için, bir COLLATE deyiminin CREATE TABLE ifadesindeki sütun tanımının bir parçası olarak belirtilmesi gerekir. İki TEXT değeri karşılaştırıldığında, karşılaştırmanın sonuçlarını aşağıdaki kurallara göre belirlemek üzere bir harmanlama sırası kullanılır:

- İkili karşılaştırma operatörleri için, işlenenlerden biri bir sütunsa, sütunun varsayılan harmanlama türü karşılaştırma için kullanılan harmanlama sırasını belirler. Her iki işlenen de sütunsa, soldaki işlenenin harmanlama türü kullanılan harmanlama sırasını belirler. İşlenenlerden hiçbiri sütun değilse, BINARY harmanlama sırası kullanılır.
- BETWEEN...AND operatörü, >= ve <= operatörleriyle iki ifade kullanmaya eşdeğerdir. Örneğin, x BETWEEN y AND z ifadesi x >= y AND x <= z ifadesine eşdeğerdir. Sonuç olarak, BETWEEN...AND operatörü, harmanlama sırasını belirlemek için önceki kuralı izler.
- IN operatörü, kullanılacak harmanlama sırasını belirlemek için = operatörü gibi hareket eder. Örneğin, x IN (y, z) ifadesi için kullanılan harmanlama sırasını, x bir sütunsa, x için varsayılan harmanlama türüdür. Tersi durumda, BINARY harmanlama kullanılır.
- SELECT ifadesinin bir parçası olan bir ORDER BY deyimi, sıralama işlemi için kullanmak üzere bir harmanlama sırasına açık şekilde atanabilir. Bu durumda, açık harmanlama sırası her zaman kullanılır. Tersi durumda, ORDER BY tarafından sıralanan ifade bir sütunsa, sütunun varsayılan harmanlama türü sıralama düzenini belirlemek için kullanılır. İfade bir sütun değilse, BINARY harmanlama sırası kullanılır.

EXPLAIN

EXPLAIN komutu değiştiricisi, standart olmayan bir SQL uzantısıdır.

```
sql-statement  ::=  EXPLAIN sql-statement
```

EXPLAIN anahtar sözcüğü başka bir SQL ifadesinden önce gelirse, sonuç, komutu gerçekte çalıştırmak yerine EXPLAIN anahtar sözcüğü bulunmadığında komutu çalıştırmak için kullandığı sanal makine talimatlarının dizisini bildirir. EXPLAIN özelliği gelişmiş bir özelliktir ve geliştiricilerin performansı iyileştirmeyi denemek veya düzgün çalışmayan bir ifadede hata ayıklama işlemi gerçekleştirmek için SQL ifadesi metnini değiştirmelerini sağlar.

ON CONFLICT (çakışma algoritmaları)

ON CONFLICT deyimi ayrı bir SQL komutu değildir. Diğer birçok SQL komutunda görülebilen standart olmayan bir deyimdir.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

ON CONFLICT anahtar sözcüklerini kullanan ON CONFLICT deyiminin ilk biçimi bir CREATE TABLE ifadesine kullanılır. Bir INSERT veya UPDATE ifadesi için ikinci biçim kullanılır. Bu biçimde, söz diziminin daha doğal görünmesi için ON CONFLICT ifadesinin yerini OR ifadesi alır. Örneğin, INSERT ON CONFLICT IGNORE yerine, ifade INSERT OR IGNORE biçimine dönüşür. Anahtar sözcükler farklı olsa da, deyimin anlamı her iki biçimde de aynıdır.

ON CONFLICT deyimi, sınırlama çakışmalarını çözümlemek için kullanılan algoritmaları belirler. Beş algoritma şunlardır: ROLLBACK, ABORT, FAIL, IGNORE ve REPLACE. Varsayılan algoritma ABORT şeklindedir. Aşağıda, beş çakışma algoritmasının açıklaması mevcuttur:

ROLLBACK Bir sınırlama ihlali ortaya çıktığında, geçerli işlemi sona erdiren bir ROLLBACK hemen devreye girer. Komut iptal edilir ve SQLStatement örneği bir hata olayı gönderir. Hiçbir işlem etkin değilse (her komutta oluşturulan üstü kapalı işlemin dışında), bu algoritma ABORT ile aynı şekilde işler.

ABORT Bir sınırlama ihlali ortaya çıktığında, komut daha önce yapmış olabileceği değişiklikleri geri alır ve SQLStatement örneği bir hata olayı gönderir. Hiçbir ROLLBACK çalıştırılmaz; dolayısıyla bir işlem içerisindeki önceki komutlardan gelen değişiklikler korunur. ABORT varsayılan davranıştır.

FAIL Bir sınırlama ihlali ortaya çıktığında, komut iptal edilir ve SQLStatement bir hata olayı gönderir. Ancak, ifadenin sınırlama ihlaliyle karşılaşmadan önce veritabanında yaptığı değişiklikler korunur ve geri alınmaz. Örneğin, bir UPDATE ifadesi, güncellemeye çalıştığı 100. satırda bir sınırlama ihlaliyle karşılaşırsa, ilk 99 satırdaki değişiklikler korunur; ancak 100. satırda ve sonraki satırlarda yapılan değişiklikler uygulanmaz.

IGNORE Bir sınırlama ihlali ortaya çıktığında, sınırlama ihlalini içeren satır eklenmez veya değiştirilmez. Bu satırın yok sayılmasının yanı sıra, komut olağan şekilde çalışmaya devam eder. Sınırlama ihlalini içeren satırdan önce ve sonra gelen diğer satırlar olağan şekilde eklenmeye ve güncellenmeye devam eder. Hiçbir hata döndürülmez.

REPLACE UNIQUE sınırlama ihlali oluştuğunda, buna neden olan daha önceden varolan satırlar geçerli satır eklenmeden ve güncellenmeden önce kaldırılır. Sonuç olarak, ekleme ve güncelleme her zaman gerçekleşir ve komut olağan şekilde çalışmaya devam eder. Hiçbir hata döndürülmez. Bir NOT NULL sınırlama ihlali ortaya çıkarsa, NULL değeri bu sütun için varsayılan değerle değiştirilir. Sütun hiçbir varsayılan değere sahip değilse, ABORT algoritması kullanılır. CHECK sınırlama ihlali ortaya çıkarsa, IGNORE algoritması kullanılır. Bu çakışma çözümleme stratejisi, sınırlama işlemini yerine getirmek için satırlar silerse, bu satırlarda tetikleyicileri silme işlemini çağırmaz.

INSERT veya UPDATE ifadesinin OR deyiminde belirtilen algoritma, CREATE TABLE ifadesinde belirtilen bir algoritmayı geçersiz kılar. CREATE TABLE ifadesinde hiçbir algoritma belirtilmediğinde veya INSERT veya UPDATE ifadesi çalıştırılırken ABORT algoritması kullanılır.

REINDEX

REINDEX komutu, bir veya birden fazla dizini silmek veya yeniden oluşturmak için kullanılır. Bu komut, bir harmanlama sırasının tanımı değiştirildiğinde faydalıdır.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

İlk biçimde, bağlı tüm veritabanlarında belirtilen harmanlama sırasını kullanan tüm dizinler yeniden oluşturulur. İkinci biçimde, bir table-name belirtilirse, tablo ile ilişkilendirilmiş tüm dizinler yeniden oluşturulur. Bir index-name verilirse, yalnızca belirtilen dizin silinir ve yeniden oluşturulur.

YORUMLAR

Yorumlar, SQL komutu değildir, ancak SQL sorgularında görülebilir. Çalışma zamanı tarafından beyaz boşluk olarak işlenirler. Birden çok satıra yayılan ifadelerin içerisi de dahil olmak üzere, beyaz boşluğun bulunduğu her yerde başlayabilir.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

Tek satır yorumu iki tire işareti ile gösterilir. Tek satır yorumu yalnızca geçerli satırın sonuna kadar uzanabilir.

Blok yorumlar birden çok satıra yayılabilir veya tek bir satıra gömülebilir. Hiçbir sonlandırıcı ayırıcısı yoksa, blok yorum, girişin sonuna kadar yayılır. Bu durum bir hata olarak ele alınmaz. Yeni SQL ifadesi, blok yorum sona erdikten sonra bir satırda başlayabilir. Blok yorumlar, ifadelerin içerisi ve diğer SQL ifadelerinin ortası da dahil olmak üzere beyaz boşluğun bulunduğu her yere yerleştirilebilir. Blok yorumlar yuvalanmaz. Bir blok yorumun içindeki tek satırlı yorumlar yok sayılır.

İFADELER

İfadeler, diğer SQL bloklarının içindeki alt komutlardır. Aşağıda bir SQL ifadesinin içindeki bir ifade için geçerli sözdizimi açıklanmaktadır:

```
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
```

İfade, tek bir değere çözümlenebilen değerler ve operatörlerin bir birleşimidir. İfadeler, boolean değerine (true veya false) çözümlenmelerine veya boolean dışı bir değere çözümlenmelerine göre iki genel türe ayrılabilir.

WHERE deyimi, HAVING deyimi, JOIN deyimindeki ON ifadesi ve bir CHECK ifadesi de dahil olmak üzere çoğu yaygın durumda, ifadenin bir boolean değerine çözümlenmesi gerekir. Aşağıdaki ifade türleri bu koşulu yerine getirir:

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- Belirli işlevler
- Belirli operatörler (özellikle karşılaştırma operatörleri)

Değişmez değerler

Değişmez sayısal değer bir tam sayı veya kayar nokta sayısı olarak yazılır. Bilimsel gösterim desteklenir. . (nokta) karakteri her zaman ondalık ayırıcı olarak kullanılır.

Dize değişmezi, dizeyi tek tırnak işareti ' içine alarak gösterilir. Dizeye tek tırnak işareti eklemek için, satıra bu örnekteki gibi iki tek tırnak işareti ekleyin: ''.

Boolean değişmezi, true veya false değeriyle gösterilir. Değişmez boolean değerleri, Boolean sütunu veri türü ile kullanılır.

BLOB değişmezi, onaltılı veriler içeren ve başında tek bir x veya X karakteri bulunan bir dize değişmezidir; örneğin, X'53514697465'.

Değişmez değer NULL işareti de olabilir.

