

Flex 2.0.1: Arquitecturas modulares

Xavi Beumala



USER GROUP

m · a · d · e i · n



Outline

- Escenarios de aplicación existentes
- Distintas problemáticas
- Estructura e interpretación del formato swf
- ApplicationDomains
- RSL's
- Carga dinámica de estilos
- Carga dinámica de módulos
- Técnicas de modularización

Escenarios de aplicación existentes

- La nueva AVM2 introduce grandes mejoras en el rendimiento.
- Al tener mayor rendimiento las expectativas funcionales crecen
- Las aplicaciones resultantes son más grandes

Problemáticas a afrontar

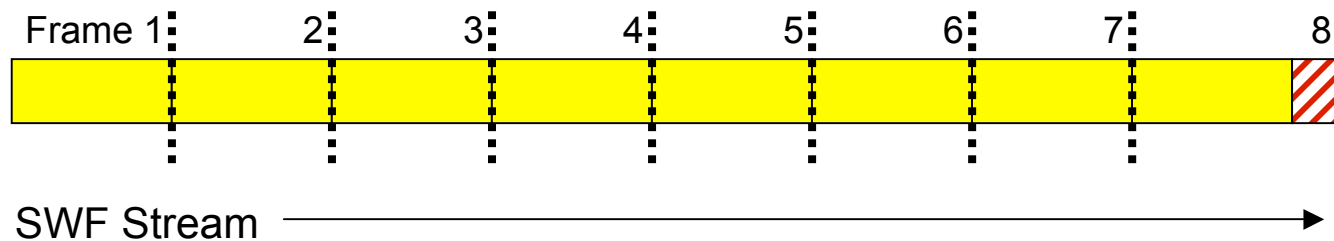
- Penalización en el tiempo de descarga y de carga de las aplicaciones
- Abuso de ancho de bando usado
- Tiempo de compilación in-crecendo
- Extensibilidad de las aplicaciones. Trabajo concurrente entre distintos equipos. Distribución de la carga de trabajo
- Versionado de porciones de código
- Uso de frameworks





Estructura e interpretación del formato SWF I

- El formato SWF se concibió desde sus orígenes para permitir cargas progresivas interactivas.
- Cada frame contiene definiciones de assets y de acciones
- Un frame es reproducible sí y sólo si se ha completado la carga íntegra de sus assets y acciones asociadas

Estructura e interpretación del formato SWF II

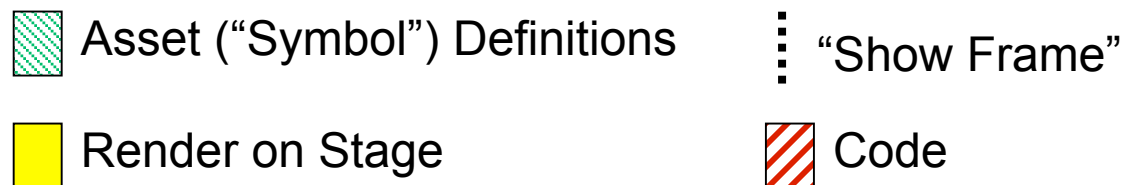
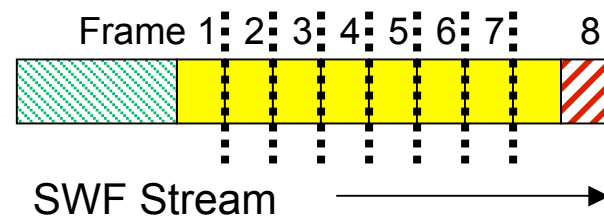
- Aplicación en línea de tiempo sin utilizar la librería



-  Asset ("Symbol") Definitions
-  Render on Stage
-  "Show Frame"
-  Code

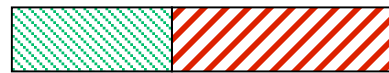
Estructura e interpretación del formato SWF III

- Aplicación en línea de tiempo utilizando la librería.



Estructura e interpretación del formato SWF IV


- Aplicación por código utilizando la librería.



