



Flex 3 SDK:

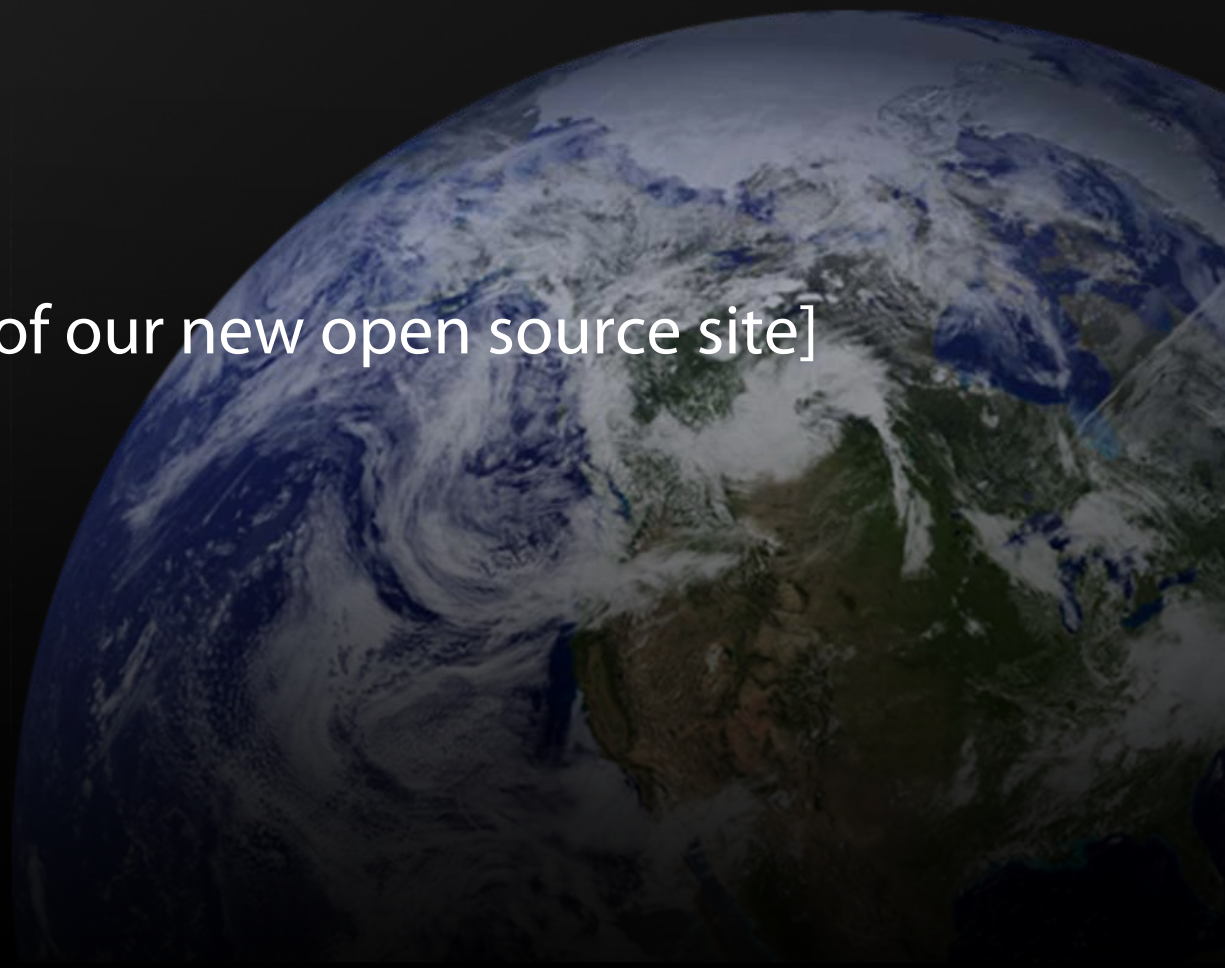
A deep dive tour into open source Flex SDK

Adobe



<http://opensource.adobe.com>

[A very quick tour of our new open source site]



SVN Repository

- The Flex team switched to **SVN** for our open source effort
- Our subversion repository base URL:
<http://opensource.adobe.com/svn/opensource>
 - ***Flex SDK 3.0.x Branch***
[/flex/sdk/branches/3.0.x/](#)
 - ***Beyond Flex SDK 3 - our “mainline” or “trunk”***
[/flex/sdk/trunk/](#)
- We’re currently using TortoiseSVN as a client for Windows, other clients like SmartSVN for other OS’es. Haven’t really needed SVN Eclipse integration.