Sütun adı

Sütun adı, CREATE TABLE ifadesinde tanımlı adlardan biri veya aşağıdaki özel kimliklerden biri olabilir: ROWID, OID veya _ROWID_ Bu özel kimliklerin hepsi, her tablonun her satırıyla ilişkilendirilmiş benzersiz rastgele tam sayı anahtarını ("satır anahtarı") açıklar. Özel tanımlayıcılar yalnızca, CREATE TABLE ifadesi, aynı adı taşıyan gerçek bir sütunu tanımlamadığında satır anahtarına başvuruda bulunur. Satır anahtarları salt okunur sütunlar gibi hareket eder. Satır anahtarı, UPDATE veya INSERT ifadesindeki bir satır anahtarının değerini değiştirememeniz dışında, normal bir sütunun kullanılabileceği her yerde kullanılabilir. SELECT * FROM tablo ifadesi, sonuç kümesinde satır anahtarı içermez.

SELECT ifadesi

SELECT ifadesi, bir ifadede IN operatörünün sağ taraftaki işleneni, ölçekleme miktarı (tek sonuç değeri) veya EXISTS operatörünün işleneni olarak görülebilir. Ölçekleme miktarı veya bir IN operatörünün işleneni olarak kullanıldığında, SELECT ifadesi, sonucunda yalnızca tek bir sütuna sahip olabilir. Bileşik SELECT ifadesine (UNION veya EXCEPT gibi anahtar sözcüklerle bağlanan) izin verilir. EXISTS operatörüyle, SELECT ifadesinin sonuç kümesindeki sütunlar yok sayılır ve bir veya birden fazla sütun varsa ifade TRUE değerini ve sonuç kümesi boşsa FALSE değerini döndürür. SELECT ifadesindeki hiçbir terim, içeren sorgudaki değere başvurmazsa, ifade diğer işlemelerden önce değerlendirilir ve sonuç gerektiği şekilde yeniden kullanılır. SELECT ifadesi, bağıntılı alt sorgu olarak bilinen dış sorgudan değişkenler içermezse, SELECT gerek olan her defasında yeniden değerlendirilir.

SELECT ifadesi IN operatörünün sağ işleneni olduğunda, sol işlenenin sonucu SELECT ifadesinin sonuç kümesindeki değerlerden birine eşitse, IN operatörü TRUE değerini döndürür. Testin şeklini ters çevirmek için IN operatöründen önce NOT anahtar sözcüğü gelebilir.

SELECT ifadesi bir ifadede görülürse ancak, IN operatörünün sağ işleneni değilse, SELECT ifadesinin sonucunun ilk satırı ifadede kullanılan değer olur. SELECT birden fazla sonuç satırı döndürürse, ilk satırdan sonraki tüm satırlar yok sayılır. SELECT hiçbir satır döndürmezse, SELECT ifadesinin değeri NULL olur.

CAST ifadesi

CAST ifadesi, belirtilen değerin veri türünü belirtilen türle değiştirir. Belirtilen tür, bir CREATE TABLE ifadesinin sütun tanımındaki tür için geçerli herhangi bir boş olmayan tür olabilir. Ayrıntılar için Veri türü desteği bölümüne bakın.

Ek ifade öğeleri

Şu SQL öğeleri ifadelerde de kullanılabilir:

- Yerleşik işlevler (Toplama, ölçekleme ve tarih/saat biçimlendirme işlevleri)
- Operatörler
- Parametreler

Yerleşik işlevler

Yerleşik işlevler üç ana kategoriye ayrılır:

- Toplu işlevler
- Ölçekleme işlevleri
- Tarih ve saat işlevleri

Bu işlevlere ek olarak, bir tetikleyicinin çalıştırılması sırasına bir hatanın oluştuğunu bildirmek için kullanılan RAISE() özel işlevi de bulunur. Bu işlev, yalnızca bir CREATE TRIGGER ifadesinin gövdesinde kullanılabilir. RAISE() işlevine ilişkin bilgi için bkz. >CREATE TRIGGER > RAISE().

SQL'deki tüm anahtar sözcüklerde olduğu gibi, işlev adları da büyük/küçük harf duyarlı değildir.

Toplama işlevleri

Toplu işlevler, birden çok satırdan değerler üzerine işlemler gerçekleştirir. Bu işlevler çoğunlukla, SELECT ifadelerinde GROUP BY deyimiyle birlikte kullanılır.

AVG(X)	Bir grup içinde NULL olmayan tüm X'lerin ortalama değerini döndürür. Sayıya benzemeyen dize ve BLOB değerleri 0 olarak yorumlanır. Tüm girişler tam sayı olsa da, AVG() sonucu her zaman kayar nokta değeridir.
COUNT(X) COUNT(*)	İlk biçim, X'in bir grupta kaç kez NULL olmadığını döndürür. İkinci biçim (* argümanıyla), gruptaki toplam satır sayısını döndürür.
MAX(X)	Gruptaki tüm değerlerin maksimum değerini döndürür. Maksimum değeri belirlemek için her zamanki sıralama düzeni kullanılır.
MIN(X)	Gruptaki tüm değerlerin NULL olmayan minimum değerini döndürür. Minimum değeri belirlemek için her zamanki sıralama düzeni kullanılır. Gruptaki tüm değerler NULL ise, NULL döndürülür.
SUM(X) TOTAL(X)	Grupta NULL olmayan tüm değerlerin sayısal toplamını döndürür. Bütün değerler NULL ise; SUM(), NULL ve TOTAL(), 0 döndürür. TOTAL() sonucu her zaman kayan nokta değeridir. NULL olmayan tüm girişler tam sayı ise SUM() sonucu bir tam sayı değeri olur. SUM() girişlerinden biri tam sayı ve NULL değilse, SUM() bir kayan nokta değeri döndürür. Bu değer, gerçek toplamın yaklaşık bir sonucu olabilir.

Tek bir argüman alan önceki toplu işlevlerden herhangi birinde, bu argümandan önce DISTINCT anahtar sözcüğü gelebilir. Bu durumda, yinelenen öğeler toplu işleve iletilmeden önce filtrelenir. Örneğin, COUNT(DISTINCT x) işlev çağırısı, x sütununda NULL olmayan değerlerin toplam sayısı yerine X sütununun belirgin değerlerinin sayısını döndürür.

Ölçekleme işlevleri

Ölçekleme işlevleri değerler üzerinde bir defada bir satır olacak şekilde çalışır.

ABS(X)	X argümanının mutlak değerini döndürür.
COALESCE(X, Y, ...)	NULL olmayan ilk argümanın bir kopyasını döndürür. Tüm argümanlar NULL ise, NULL değeri döndürülür. En az iki argümanın olması gerekir.
GLOB(X, Y)	Bu işlev, X GLOB Y sözdizimini uygulamak için kullanılır.
IFNULL(X, Y)	NULL olmayan ilk argümanın bir kopyasını döndürür. Her iki argüman da NULL ise, NULL döndürülür. Bu işlev COALESCE() ile aynı şekilde işler.
HEX(X)	Argüman BLOB saklama türünün bir değeri olarak yorumlanır. Sonuç, bu değerin içeriğinin onaltılı oluşturmasıdır.
LAST_INSERT_ROWID( )	Geçerli SQLConnection üzerinden veritabanına son eklenen satırın satır kimliğini (oluşturulan ana anahtar) döndürür. Değer SQLConnection.lastInsertRowID özelliği tarafından döndürülen değerle aynıdır.
LENGTH(X)	X'in dize uzunluğunu karakter cinsinden döndürür.
LIKE(X, Y [, Z])	Bu işlev, SQL'in X LIKE Y [ESCAPE Z] sözdizimini uygulamak için kullanılır. İsteğe bağlı ESCAPE deyimi varsa, işlev üç argümanla çağrılır. Tersi durumda, yalnızca iki argümanla çağrılır.
LOWER(X)	X dizesinin bir kopyasını, tüm karakterler küçük harfe dönüştürülmüş şekilde döndürür.
LTRIM(X) LTRIM(X, Y)	X'in sol tarafındaki boşluklar kaldırılarak oluşturulan bir dize döndürür. Y argümanı belirtilirse işlev, Y altındaki herhangi bir karakteri X'in sol tarafından kaldırır.
MAX(X, Y, ...)	Maksimum değerli argümanı döndürür. Argümanlar, sayıların yanı sıra dizeler olabilir. Maksimum değer, sıralama düzeni tarafından belirlenir. MAX() işlevinin, 2 veya daha fazla argüman olduğunda basit bir işlev; ancak, tek bir argüman olduğunda toplu bir işlev olduğunu unutmayın.
MIN(X, Y, ...)	Minimum değerli argümanı döndürür. Argümanlar, sayıların yanı sıra dizeler olabilir. Minimum değer, sıralama düzeni tarafından belirlenir. MIN() işlevinin, 2 veya daha fazla argüman olduğunda basit bir işlev; ancak, tek bir argüman olduğunda toplu bir işlev olduğunu göz önünde bulundurun.
NULLIF(X, Y)	Argümanlar farklıysa ilk argümanı; tersi durumda, NULL değerini döndürür.
QUOTE(X)	Bu yordam, başka bir SQL ifadesine eklemek için uygun olan argümanının değeri olan bir dize döndürür. Dizeler, gerektiğinde iç tırnak imlerinde çıkışlarla birlikte tek tırnak işareti içine alınır. BLOB saklama sınıfları, onaltılı değişmezler olarak kodlanır. Bu işlev, geri al/yinele işlevini uygulamak için tetikleyiciler yazarken faydalıdır.
RANDOM(*)	-9223372036854775808 ile 9223372036854775807 arasında yapay olarak rastgele bir tam sayı döndürür. Bu rastgele değer güçlü bir şifrelemeye sahip değildir.
RANDOMBLOB(N)	Boş rastgele baytlar içeren bir N-bayt BLOB döndürür. N, pozitif bir tam sayı olmalıdır. Bu rastgele değer güçlü bir şifrelemeye sahip değildir. N değeri negatif ise, tek bayt döndürülür.
ROUND(X) ROUND(X, Y)	X sayısını, ondalık basamağın sağındaki Y basamağa yuvarlar. Y argümanı atılırsa, 0 kullanılır.
RTRIM(X) RTRIM(X, Y)	X'in sağ tarafındaki boşluklar kaldırılarak oluşturulan bir dize döndürür. Y argümanı belirtilirse, işlev Y içindeki karakterleri X'in sağ tarafından kaldırır.
SUBSTR(X, Y, Z)	Y. sıradaki karakterle başlayan ve Z uzunluğunda olan X giriş dizesinin alt dizesini döndürür. X'in en soldaki karakteri dizin konumu 1'dir. Y negatif ise, alt dizenin ilk karakteri soldan sağa yerine sağdan sola doğru sayılarak bulunur.
TRIM(X) TRIM(X, Y)	X'in sağ tarafındaki boşluklar kaldırılarak oluşturulan bir dize döndürür. Y argümanı belirtilirse, işlev Y içindeki karakterleri X'in sağ tarafından kaldırır.
TYPEOF(X)	X ifadesinin türünü döndürür. Döndürülen olası değerler, 'null', 'integer', 'real', 'text' ve 'blob'dur. Veri türleri hakkında daha fazla bilgi için bkz. Veri türü desteği.
UPPER(X)	X giriş dizesinin bir kopyasını tüm karakterler büyük harfe dönüştürülmüş şekilde döndürür.
ZEROBLOB(N)	0x00'ın N baytını içeren bir BLOB döndürür.