SWF Stream →

 Asset (“Symbol”) Definitions

 “Show Frame”

 Render on Stage

 Code

Estructura e interpretación del formato SWF V

- Conclusiones y aplicación a Flex
 - Flex trabaja tradicionalmente con 2 frames
 - Uno lightWeight donde se ejecuta el preload
 - Otro heavyWeight donde se carga todo el código de las aplicaciones (>100 Kb)
 - Un punto de optimización sería eliminar el código no necesario en start-up (disminuiríamos el tamaño del frame 2)

ApplicationDomains

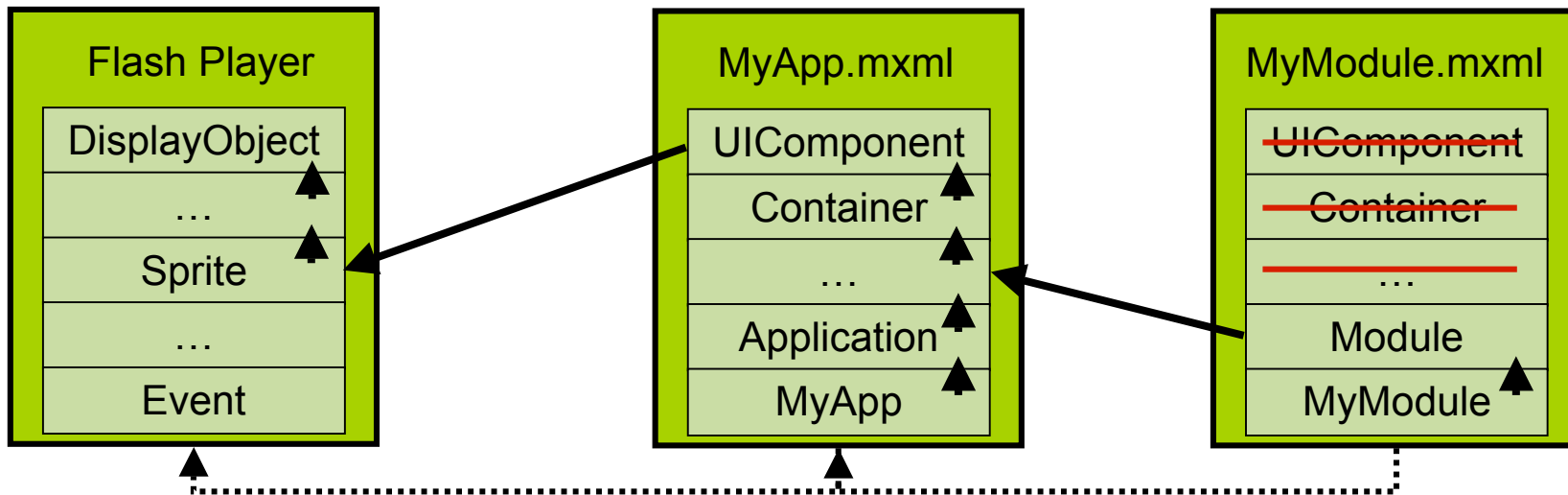
- Un ApplicationDomain es un contexto en el que se cargan elementos y definiciones (funciones, clases, etc).
- Es el equivalente de los classLoaders en Java
- Se pueden utilizar para añadir seguridad a nuestro código. Impedir que módulos cargados externamente accedan a las definiciones del cargador o impidiendo la colaboración entre distintas partes

ApplicationDomains II

- Un mx:Application tiene un applicationDomain
- El operador 'new' busca en el applicationDomain de su contexto la clase que toque. Si no la encuentra busca en el applicationDomain padre y así sucesivamente: chain of responsibility
- Si dos partes de la aplicación intentan censar la misma clase en un mismo ApplicationDomain se aplica la regla: first in win!

Application Domains III

- Una misma definición puede estar presente en dos ApplicationDomains hermanos. En este caso las clases no se reconocerán mutuamente y sus instancias no serán polimórficas.



RSL's

- Resource Shared Libraries
- Son el equivalente de las librerías/bibliotecas compartidas de flash.
- Característica que permite la paquetización de código común entre distintas aplicaciones que se cargarán en runtime (tiempo de ejecución)
- Suelen ser clases de soporte para otras clases
- Su uso de fundamenta en la cache del navegador

RSL's II

- ¿Cuándo las tengo que usar?
 - Cuando múltiples aplicaciones utilicen partes común de código
 - Cuando una parte del código sea susceptible de cambiar de forma más rápida que el resto
 - I.e: frameworks
 - I.e: gateways

RSL's III

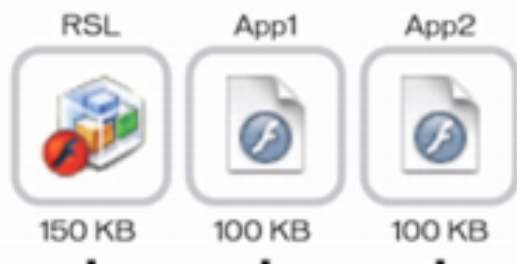
- ¿Qué ventajas consigo?
 - En caso de utilizarlas en múltiples aplicaciones el usuario notará una reducción sustancial en el total de Kb a descargar
 - Agilidad en el mantenimiento evolutivo de aplicaciones.
 - El código que exponen estará disponible en la aplicación en startup.