SVN Repository

[A quick tour of our SVN repository... checking out source,
how to run our build script]

Build Process

- Flex SDK trunk currently requires **Ant 1.7.0** and **JDK 1.5.0**
- In general, see the **README.txt** in the root directory of any Flex SDK branch for instructions
- Typically sync to head revision and run '**ant**' from the command line
- You need to run the ant build in order to create the necessary binaries before using the SDK from SVN or importing the Eclipse projects

Questions about SVN or the build?

Eclipse IDE

- Flex 3 targeted JDK 1.4.2 (latest version is recommended) - **going forward** we're targeting **JDK 1.5.0**
- Flex team is typically using **Eclipse Europa** (3.3)
 - ***Eclipse Projects in SVN under /development/eclipse***
e.g. ***/flex/sdk/branches/3.0.x/development/eclipse***
- Code Formatting Guidelines also in this folder – simply import *formatter.xml* into your Java settings
(quick points: tabs as 4 spaces, braces on the next line)
- Also check out coding conventions

<http://opensource.adobe.com/wiki/display/flexsdk/Developer+Documentation>

Eclipse IDE

[A quick look at setting up Eclipse...]

Questions about Eclipse IDE setup?

Frameworks Source

- Much of the framework.swc source has been available before today, though now you have access to internal classes
- But now you have access to other sources – check out the rpc.swc project
 - WSDL or just plain XML Schema / Typed Object Mapping
 - Write your own custom Channels
 - New high-level services abstractions

MXML Compiler

- **MXMLC** is really a **collection of subcompilers** that ultimately produces byte code in a SWF
- The main subcompilers are for .mxml and .as files
- Others are used to include precompiled ABC, or compile runtime CSS, or compile translation resource files

MXML Compiler

- Each subcompiler implements an interface that defines a **series of phases** (preprocess, parse1, parse2, analyze1, analyze2, analyze3, analyze4, generate, postprocess)
- Most of these line up with how **ASC** is designed
- Each compiler takes a Source (a single “file”) which is parsed into a CompilationUnit

MXML Compiler

- The CompilationUnit is then passed to the other phases...
- The **analyze** phase largely resolves the **type dependencies** of the CompilationUnit
- The **generate** phase emits **byte code**
- Most of the subcompilers wrap an AS3 compiler. In each phase, some work is done and then the same phase is called on the AS3 compiler

MXML Compiler

- flex2.compiler.API **orchestrates** all of the subcompilers and handles the **batch compilation** of the files that make up an Application
- It also handles Flex Builder's **incremental compilation** mode by persisting and restoring CompilationUnits to disk

MXML Compiler Extensions

- There are also several compiler extensions to handle Data Binding, Embed, CSS
- Let's take a look at one of these extensions.

Data Binding

- Two parts to **data binding**
- The first part is the in-line data binding expressions (i.e. **{person.name}** curly braces) and the **<mx:Binding>** tag
- The compiler keeps track of these by adding **BindingExpressions** to the **MxmlDocument** context to make them available to the **DataBindingFirstPassEvaluator**

Data Binding

- The second part involves the variables / properties with **[Bindable] metadata** that report what events are dispatched when the property changes
- These metadata advertises to the `DataBindingFirstPassEvaluator` that data binding expressions can listen for these change events
- For each property that can be **watched**, a `Watcher` is created and used by a Velocity template to generate the AS code to set them up at runtime

Some other uses of the Flex SDK?

ASC:

- Optimizers
- Obfuscators?

SWF Utils:

- Link Reports
- Media Transcoders
- SVG
- SWFX and SWF Dump

JIRA – Flex Bug Base

- As you know, Flex moved to a **JIRA** based **bug tracking system** for Flex 3
 - <http://bugs.adobe.com/flex/>
- Now Flex **community members** who have submitted the **Contributor Agreement** will be able to attach **.patch** files to bugs in JIRA to **contribute** fixes

Final Questions?