Tarih ve saat biçimlendirme işlevleri

Tarih ve saat biçimlendirme işlevleri, biçimlendirilmiş tarih ve saat verileri oluşturmak için kullanılan bir ölçekleme işlevleri grubudur. Bu işlevlerin çalıştığını ve dize ve sayı değerleri döndürdüğünü göz önünde bulundurun. Bu işlevler, DATE veri türüyle kullanılmak üzere tasarlanmamıştır. Bu işlevleri, belirtilmiş veri türü DATE olan bir sütunda kullanırsanız, beklenildiği gibi davranış sergilemezler.

DATE(T, ...)	DATE() işlevi, şu biçimdeki tarihi içeren bir dize döndürür: YYYY-AA-GG. İlk parametre (T), Saat biçimleri altında bulunan biçimin saat dizesini belirler. Saat dizesinden sonra istenilen sayıda değiştirici belirtilebilir. Değiştiriciler, Değiştiriciler'in altında bulunabilir.
TIME(T, ...)	TIME() işlevi, saati SS:DD:SS olarak içeren bir dize döndürür. İlk parametre (T), Saat biçimleri altında bulunan biçimin saat dizesini belirler. Saat dizesinden sonra istenilen sayıda değiştirici belirtilebilir. Değiştiriciler, Değiştiriciler'in altında bulunabilir.
DATETIME(T, ...)	DATETIME() işlevi, YYYY-AA-GG SS:DD:SS biçiminde tarih ve saat içeren bir dize döndürür. İlk parametre (T), Saat biçimleri altında bulunan biçimin saat dizesini belirler. Saat dizesinden sonra istenilen sayıda değiştirici belirtilebilir. Değiştiriciler, Değiştiriciler'in altında bulunabilir.
JULIANDAY(T, ...)	JULIANDAY() işlevi, M.Ö 24 Kasım 4714 öğlen tarihi ile belirtilen tarih arasında Greenwich saatine göre geçen gün sayısını gösteren bir sayı döndürür. İlk parametre (T), Saat biçimleri altında bulunan biçimin saat dizesini belirler. Saat dizesinden sonra istenilen sayıda değiştirici belirtilebilir. Değiştiriciler, Değiştiriciler'in altında bulunabilir.
STRFTIME(F, T, ...)	STRFTIME() yordamı, ilk F argümanı olarak belirtilen biçim dizesine göre biçimlendirilen tarihi döndürür. Biçim dizesi yerine şunların kullanılması desteklenir: %d - ayın günü %f - kesirli saniye SS.SSS %H - saat 00-24 %j - yılın günü 001-366 %J - Julyen gün sayısı %m -ay 01-12 %M - dakika 00-59 %s - 01.01.1970'ten bu yana geçen saniye %S - saniye 00-59 %w - haftanın günü 0-6 (pazar = 0) %W - yılın haftası 00-53 %Y - yıl 0000-9999 %% - % İkinci parametre (T), Saat biçimleri altında bulunan biçimin saat dizesini belirler. Saat dizesinden sonra istenilen sayıda değiştirici belirtilebilir. Değiştiriciler, Değiştiriciler'in altında bulunabilir.

Saat biçimleri

Saat dizesi aşağıdaki biçimlerden birinde olabilir:

YYYY-AA-GG	2007-06-15
YYYY-AA-GG SS DD	2007-06-15 07:30
YYYY-AA-GG SS DD:SS	2007-06-15 07:30:59
YYYY-AA-GG SS DD:SS.SSS	2007-06-15 07:30:59.152
YYYY-AA-GGTSS:DD	2007-06-15T07:30
YYYY-AA-GGTSS:DD:SS	2007-06-15T07:30:59
YYYY-AA-GGTSS:DD:SS.SSS	2007-06-15T07:30:59.152
SS:DD	07:30 (tarih 2000-01-01)
SS DD:SS	07:30:59 (tarih 2000-01-01)
SS DD:SS.SSS	07:30:59:152 (tarih 2000-01-01)

şimdi	Eşgüdümlü Evrensel Saat cinsinden o sıradaki tarih ve saat.
GGGG.GGGG	Kayar nokta sayısı olarak Julyen gün sayısı

Bu biçimlerdeki T karakteri, tarih ve saati birbirinden ayıran değişmez "T" karakteridir. Yalnızca saatin olduğu biçimler tarih olarak 2001-01-01 tarihini kabul eder.

Değiştiriciler

Saat dizesinden sonra, tarihi veya tarihin yorumlanmasını değiştiren sıfır veya daha fazla değiştirici gelebilir. Kullanılabilen değiştiriciler şunlardır:

NNN gün	Saate eklenecek gün sayısı.
NNN saat	Saate eklenecek saat sayısı.
NNN dakika	Saate eklenecek dakika sayısı.
NNN.NNNN saniye	Saate eklenecek saniye ve milisaniye sayısı.
NNN ay	Saate eklenecek ay sayısı.
NNN yıl	Saate eklenecek yıl sayısı.
ay başı	Saati ayın başına gelecek şekilde geriye doğru kaydırır.
yıl başı	Saati yılın başına gelecek şekilde geriye doğru kaydırır.
gün başı	Saati günün başına gelecek şekilde geriye doğru kaydırır.
haftasonu N	Saati belirtilen haftanın gününe gelecek şekilde ileriye doğru kaydırır. (0 = Pazar, 1 = Pazartesi, vb.)
yerelsaat	Tarihi yerel saate dönüştürür.
utc	Tarihi Eşgüdümlü Evrensel Saat'e dönüştürür.

Operatörler

SQL, SQL'e özgü birçok operatörün yanı sıra birçok programlama dilinde olan yaygın operatörler de dahil olmak üzere, geniş çapta operatör seçimini destekler.

Yaygın operatörler

Aşağıdaki ikili operatörlere SQL bloğunda izin verilir ve en yüksek öncelikli olandan en düşük öncelikli olana doğru listelenir:

```
*     /     %
+     -
<<  >>  &  |
<  >=    >  >=
=     ==    !=    <>  IN
AND
OR
```

Desteklenen tekli önek operatörleri şunlardır:

```
 !     ~     NOT
```

COLLATE operatörü, tekli sonek operatörü olarak düşününebilir. COLLATE operatörü en yüksek önceliğe sahiptir. Bir önek tekli operatöründen veya ikili bir operatörden her zaman daha sıkı bağlar.

Eşittir veya eşit değildir operatörlerinin iki çeşitlemesinin olduğunu göz önünde bulundurun. Eşittir = veya == olabilir. Eşit değildir operatörü != veya <> olabilir.

|| operatörü dize bitiştirme operatörüdür - iki dizeyi işlenenlerinden birbiriyle birleştirir.

% operatörü, sol işlenen modülünün kalan kısmını sağ işlenenine çıkarır.

İkili bir operatörün sonucu, dize sonucu veren || birleştirme operatörü dışında sayısal bir değerdir.

SQL operatörleri

LIKE

LIKE operatörü bir model eşleştirme karşılaştırması gerçekleştirir.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

LIKE operatörünün sağındaki işlenen, modeli içerir ve soldaki işlenen modelle eşleştirilecek dizeyi içerir. Modeldeki yüzde sembolü (%) bir joker karakterdir; dizedeki sıfır dizisi veya daha fazla karakterle eşleşir. Modeldeki alt çizgi (_) dizedeki herhangi bir tek karakterle eşleşir. Diğer karakter kendiyle veya küçük/büyük harf eşdeğeriyle eşleşir; diğer bir deyişle, eşleştirmeler büyük/küçük harf duyarsız şekilde gerçekleştirilir. Not: veritabanı motoru yalnızca 7 bitlik Latin karakterler için büyük/küçük harfi anlar. Sonuç olarak, LIKE operatörü, 8 bitlik iso8859 karakterleri veya UTF-8 karakterleri için büyük/küçük harf duyarlıdır. Örneğin, 'a' LIKE 'A' ifadesi TRUE; ancak 'æ' LIKE 'Æ' ifadesi FALSE) değerine sahiptir. Latin karakterleri için büyük/küçük harf duyarlılığı SQLConnection.caseSensitiveLike özelliği kullanılarak değiştirilebilir.

İsteğe bağlı ESCAPE deyimi varsa, ESCAPE anahtar sözcüğünü izleyen ifade, tek bir karakterden oluşan bir dizede değerlendirilmelidir. Bu karakter, değişmez yüzde veya alt çizgi karakterlerini eşleştirmek için LIKE modelinde kullanılabilir. Ardından yüzde sembolü, alt çizgi veya kendisi gelen bir çıkış karakteri, sırasıyla dizedeki değişmez yüzde sembolü, alt çizgi veya çıkış karakteriyle eşleşir.

GLOB

GLOB operatörü LIKE operatörüne benzer; ancak, joker karakterleri için Unix dosyası genelleştirme sözdizimini kullanır. LIKE operatöründen farklı olarak, GLOB operatörü büyük/küçük harf duyarlıdır.