Without RSLs
(Using statically-linked
component libraries)



Flash Player

With RSLs



RSL's IV

- ¿Qué problemas introducen?
 - Tamaño considerable -> penalizará la primera carga.
 - Si hay distintas aplicaciones que utilicen la misma RSL se tendrá que ofrecer un mantenimiento coherente y controlado que no introduzca bugs colaterales.

Modules

- Característica nativa para modularizar aplicaciones Flex
- Un módulo es un conjunto de clases no necesario en el momento de arranque de la aplicación. Suelen requerirse on-demand o bien de forma dinámica y parametrizada en el arranque.
- Conceptualmente igual que loadMovie en flash

Modules II

- ¿Cuándo las tengo que usar?
 - Aplicaciones grandes. No todo el código siempre es necesario. Lazy loading de prestaciones de nuestra aplicación
 - Carga dinámica de funcionalidades de la misma naturaleza pero con distinta implementación (Factories funcionales)
 - Cuando no se quiere recompilar el 100% de la aplicación

Modules III

- ¿Qué ventajas consigo?
 - Tiempo de startUp. cuanto más grande es la aplicación más código tiene que cargar antes de poderse activar. El uso de módulos lo evita.
 - Optimización de la performance (uso de memoria inferior ya que no está todo el código cargado).
 - Responsabilidad distribuida -> alta cohesión (responsabilidades muy acotadas), bajo acomplamiento (pocas interdependencias) -> fomento arquitectural

Modules IV

- ¿Qué ventajas consigo?
 - Fomento de la programación orientada a Interface -> polimorfismo -> mayor rendimiento (menor tiempo de instanciación debido a una cadena de herencia menor), reducción de dependencias...
 - Bug fixing acotado
 - Reducción sustancial del tiempo de compilación

Técnicas de modularización

- Shell / Module
 - Shell: Código necesario en startUp
 - Module: Código necesario on-demand
 - La shell carga modules bajo demanda
 - La shell puede ofrecer una API de comunicación unidireccional / bidireccional con los módulos
 - La shell puede aportar una API de intercomunicación entre módulos
 - Todo basado entre Interfaces debido al funcionamiento nativo de los applicationDomains. Se establece una relación contractual a través de Ifaces. La shell no conoce la especialización del módulo

Técnicas de modularización II

- En una aplicación es común combinar ambas técnicas:
 - Proyecto aplicación
 - N Proyectos para los módulos
 - Proyecto commons

Técnicas de modularización III

- Opciones de compilación
 - `--link-report`. Nos aporta un listado de todas las clases linkadas a una aplicación. Esto es, dada una aplicación X qué clases usa?
 - `--load-externs`. Excluye todas las clases definidas en el report especificado (`link-report`)
 - `--externs`. Define una lista de clases que se excluirá de la compilación
 - `--external-library-path`. Excluye todas las clases de la librería (`swc`)
 - Opciones extremadamente útiles cuando sabemos que la aplicación que desarrollamos se va a ejecutar en un entorno determinado (que muy seguramente ya tendrá muchas clases cargadas. En estos casos eliminamos / unlink esas clases para mejorar el tamaño de la aplicación final

Ejemplo

- Creación de una pequeña aplicación de medias:
 - Existirá un módulo para cada tipo de media
 - ImageModule
 - AudioModule
 - VideoModule
 - La shell se comunicará con el módulo a través de una relación contractual a través de una interface con un método loadMedia (file:ResourceVO)
 - Tendremos un proyecto commons compartido con la clase ResourceVO

Ejemplo II

- Aplicación gateway
 - Tenemos una aplicación X. Esta aplicación tiene que ser capaz de guardar y leer datos
 - Dependiendo de la integración que se haga se podrá leer y guardar de distintas formas:
 - SOAP
 - RemoteObject
 - FDS
 - Si desarrollamos a producto es muy conveniente el poder evolucionar la aplicación de forma independiente a su comunicación con el exterior (es variable, otro cliente lo puede querer de forma distinta).

Referencias

- Roger Gonzalez:
 - <http://blogs.adobe.com/rgonzalez/2006/06/applicationdomain.html>
 - http://blogs.adobe.com/rgonzalez/2006/11/my_max_preso.html
- Alex Harui
 - <http://blogs.adobe.com/aharui/2007/03/modules.html>
- Livedocs
 - http://livedocs.adobe.com/flex/201/html/rsl_124_3.html
 - http://livedocs.macromedia.com/flex/201/html/modular_083_1.html