IN

IN operatörü, sol işleneninin sağ işlenenindeki (parantez içindeki değerler kümesi) değerlerden birine eşit olup olmadığını hesaplar.

```
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]*
```

Sağ işlenen, virgülle ayrılmış değişmez değerler kümesi veya bir SELECT ifadesinin sonucu olabilir. IN operatörünün sağ işleneni olarak SELECT ifadesinin kullanılmasına yönelik açıklama ve sınırlamalar için ifadelerdeki SELECT ifadelerine bakın.

BETWEEN...AND

BETWEEN...AND operatörü, >= ve <= operatörleriyle iki ifade kullanmaya eşdeğerdir. Örneğin, x BETWEEN y AND z ifadesi x >= y AND x <= z ifadesine eşdeğerdir.

NOT

NOT operatörü bir negatiflik operatörüdür. GLOB, LIKE ve IN operatörlerinin başına NOT anahtar sözcüğü gelerek testin şekli tersine çevrilir (diğer bir deyişle, bir değerin belirtilen modelle eşleşmediğini göstermek için).

Parametreler

Bir parametre, çalışma zamanı sırasında SQLStatement.parameters ilişkilendirici dizisine bir değer atayarak doldurulan bir değişmez değer için ifadede bir yer tutucu belirler. Parametreler üç biçimde olabilir:

?	Soru işareti dizin oluşturulmuş bir parametreyi gösterir. Parametrelere ifadedeki sıralarına göre sayısal (sıfır tabanlı) dizin değerleri atanır.

:AAAA	Ardından bir kimlik gelen bir iki nokta üst üste imi, AAAA adındaki adlandırılmış parametre için bir yer tutar. Adlandırılmış parametreler, SQL ifadesindeki sıralarına göre de numaralandırılır. Karışıklığı önlemek için, adlandırılmış ve numaralandırılmış parametreleri karışık kullanmaktan kaçının.
@AAAA	Bir "at işareti" iki nokta üst üsteye eşdeğerdir.

Desteklenmeyen SQL özellikleri

Aşağıda, Adobe AIR'de desteklenmeyen standart SQL öğelerinin bir listesi bulunmaktadır:

FOREIGN KEY sınırlamaları FOREIGN KEY sınırlamaları ayrıştırılır; ancak zorlanmaz.

Tetikleyiciler FOR EACH STATEMENT tetikleyicileri desteklenmez (her tetikleyicinin FOR EACH ROW tetikleyicisi olması gerekir). INSTEAD OF tetikleyicileri tablolarda desteklenmez (INSTEAD OF tetikleyicilerine yalnızca görünümlerde izin verilir). Yinelenen tetikleyiciler - kendi kendini tetikleyen tetikleyiciler- desteklenmez.

ALTER TABLE ALTER TABLE komutunun yalnızca RENAME TABLE ve ADD COLUMN varyasyonları desteklenir. DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT ve benzeri diğer ALTER TABLE işlemleri yok sayılır.

Yuvalanmış işlemler Yalnızca tek bir etkin işleme izin verilir.

RIGHT ve FULL OUTER JOIN RIGHT OUTER JOIN ve FULL OUTER JOIN desteklenmez.

Güncellenebilir VIEW Görünüm salt okunurdur. Bir görünümde DELETE, INSERT veya UPDATE ifadesi çalıştıramazsınız. Bir görünümde DELETE, INSERT veya UPDATE ifadesi çalıştırma girişimini tetikleyen bir INSTEAD OF tetikleyicisi desteklenir ve tetikleyicinin gövdesinde desteklenen tabloları güncellemek için kullanılabilir.

GRANT ve REVOKE Veritabanı normal bir disk dosyasıdır; uygulanabilecek erişim izinleri temeldeki işletim sisteminin dosya erişim izinleridir. Genelde istemci/sunucu RDBMS'lerinde bulunan GRANT ve REVOKE komutları uygulanmaz.

Aşağıdaki SQL öğeleri ve SQLite özellikleri bazı SQLite uygulamalarında desteklenir; ancak, Adobe AIR'de desteklenmez. Bu işlevlerin çoğu SQLConnection sınıfı yöntemleri üzerinden kullanılabilir:

İşlemle ilgili SQL öğeleri (BEGIN, END, COMMIT, ROLLBACK) Bu işlev SQLConnection sınıfının işlemle ilgili yöntemleri üzerinden kullanılabilir: SQLConnection.begin()SQLConnection.commit()SQLConnection.rollback()

ANALYZE Bu işlev, SQLConnection.analyze() yöntemi üzerinden kullanılabilir.

ATTACH Bu işlev, SQLConnection.attach() yöntemi üzerinden kullanılabilir.

COPY Bu ifade desteklenmez.

CREATE VIRTUAL TABLE Bu ifade desteklenmez.

DETACH Bu işlev SQLConnection.detach() yöntemi üzerinden kullanılabilir.

PRAGMA Bu ifade desteklenmez.

VACUUM Bu işlev SQLConnection.compact() yöntemi üzerinden kullanılabilir.

Sistem tablo erişimi kullanılamaz sqlite_master ve "sqlite_" önekinin olduğu diğer tabloların da içinde olduğu sistem tabloları SQL ifadelerinde kullanılamaz. Çalışma zamanı, şema verilerine nesne odaklı erişim sunan bir şema API'si içerir. Daha fazla bilgi için SQLConnection.loadSchema() yöntemine bakın.

Normal ifade işlevleri (MATCH() ve REGEX() Bu işlevler SQL ifadelerinde kullanılmaz.

Aşağıdaki işlev, birçok SQLite uygulamasında ve Adobe AIR'de farklılık gösterir:

Dizinlenmiş ifade parametreleri Birçok uygulamada, dizinlenmiş ifade parametreleri bir tabanlıdır. Ancak, Adobe AIR dizinlenmiş ifade parametreleri sıfır tabanlıdır (başka bir deyişle, ilk parametreye 0 dizini, ikinci parametreye 1 dizini verilir ve bu şekilde devam eder).

INTEGER PRIMARY KEY sütun tanımları Bir çok uygulamada, tablo için esas birincil anahtar sütun olarak yalnızca INTEGER PRIMARY KEY olarak tanımlanmış sütunlar kullanılır. Bu uygulamalarda çoğu zaman INTEGER ile eşanlamlı olan bir veri türünün (örn. int) kullanılması, sütunun dahili birincil anahtar olarak kullanılmasına neden olmaz. Ancak Adobe AIR'de, int veri türü (ve diğer INTEGER ile eşanlamlı olan öğeler), INTEGER öğesine tam olarak eşdeğer sayılır. Sonuç olarak, PRIMARY KEY olarak tanımlanmış bir sütun tablo için dahili birincil anahtar olarak kullanılır. Daha fazla bilgi için CREATE TABLE ve Sütun yakınlığı bölümlerine bakın.

Ek SQL özellikleri

Aşağıdaki sütun yakınlık türleri SQLite'ta varsayılan olarak desteklenmez; ancak, Adobe AIR'de desteklenir (SQL'deki bütün anahtar sözcükler gibi veri türü adlarının da büyük/küçük harf duyarlı olmadığına dikkat edin):

Boolean Boolean sınıfına karşılık gelir.

Date Date sınıfına karşılık gelir.

int int sınıfına karşılık gelir (INTEGER sütun yakınlığına eşdeğerdir).

Number Number sınıfına karşılık gelir (REAL sütun yakınlığına eşdeğerdir).

Nesne Object sınıfına veya AMF3 kullanılarak serileştirilebilen ve serileştirmesi kaldırılabilen alt sınıfa karşılık gelir. (Bu, özel sınıflar da dahil olmak üzere birçok sınıfı içerir; ancak, ekran nesneleri ve ekran nesnelerini özellik olarak içeren nesnelerin de aralarında bulunduğu bazı nesneleri hariç bırakır.)

Dize String sınıfına karşılık gelir (TEXT sütun yakınlığına eşdeğerdir).

XML ActionScript (E4X) XML sınıfına karşılık gelir.

XMLList ActionScript (E4X) XMLList sınıfına karşılık gelir.

Aşağıdaki değişmez değerler SQLite'ta varsayılan olarak desteklenmez; ancak, Adobe AIR'de desteklenir:

true BOOLEAN sütunlarıyla çalışmak üzere true değişmez boolean değerini göstermek için kullanılır.

false BOOLEAN sütunlarıyla çalışmak üzere false değişmez boolean değerini göstermek için kullanılır.

Veri türü desteği

Çoğu SQL veritabanından farklı olarak, Adobe AIR SQL veritabanı motoru, bu tablo sütunlarının belirli türde değerler içermesini gerektirmez veya bunu zorlamaz. Çalışma zamanı bunun yerine, veri türlerini denetlemek için saklama sınıfları ve sütun yakınlığı olmak üzere iki kavram kullanır. Bu bölümde, saklama sınıfları ve sütun yakınlığının yanı sıra, çeşitli koşullar altında veri türü farklılıklarının nasıl çözümlendiği açıklanır:

- "Saklama sınıfları" sayfa 1138
- "Sütun yakınlığı" sayfa 1138
- "Veri türleri ve karşılaştırma operatörleri" sayfa 1141
- "Veri türleri ve matematik operatörler" sayfa 1141
- "Veri türleri ve sıralama" sayfa 1142

- "Veri türleri ve gruplama" sayfa 1142
- "Veri türleri ve bileşik SELECT ifadeleri" sayfa 1142

Saklama sınıfları

Saklama sınıfları, veritabanında değerleri saklamak için kullanılan gerçek veri türlerini gösterir. Şu saklama sınıfları veritabanı tarafından kullanılır:

NULL Değer NULL değeridir.

INTEGER Değer, imzalı bir tam sayıdır.

REAL Değer, bir kayan nokta sayısı değeridir.

TEXT Değer, bir metin dizesidir (256 MB ile sınırlıdır).

BLOB Değer, bir BLOB'dur (İkili Büyük Nesne); başka bir deyişle, işlenmemiş ikili verilerdir (256 MB ile sınırlıdır).

Veritabanına değişmez değerler olarak gönderilen tüm değerler SQL ifadesine gömülür veya hazırlanmış bir SQL ifadesine parametreler kullanılarak bağlanan değerler SQL ifadesi çalıştırılmadan önce bir saklama sınıfına atanır.

SQL ifadesinin bir parçası olan değişmez değerler, tek veya çift tırnak işareti içine alınırsa, alınan TEXT saklama sınıfına; değişmez değer tırnak içinde olmayan ondalık basamağı veya üssü olmayan bir sayı olarak belirtilirse INTEGER saklama sınıfına; değişmez değer tırnak içinde olmayan ondalık basamağı veya üssü olan bir sayı olarak belirtilirse REAL saklama sınıfına ve değer NULL ise NULL saklama sınıfına atanır. BLOB saklama sınıfındaki değişmez değerler X'ABCD' gösterimi kullanılarak belirtilir. Daha fazla bilgi için İfadelerdeki değişmez değerler konusuna bakın.

SQLStatement.parameters ilişkilendirici dizisi kullanılarak parametre olarak girilen değerlere yerel veri türü bağıyla en yakından eşleşen saklama sınıfı atanır. Örneğin, int değerleri INTEGER saklama sınıfı olarak bağlanır; Number değerleri REAL saklama sınıfı olarak girilir; String değerlerine TEXT saklama sınıfı verilir ve ByteArray nesnelerine BLOB saklama sınıfı verilir.

Sütun yakınlığı

Bir sütunun *sütun yakınlığı*, o sütunda saklanan veri için önerilen türdür. Bir değer sütunda saklandığında (INSERT veya UPDATE ifadesiyle), çalışma zamanı bu değeri kendi veri türünden belirtilen yakınlığa dönüştürmeyi dener. Örneğin, bir Date değeri (ActionScript veya JavaScript Date örneği) yakınlığı TEXT olan bir sütuna eklenir, Date değeri String gösterimine dönüştürülür (veritabanında saklanmadan önce nesnenin toString() yöntemini çağırmaya eşdeğerdir). Değer belirtilen yakınlığa dönüştürülemezse, bir hata oluşur ve işlem gerçekleştirilmez. Bir değer SELECT ifadesi kullanılarak veritabanından alındığında, saklandığında farklı bir veri türüne dönüştürülüp dönüştürülmediğine bakılmaksızın yakınlığa karşılık gelen sınıfın bir örneği olarak döndürülür.

Bir sütun NULL değerleri kabul ederse, ActionScript veya JavaScript null değeri, NULL değerini sütunda saklamak için bir parametre değeri olarak kullanılabilir. NULL saklama sınıfı değeri bir SELECT ifadesinde alındığında, sütunun yakınlığına bakılmaksızın her zaman ActionScript veya JavaScript null değeri olarak döndürülür. Bir sütun NULL değerlerini kabul ederse, değerleri boş olamayan bir değere (Number veya Boolean gibi) dağıtmaya çalışmadan önce null olup olmadıklarını belirlemek üzere bu sütundan alınan değerleri her zaman denetleyin.

Veritabanındaki her sütuna aşağıdaki tür yakınlıklarından biri atanır:

- TEXT (veya String)
- NUMERIC
- INTEGER (veya int)

- REAL (veya Number)
- Boolean
- Date
- XML
- XMLLIST
- Nesne
- NONE

TEXT (veya String)

TEXT veya String yakınlığına sahip bir sütun, NULL, TEXT veya BLOB saklama sınıflarını kullanarak tüm verileri saklar. TEXT yakınlığına sahip bir sütuna sayısal veriler eklenirse, saklanmadan önce metin biçimine dönüştürülür.

NUMERIC

NUMERIC yakınlığına sahip bir sütun, NULL, REAL veya INTEGER saklama sınıflarını kullanarak değerleri içerir. NUMERIC bir sütuna metin verileri girildiğinde, saklanmadan önce bu değeri bir tam sayı veya gerçek sayıya dönüştürmek için girişimde bulunulur. Dönüştürme başarılı olursa, değer INTEGER veya REAL saklama sınıfı kullanılarak saklanır (örneğin, '10.05' değeri saklanmadan önce REAL saklama sınıfına dönüştürülür). Dönüştürme gerçekleştirilemezse, bir hata oluşur. NULL değerini dönüştürmek için hiçbir girişimde bulunulmaz. NUMERIC sütunundan alınan bir değer, değerin uyacağı en belirli sayısal türün bir örneği olarak döndürülür. Diğer bir deyişle, değer bir pozitif tam sayı veya 0 ise, bir uint örneği olarak döndürülür. Negatif bir tam sayı ise, int örneği olarak döndürülür. Son olarak, kayan nokta bileşenine (tam sayı değildir) sahipse, Number örneği olarak döndürülür.

INTEGER (veya int)

INTEGER yakınlığını kullanan bir sütun, bir istisna dışında NUMERIC yakınlığına sahip bir sütunla aynı şekilde hareket eder. Saklanacak değer, hiçbir kayan nokta bileşeni olmayan gerçek bir değerse (Number örneği gibi) veya değer, kayan nokta bileşeni olmayan gerçek bir değere dönüştürülebilecek bir metin değeriyse, bir tam sayıya dönüştürülür ve INTEGER saklama sınıfı kullanılarak saklanır. Kayar nokta bileşeni olan bir gerçek değeri saklamak için bir girişimde bulunulursa, bir hata oluşur.

REAL (veya Number)

REAL veya NUMBER yakınlığına sahip bir sütun, tam sayı değerlerini kayan nokta gösterimine zorlamasının dışında NUMERIC yakınlığa sahip bir sütun gibi hareket eder. REAL sütunundaki bir değer veritabanından her zaman Number örneği olarak döndürülür.

Boolean

BOOLEAN yakınlığına sahip bir sütun true veya false değerlerini saklar. BOOLEAN sütunu, bir ActionScript veya JavaScript Boolean örneği olan bir değeri kabul eder. Kod bir String değerini saklamaya çalışırsa, sıfırdan daha büyük uzunluğa sahip bir String true olarak kabul edilir ve boş bir String ise false olarak kabul edilir. Kod, sayısal verileri saklamaya çalışırsa, sıfır olmayan bir değer true olarak ve 0 değeri false olarak saklanır. Boolean değeri SELECT ifadesi kullanılarak alındığında, Boolean örneği olarak döndürülür. NULL olmayan değerler INTEGER saklama sınıfı (false için 0 ve true için 1) kullanılarak saklanır ve veriler alındığında Boolean nesnelerine dönüştürülür.

Date

DATE yakınlığına sahip bir sütun tarih ve saat değerlerini saklar. DATE sütunu, ActionScript veya JavaScript Date örnekleri olan değerleri kabul edecek şekilde tasarlanmıştır. Bir String değerini DATE sütununa saklamak için girişimde bulunulursa, çalışma zamanı bunu Julyen tarihine dönüştürmeye çalışır. Dönüştürme başarısız olursa bir hata oluşur. Kod, bir Number, int veya uint değerini saklamaya çalışırsa, verileri doğrulamak için hiçbir girişimde bulunulmaz ve geçerli bir Julyen tarih değeri olduğu varsayılır. SELECT ifadesi kullanılarak alınan bir DATE değeri otomatik olarak bir Date örneğine dönüştürülür. DATE değerleri REAL saklama sınıfı kullanılarak Julyen tarih değerleri olarak saklanır; dolayısıyla, sıralama ve karşılaştırma işlemleri beklediğiniz şekilde çalışır.

XML veya XMLList

XML veya XMLLIST yakınlığını kullanan bir sütun XML yapılarını saklar. Kod, SQLStatement parametresi kullanarak verileri bir XML sütununda saklamaya çalıştığında, çalışma zamanı, değeri ActionScript XML() veya XMLList() işlevini kullanarak dönüştürmeye ve doğrulamaya çalışır. Değer geçerli bir XML'e dönüştürülemezse, bir hata oluşur. Verileri saklama girişimi değişmez bir SQL metin değeri (örneğin, INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)') kullanırsa, değer ayrıştırılmaz veya doğrulanmaz - düzgün biçimli olduğu varsayılır. Geçersiz bir değer saklanırsa, alındığında boş bir XML nesnesi olarak döndürülür. XML ve XMLLIST verileri TEXT saklama sınıfı veya NULL saklama sınıfı kullanılarak saklanır.

Nesne

Nesne yakınlığına sahip bir sütun, Object sınıfı örneklerinin yanı sıra Array örnekleri ve hatta özel sınıf örnekleri gibi Object alt sınıfları da dahil olmak üzere ActionScript veya JavaScript karmaşık nesnelerini saklar. OBJECT sütunu verileri AMF3 biçiminde serileştirilir ve BLOB saklama sınıfı kullanılarak saklanır. Bir değer alındığında, AMF3'ten serileştirilmesi kaldırılır ve saklandığı sınıf gibi bir örnek olarak döndürülür. Bazı ActionScript sınıflarının, özellikle de ekran nesnelerinin serileştirilmesinin, özgün veri türlerinin örnekleri olarak kaldırılamayacağını göz önünde bulundurun. Özel sınıf örneğini saklamadan önce, flash.net.registerClassAlias() yöntemini (veya Flex'te sınıf bildirimine [RemoteObject] meta verileri ekleyerek) kullanarak sınıf için bir takma ad kaydetmeniz gerekir. Ayrıca, bu verileri almadan önce, sınıf için aynı takma adı kaydetmeniz gerekir. Yapısı gereği veya eksik veya hatalı eşleşen sınıf takma adı nedeniyle serileştirilmesi kaldırılamayan bir sınıf gibi, serileştirilmesi düzgün şekilde kaldırılamayacak veriler, orijinal örneğe karşılık gelen özellikler ve değerlerle birlikte adsız bir nesne (Object sınıfı örneği) olarak döndürülür.

NONE

NONE yakınlığına sahip bir sütun, bir saklama sınıfını bir diğerine tercih etmez. Eklemeden önce verileri dönüştürmek için hiçbir girişimde bulunmaz.

Yakınlığı belirleme

Bir sütunun tür yakınlığı, sütunun CREATE TABLE ifadesinde belirtilen türüne göre belirlenir. Türü belirlerken aşağıdaki kurallar (büyük/küçük harf duyarlı değil) geçerlidir:

- Sütunun veri türü "CHAR", "CLOB", "STRI" veya "TEXT" dizelerinden birini içerirse, bu sütun TEXT/String yakınlığına sahiptir. VARCHAR türünde "CHAR" dizesinin olduğuna ve dolayısıyla TEXT yakınlığının atandığına dikkat edin.
- Sütunun veri türü "BLOB" dizesini içerirse veya hiçbir veri türü belirtilmezse, sütun NONE yakınlığına sahiptir.
- Sütun için veri türü "XMLL" dizesini içerirse, sütun XMLLIST yakınlığına sahiptir.
- Veri türü "XML" dizesini içerirse, sütun XML yakınlığına sahiptir.
- Veri türü "OBJE" dizesini içerirse, sütun Object yakınlığına sahiptir.
- Veri türü "BOOL" dizesini içerirse, sütun Boolean yakınlığına sahiptir.
- Veri türü "DATE" dizesini içerirse, sütun Date yakınlığına sahiptir.
- Veri türü "INT" dizesini içerirse ("UINT" dahil), INTEGER/int yakınlığına atanır.

- Sütunun veri türü "REAL", "NUMB", "FLOA" veya "DOUB" dizelerinden birini içerirse, sütun REAL/Number yakınlığına sahiptir.
- Tersi durumda, yakınlık NUMERIC olur.
- Bir tablo CREATE TABLE t AS SELECT... ifadesi kullanılarak oluşturulursa, tüm sütunlarda hiçbir veri türü belirtilmez ve NONE yakınlığı atanır.

Veri türleri ve karşılaştırma operatörleri

Aşağıdaki ikili karşılaştırma operatörleri =, <, <=, >= ve !=, belirtilen üyeliği test etmek için IN işlemi ve üç bölümden oluşan BETWEEN karşılaştırma operatörüyle birlikte desteklenir. Bu operatörler hakkında bilgi için bkz. Operatörler.

Karşılaştırmanın sonucu, karşılaştırılan iki değerin saklama sınıflarına bağlıdır. İki değeri karşılaştırırken aşağıdaki kurallar geçerlidir:

- NULL saklama sınıfına sahip bir değer, başka bir değerden küçük kabul edilir (NULL saklama sınıfındaki başka bir değer de dahil olmak üzere).
- INTEGER veya REAL değeri TEXT veya BLOB değerinden küçüktür. INTEGER veya REAL değeri başka bir INTEGER veya REAL değeriyle karşılaştırıldığında, sayısal bir karşılaştırma gerçekleştirilir.
- TEXT değeri bir BLOB değerinden küçüktür. İki TEXT değeri karşılaştırıldığında, ikili bir karşılaştırma gerçekleştirilir.
- İki BLOB değeri karşılaştırıldığında, sonuç her zaman ikili karşılaştırma kullanılarak belirlenir.

Üç bölümden oluşan BETWEEN operatörü her zaman, eşdeğer ikili ifade olarak yeniden dağıtılır. Örneğin, a BETWEEN b AND c ifadesi, ifadeyi değerlendirmek için gerekli her karşılaştırmadaki a için farklı yakınlıkların uygulanması anlamına gelse de, a >= b AND a <= c olarak yeniden dağıtılır.

a IN (SELECT b ....) türündeki ifadeler, ikili karşılaştırmalar için daha önce belirtilen üç kural tarafından işlenir; diğer bir deyişle, a = b ifadesiyle aynı şekilde. Örneğin, b bir sütun değeriyse ve a bir ifadeyse, b yakınlığı, bir karşılaştırma yapılmadan önce a öğesine uygulanır. a IN (x, y, z) ifadesi, a = +x OR a = +y OR a = +z olarak yeniden dağıtılır. IN operatörünün sağındaki değerler (bu örnekte, x, y ve z değerleri), sütun değerleri olsalar bile ifade olarak kabul edilir. IN operatörünün solundaki değer bir sütunsa, bu sütunun yakınlığı kullanılır. Değer bir ifadeyse, hiçbir dönüştürme gerçekleştirilmez.

Karşılaştırmaların nasıl gerçekleştirildiği bir COLLATE deyiminin kullanılmasına da bağlıdır. Daha fazla bilgi için COLLATE ifadesine bakın.

Veri türleri ve matematik operatörler

Desteklenen her matematik operatörü; *, /, %, + ve -; için ifade değerlendirilmeden önce her işlenene sayısal yakınlık uygulanır. Bir işlenen NUMERIC saklama sınıfına başarıyla dönüştürülemezse, ifade NULL olarak değerlendirilir.

|| bitiştirme operatörü kullanıldığında, her işlenen, ifade değerlendirilmeden önce TEXT saklama sınıfına dönüştürülür. Bir işlenen TEXT saklama sınıfına dönüştürülemezse, ifadenin değeri NULL olur. Değerin dönüştürülememesi durumu iki koşulda görülür; işlenenin değeri NULL ise veya TEXT dışı bir saklama sınıfı içeren bir BLOB ise.

Veri türleri ve sıralama

Değerler ORDER BY deyimi ile sıralandığında, NULL saklama sınıfındaki değerler ilk sırada yer alır. Bunları, sayısal düzende araya dağıtılan INTEGER ve REAL değerleri izler; bunları da ikili düzendeki veya belirtilen harmanlamaya dayalı (BINARY veya NOCASE) TEXT değerleri izler. Son olarak ikili düzende BLOB değerleri gelir. Sıralamadan önce hiçbir saklama sınıfı dönüştürmesi gerçekleştirilmez.

Veri türleri ve gruplama

Değerleri GROUP BY deyimiyle grupladığınızda, farklı saklama sınıflarına sahip değerler belirgin olarak düşünülür. Bir istisna, sayısal olarak eşdeğerse eşit olarak kabul edilen INTEGER ve REAL değerleridir. GROUP BY deyimi sonucu değerlere hiçbir yakınlık uygulanmaz.

Veri türleri ve bileşik SELECT ifadeleri

Bileşik SELECT operatörleri olan UNION, INTERSECT ve EXCEPT, değerler arasında üstü kapalı karşılaştırmalar yapar. Bu karşılaştırmalar yapılmadan önce, her değere bir yakınlık uygulanabilir. Varsa aynı yakınlık, bileşik SELECT sonuç kümesinin tek bir sütununda döndürülebilecek tüm değerlere uygulanır. Uygulanan yakınlık, bu konumda sütun değerine sahip (başka bir ifade türünde olmayan) ilk bileşen SELECT ifadesi tarafından döndürülen sütunun yakınlığıdır. Belirtilen bir bileşik SELECT sütunu için bileşen SELECT ifadelerinin hiçbiri bir sütun değeri döndürmezse, karşılaştırma gerçekleştirmeden önce bu sütundan değerlere hiçbir yakınlık uygulanmaz.

Bölüm 67: SQL hata ayrıntı mesajları, kimlikler ve argümanlar

SQLError sınıfı, bir Adobe AIR yerel SQL veritabanıyla çalışırken ortaya çıkabilecek çeşitli hataları gösterir. Belirli bir istisna için, SQLError örneği, İngilizce hata mesajını içeren `details` özelliğine sahiptir. Ayrıca, her hata mesajı SQLError nesnesinin `detailID` özelliğinde mevcut olan ilişkili benzersiz bir tanımlayıcıya sahiptir. `detailID` özelliğini kullanarak bir uygulama, hata mesajının belirli ayrıntılar hata mesajını tanımlayabilir. Uygulama son kullanıcı için kendi yerel ayarının dilinde alternatif bir metin sunabilir. `detailArguments` dizisindeki argüman değerleri, hata mesajı dizesinde uygun konumda değiştirilebilir. Bu, belirli bir yerel ayardaki son kullanıcılara bu hata için doğrudan `details` özelliği hata mesajını görüntüleyen uygulamalar için yararlıdır.

Aşağıdaki tablo, `detailID` değerlerinin ve ilişkilendirilmiş İngilizce hata mesajı metninin bir listesini içerir. Mesajlardaki yer tutucu metin, `detailArguments` değerlerinin çalışma zamanı tarafından nerede değiştirileceğini gösterir. Bu liste, SQL veritabanı işlemlerinde ortaya çıkan hata mesajlarının yerelleştirilmesinde kaynak olarak kullanılabilir.

SQLError detailID	İngilizce hata ayrıntısı mesajı ve parametreler
1001	Bağlantı kapalı.
1102	Bu işlemi gerçekleştirmek için veritabanının açılması gerekir.
1003	%s [,|ve %s] parametre adları parameters özelliğinde bulundu; ancak belirtilen SQL'de bulunamadı.
1004	Parametre sayısında hatalı eşleşme. Belirtilen SQL'de %d bulundu ve %d değerleri parameters özelliğinde ayarlandı.%s [,|ve %s] için değerler bekleniyor.
1005	Otomatik sıkıştırma açılamadı.
1006	pageSize değeri ayarlanamadı.
1007	%s' türündeki '%s' adlı şema nesnesi '%s' veritabanında bulunamadı.
1008	'%s' adlı şema nesnesi '%s' veritabanında bulunamadı.
1009	'%s' veritabanında '%s' türünde hiçbir şema nesnesi bulunamadı.
1010	'%s' veritabanında hiçbir şema nesnesi bulunamadı.
2001	Parser stack overflow. (Ayrıştırıcı yığınında taşma meydana geldi.)
2002	Too many arguments on function '%s' ('%s' işlevinde çok fazla argüman var)
2003	'%s' öğesinin yanında: sözdizimi hatası
2004	bu ada sahip başka bir tablo veya dizin zaten var: '%s'
2005	SQL'de PRAGMA öğesine izin verilmiyor.
2006	Bu yazılabilir bir dizin değil.
2007	Bilinmeyen veya desteklenmeyen birleştirme türü: '%s %s %s'
2008	RIGHT ve FULL OUTER JOIN öğeleri şu anda desteklenmiyor.
2009	NATURAL olan birleştirme ON veya USING deyimine sahip olamaz.
2010	Aynı birleştirmede hem ON hem de USING deyimleri olamaz.
2011	'%s' sütunu kullanılarak birleştirilemiyor - sütun her iki tabloda da yok.
2012	İfadenin parçası olan SELECT öğesi için yalnızca tek bir sonuca izin verilir.
2013	Böyle bir tablo yok: '[%s.]%s'
2014	Belirtilmiş tablo yok.
2015	Sonuç kümesinde çok fazla sütun var|'%s' öğesinde çok fazla sütun var.
2016	%s ORDER|GROUP BY terimi aralığın dışında - 1 ile %d arasında olmalıdır
2017	ORDER BY deyiminde çok fazla terim var.
2018	%s ORDER BY terimi aralığın dışında - 1 ile %d arasında olmalıdır.
2019	%r ORDER BY terimi, sonuç kümesindeki hiçbir sütunla eşleşmiyor.
2020	ORDER BY deyimi '%s' öğesinden sonra gelmeli, önce gelmemelidir.
2021	LIMIT deyimi '%s' öğesinden sonra gelmeli, önce gelmemelidir.
2022	'%s' öğesinin sol ve sağ tarafındaki SELECT öğeleri sonuç sütunlarıyla aynı sayıya sahip değildir.
2023	GROUP BY deyimi HAVING deyiminden önce gerekir.

2024	GROUP BY deyiminde toplama işlevlerine izin verilmez.
2025	Toplama işlevindeki DISTINCT deyiminden sonra bir ifade gelmelidir.
2026	Bileşik SELECT öğesinde çok fazla terim var.
2027	ORDER|GROUP BY deyiminde çok fazla terim var.
2028	Geçici tetiğin nitelikli bir adı olmayabilir
2030	'%s' tetiği zaten var
2032	Şu görünümde BEFORE|AFTER tetiği oluşturulamaz: '%s'.
2033	Şu tabloda INSTEAD OF tetiği oluşturulamıyor: '%s'.
2034	Böyle bir tetik yok: '%s'
2035	Yinelenen tetikleyiciler desteklenmez ('%s').
2036	Böyle bir sütun yok: %s[.%s[.%s]]
2037	SQL'de VACUUM öğesine izin verilmiyor.
2043	'%s' tablosu: dizin oluşturma işlevi geçersiz bir plan döndürdü.
2044	Bir birleştirmede en fazla %d tablo.
2046	PRIMARY KEY sütunu eklenemiyor.
2047	UNIQUE sütunu eklenemiyor.
2048	Varsayılan NULL değeriyle NOT NULL sütunu eklenemiyor.
2049	Sabit olmayan varsayılan değerle bir sütun eklenemiyor.
2050	Görünüme bir sütun eklenemiyor.
2051	SQL'de ANALYZE öğesine izin verilmiyor.
2052	Geçersiz ad: '%s'
2053	SQL'de ATTACH öğesine izin verilmiyor.
2054	%s '%s', '%s' veritabanındaki nesnelere başvuru yapamaz
2055	'[%s.]%s.%s' öğesine erişim engelleniyor.
2056	Yetkili değil.
2058	Böyle bir görünüm yok: '[%s.]%s'
2060	Geçici tablo adı niteliksiz olmalıdır.
2061	'%s' tablosu zaten var.
2062	Şu ada sahip bir dizin zaten var: '%s'
2064	Çift sütun adı: '%s'
2065	'%s' tablosu birden fazla ana anahtara sahip.
2066	AUTOINCREMENT öğesine yalnızca INTEGER PRIMARY KEY üzerinde izin verilir
2067	Böyle bir harmanlama sırası yok: '%s'
2068	Parametrelere görünümlerde izin verilmez.
2069	'%s' görünümü döngüsel olarak tanımlanır.
2070	'%s' tablosu atılamaz.
2071	'%s' görünümünü silmek için DROP VIEW öğesini kullanın
2072	'%s' tablosunu silmek için DROP TABLE öğesini kullanın
2073	'%s' öğesindeki yabancı anahtarın '%s' tablosunun yalnızca bir sütununa başvurması gerekir
2074	Yabancı anahtardaki sütunların sayısı başvurulan tablodaki sütunların sayısıyla eşleşmiyor.
2075	Yabancı anahtar tanımında '%s' bilinmeyen sütun bulunuyor.
2076	'%s' tablosu dizine eklenemez.
2077	Görünümler dizine eklenemez.
2080	Çakışan ON CONFLICT deyimleri belirtildi.
2081	Böyle bir dizin yok: '%s'
2082	UNIQUE veya PRIMARY KEY sınırlamasıyla ilişkili dizin atılamaz.
2083	SQL'de BEGIN öğesine izin verilmiyor.
2084	SQL'de COMMIT öğesine izin verilmiyor.
2085	SQL'de ROLLBACK öğesine izin verilmiyor.
2086	Geçici tabloları depolamak için geçici bir veritabanı dosyası açılamıyor.
2087	Yeniden dizine alınacak nesne tanımlamıyor.
2088	'%s' tablosu değiştirilemez.
2089	Bir görünüm olduğundan '%s' öğesi değiştirilemiyor.
2090	Değişken numarası ?0 ve ?%d< arasında olmalıdır
2092	Takma ada sahip toplamın yanlış kullanımı: '%s'
2093	Belirsiz sütun adı: '[%s.[%s.]]%s'
2094	Böyle bir işlev yok: '%s'

2095	'%s' işlevinde yanlış sayıda argüman
2096	CHECK sınırlamalarında alt sorgular kullanılamaz.
2097	CHECK sınırlamalarında parametreler kullanılamaz.
2098	İfade ağacı çok büyük (maksimum derinlik: %d)
2099	RAISE(), yalnızca bir tetikleyici programla kullanılabilir
2100	'%s' tablosu %d sütunlarına sahip ancak %d değerleri sağlandı
2101	Veritabanı şeması kilitli: '%s'
2102	İfade çok uzun.
2103	Etkin ifadeler nedeniyle karşılaştırma dizisi silinemiyor/değiştirilemiyor
2104	Çok fazla ekli veritabanı var - maks. %d
2105	İşlem içinde veritabanı EKLENEMİYOR.
2106	'%s' veritabanı zaten kullanılıyor.
2108	Eklenen veritabanları ana veritabanı olarak aynı metin kodlamasını kullanmalıdır.
2200	Bellek yetersiz.
2201	Veritabanı açılamıyor.
2202	İşlem içinde veritabanı AYRILAMIYOR.
2203	Veritabanı ayrılamıyor: '%s'
2204	'%s' veritabanı kilitli.
2205	Veritabanında bir okuma kilidi alınamıyor.
2206	[column|columns] '%s'['%s'], [unique|is] benzersiz değil.
2207	Kötü biçimlendirilmiş veritabanı şeması.
2208	Desteklenmeyen dosya biçimi.
2209	Tanınmayan simge: '%s'
2300	Metin değeri sayısal bir değere dönüştürülemiyor.
2301	Dize değeri tarihe dönüştürülemiyor.
2302	Akan nokta değeri veri kaybı olmadan tam sayıya dönüştürülemedi.
2303	İşlem geri alınamıyor - SQL ifadeleri işleniyor
2304	İşlem gerçekleştirilemiyor - SQL ifadeleri işleniyor
2305	Veritabanı tablosu kilitli: '%s'
2306	Salt okunur tablo.
2307	Dize veya blob çok büyük.
2309	Yazma için dizine alınmış sütun açılamıyor.
2400	%s türü değeri açılamıyor.
2401	Böyle bir satır kimliği yok: %s<
2402	Nesne adı dahili kullanım için saklanıyor: '%s'
2403	'%s' görünümü değiştirilemez.
2404	'%s' sütununun varsayılan değeri sabit değil.
2405	'%s' işlevini kullanma yetkisi yok
2406	'%s' toplama işlevi yanlış bir şekilde kullanıldı
2407	Şu toplu işlev yanlış kullanıldı: '%s'
2408	Böyle bir veritabanı yok: '%s'
2409	'%s' tablosunun '%s' adında bir sütunu yok
2501	Böyle bir modül yok: '%s'
2508	Böyle bir kayıt noktası yok: '%s'
2510	Geri çağırılamıyor - etkin işlem yok.
2511	Gerçekleştirilemedi - etkin işlem yok.

Bölüm 68: Adobe Grafikleri Çevirici Dili (AGAL)

Adobe Grafikleri Çevirici Dili (AGAL), köşe ve parça oluşturucu programları tanımlayan bir gölgelendirici dilidir. AGAL programları, oluşturma bağlamına bu belgede açıklanan ikili bayt kodu biçiminde yüklenmelidir.

AGAL bayt kodu biçimi

AGAL bayt kodu, Endian.LITTLE_ENDIAN biçimini kullanmalıdır.

Bayt Kodu Üstbilgisi

AGAL bayt kodu 7 baytlık bir üstbilgi ile başlamalıdır.

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

Uzaklık (bayt)	Boyut (bayt)	Ad	Açıklama
0	1	magic	0xa0 olmalıdır
1	4	sürüm	1 olmalıdır
5	1	gölgelendirici türü kimliği	0xa1 olmalıdır
6	1	gölgelendirici türü	köşe programı için 0; parça programı için 1

Belirteçler

Üstbilginin hemen arkasından herhangi bir sayıda belirteç gelir. Her belirteç 192 bit (24 bayt) boyutundadır ve her zaman şu biçime sahiptir:

```
[opcode][destination][source1][source2 or sampler]
```

Bu alanların tümünü her işlem kodu kullanmaz. Kullanılmayan alanlar 0 olarak ayarlanmalıdır.

İşlem kodları

[işlem kodu] alanı 32 bit boyutundadır ve şu değerlerden birini alabilir:

Ad	İşlem kodu	İşlem	Açıklama
mov	0x00	taşı	verileri kaynak1 öğesinden hedef öğesine bileşen yönünde taşıyın
add	0x01	add	hedef = kaynak1 + kaynak2, bileşen yönünde
sub	0x02	subtract	hedef = kaynak1 - kaynak2, bileşen yönünde
mul	0x03	çarp	hedef = kaynak1 * kaynak2, bileşen yönünde
div	0x04	böl	hedef = kaynak1 / kaynak2, bileşen yönünde

Ad	İşlem kodu	İşlem	Açıklama
rcp	0x05	ters	hedef = 1/kaynak1, bileşen yönünde
min	0x06	minimum	hedef = minimum(kaynak1,kaynak2), bileşen yönünde
max	0x07	maximum	hedef = maksimum(kaynak1,kaynak2), bileşen yönünde
frc	0x08	kesirsel	hedef = kaynak1 - (kaydır)taban(kaynak1), bileşen yönünde
sqt	0x09	kare kök	hedef = sqrt(kaynak1), bileşen yönünde
rsq	0x0a	ters kök	hedef = 1/sqrt(kaynak1), bileşen yönünde
pow	0x0b	üssü	hedef = pow(kaynak1,kaynak2), bileşen yönünde
log	0x0c	logaritma	hedef = log_2(kaynak1), bileşen yönünde
exp	0x0d	üssel	hedef = 2^kaynak1, bileşen yönünde
nrm	0x0e	normalleştir	hedef = normalleştir(kaynak1), bileşen yönünde (yalnızca 3 bileşenli bir sonuç verir; hedef, .xyz veya daha azı olarak maskelenmelidir)
sin	0x0f	sinüs	hedef = sin(kaynak1), bileşen yönünde
cos	0x10	kosinüs	hedef = cos(kaynak1), bileşen yönünde
crs	0x11	vektörel çarpım	hedef.x = kaynak1.y * kaynak2.z - kaynak1.z * kaynak2.y hedef.y = kaynak1.z * kaynak2.x - kaynak1.x * kaynak2.z hedef.z = kaynak1.x * kaynak2.y - kaynak1.y * kaynak2.x (yalnızca 3 bileşenli bir sonuç verir, hedef .xyz veya daha azı olarak maskelenmelidir)
dp3	0x12	skaler çarpım	hedef = kaynak1.x*kaynak2.x + kaynak1.y*kaynak2.y + kaynak1.z*kaynak2.z
dp4	0x13	skaler çarpım	hedef = kaynak1.x*kaynak2.x + kaynak1.y*kaynak2.y + kaynak1.z*kaynak2.z + kaynak1.w*kaynak2.w
abs	0x14	mutlak	hedef = abs(kaynak1), bileşen yönünde
neg	0x15	negatif yapma	hedef = -kaynak1, bileşen yönünde
sat	0x16	doygun hale getirme	hedef = maksimum(minimum(kaynak1,1),0), bileşen yönünde
m33	0x17	çarpma matrisi 3x3	hedef.x = (kaynak1.x * kaynak2[0].x) + (kaynak1.y * kaynak2[0].y) + (kaynak1.z *kaynak2[0].z) hedef.y = (kaynak1.x * kaynak2[1].x) + (kaynak1.y * kaynak2[1].y) + (kaynak1.z * kaynak2[1].z) hedef.z = (kaynak1.x * kaynak2[2].x) + (kaynak1.y * kaynak2[2].y) + (kaynak1.z * kaynak2[2].z) (yalnızca 3 bileşenli bir sonuç verir, hedef .xyz veya daha azı olarak maskelenmelidir)

Ad	İşlem kodu	İşlem	Açıklama
m44	0x18	çarpma matrisi 4x4	hedef.x = (kaynak1.x * kaynak2[0].x) + (kaynak1.y * kaynak2[0].y) + (kaynak1.z * kaynak2[0].z) + (kaynak1.w * kaynak2[0].w) hedef.y = (kaynak1.x * kaynak2[1].x) + (kaynak1.y * kaynak2[1].y) + (kaynak1.z * kaynak2[1].z) + (kaynak1.w * kaynak2[1].w) hedef.z = (kaynak1.x * kaynak2[2].x) + (kaynak1.y * kaynak2[2].y) + (kaynak1.z * kaynak2[2].z) + (kaynak1.w * kaynak2[2].w) hedef.w = (kaynak1.x * kaynak2[3].x) + (kaynak1.y * kaynak2[3].y) + (kaynak1.z * kaynak2[3].z) + (kaynak1.w * kaynak2[3].w)
m34	0x19	çarpma matrisi 3x4	hedef.x = (kaynak1.x * kaynak2[0].x) + (kaynak1.y * kaynak2[0].y) + (kaynak1.z * kaynak2[0].z) + (kaynak1.w * kaynak2[0].w) hedef.y = (kaynak1.x * kaynak2[1].x) + (kaynak1.y * kaynak2[1].y) + (kaynak1.z * kaynak2[1].z) + (kaynak1.w * kaynak2[1].w) hedef.z = (kaynak1.x * kaynak2[2].x) + (kaynak1.y * kaynak2[2].y) + (kaynak1.z * kaynak2[2].z) + (kaynak1.w * kaynak2[2].w) (yalnızca 3 bileşenli bir sonuç verir, hedef .xyz veya daha azı olarak maskelenmelidir)
kil	0x27	sonlandır/at (yalnızca parça gölgelendiricisi)	Tek skaler kaynak bileşeni sıfırdan daha düşükse, parça atılır ve çerçeve arabelleğine çizilmez. (Hedef kaydının tümü 0 olarak ayarlanmalıdır)
tex	0x28	doku örneği (yalnızca parça gölgelendiricisi)	hedef öğesi, kaynak2 öğesinden gelen yüke kaynak1 koordinatlarında eşittir. Bu durumda kaynak2 öğesi, örnekleyici biçiminde olmalıdır.
sge	0x29	büyük-eşitse-ayarla	hedef = kaynak1 >= kaynak2 ? 1 : 0, bileşen yönünde
slt	0x2a	daha-küçükse-ayarla	hedef = kaynak1 < kaynak2 ? 1 : 0, bileşen yönünde
seq	0x2c	eşitse-ayarla	hedef = kaynak1 == kaynak2 ? 1 : 0, bileşen yönünde
sne	0x2d	eşit-değilse-ayarla	hedef = kaynak1 != kaynak2 ? 1 : 0, bileşen yönünde

AGAL2'de aşağıdaki işlem kodları kullanıma sunulmuştur:

Ad	İşlem kodu	İşlem	Açıklama
ddx	0x1a	X içinde kısmi türev	Kaynak1'in X içindeki kısmi türevini hedefe yükleyin.
ddy	0x1b	Y içinde kısmi türev	Kaynak1'in Y içindeki kısmi türevini hedefe yükleyin.
ife	0x1c	eşitse	Kaynak1 ile kaynak2 eşitse atlayın.
ine	0x1d	eşit değilse	Kaynak1 ile kaynak2 eşit değilse atlayın.
ifg	0x1e	büyükse	Kaynak1 kaynak2 ile eşit veya daha büyükse atlayın.
ifl	0x1f	küçükse	Kaynak1 kaynak2'den küçükse atlayın.
els	0x20	else	Else bloğu
eif	0x21	Endif	If veya else bloğu ise kapatın.

Hedef alanı biçimi

[hedef] alanı 32 bit boyutundadır:

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = Kayıt türü (4 bit)

M = Yazma maskesi (4 bit)

N = Kayıt numarası (16 bit)

- = tanımsız, 0 olmalı

Kaynak alanı biçimi

[kaynak] alanı 64 bit boyutundadır:

```
63.............................................................0
D------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

Doğrudan Q ve I için D = Doğrudan=0/Dolaylı=1 yoksayılır, 1 bit

Q = Dizin kaydı bileşeninin seçimi (2 bit)

I = Dizin kaydı türü (4 bit)

T = Kayıt türü (4 bit)

S = Yeniden düzenle (8 bit, bileşen başına 2 bit)

O = Dolaylı uzaklık (8 bit)

N = Kayıt numarası (16 bit)

- = tanımsız, 0 olmalı

Örnekleyici alanı biçimi

Metin işlem kodunun ikinci kaynak alanı 64 bit boyutundaki [örnekleyici] formatında olmalıdır:

```
63.............................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = Örnekleyici kayıt numarası (16 bit)

B = Doku ayrıntı seviyesinin (LOD) sapması, işaretli tamsayı, 8'e kadar ölçek. Kullanılan kayan nokta değeri b/8.0'dır (8 bit).

T = Kayıt türü, 5 olmalı, Örnekleyici (4 bit)

F = Filtre (0=en yakın,1=doğrusal) (4 bit)

M = Mipmap (0=devre dışı bırak,1=en yakın, 2=doğrusal)

W = Kaydırma (0=kıskaç,1=tekrarla)

S = Özel bayrak bit'i (0 olmalı)

D = Boyut (0=2B, 1=Küp)

Program Kayıtları

Kullanılan kayıt sayısı, kullanılan Context3D profiline bağlıdır. Kullanımlarıyla birlikte kayıt sayısı aşağıdaki tabloda tanımlanmıştır:

Ad **AGAL2** **AGAL3** **Kullanım**	**Değer**	**AGAL**	
Parça programı başına sayı Köşe programı başına sayı Parça programı başına sayı Köşe programı başına sayı		Parça programı başına sayı	Köşe programı başına sayı
Context 3D Profilleri Desteği Standart Genişletilmiş Standart		Standardın Altında	
SWF sürümü 25 28 ve üzeri		25'in altında	
Nitelik Yok 8 Yok 16 Köşe gölgelendirici girdisi; Context3D.setVertexBufferAt() öğesini kullanarak belirtilen köşe arabelleğinden okuma.	0	Yok	8

Ad **AGAL2** **AGAL3** **Kullanım**	**Değer**	**AGAL**	
Sabit 64 250 200 250 Gölgelendirici girdisi; işlevlerin Context3D.setProgramConstants() ailesini kullanarak ayarlama.	1	28	128
Geçici 26 26 26 26 Programın dışından erişilmeyen geçici hesaplama kaydı.	2	8	8
Çıktı 1 1 1 1 Gölgelendirici çıktısı: çıktı, köşe programında klip alanı konumudur; parça programında ise renktir.	3	1	1

Ad **AGAL2** **AGAL3** **Kullanım**	**Değer**	**AGAL**	
Çeşitli 10 10 10 10 Köşe ve parça gölgelendiriciler arasındaki ara enterpolasyonlu verileri aktarma. Köşe programında değişkenlik gösteren kayıtlar, parça programına girdi olarak uygulanır. Değerler, üçgen köşe noktalarından olan uzaklığa göre enterpolasyonlanır.	4	8	8

Ad **AGAL2** **AGAL3** **Kullanım**	**Değer**	**AGAL**	
Örnekleyici 16 Yok 16 Yok Parça gölgelendirici girdisi; Context3D.setTextureAt() öğesini kullanarak belirtilen dokudan okuma.	5	8	Yok
Parça kaydı 1 Yok 1 Yok Salt yazılır niteliktedir ve köşe gölgelendiriciye yazılmış z değerini (veya derinlik değerini) yeniden yazmak için kullanılır.	6	Yok	Yok
Belirteçler 1024 2048		200	

En son AGAL Mini Assembler burada bulunabilir.