MAX

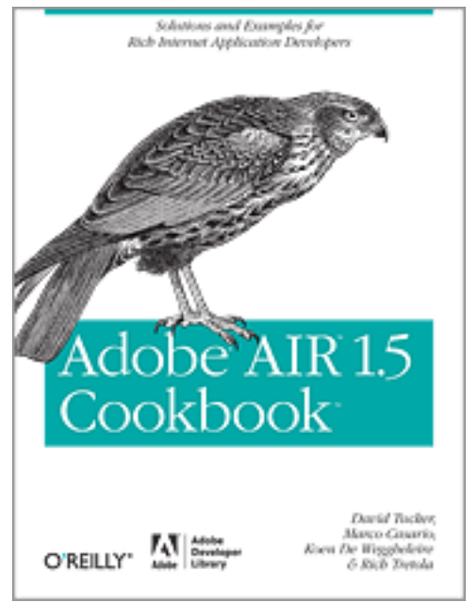
Adobe Presentation

David Tucker

Maven and Jenkins for Enterprise Flex Applications

About Me

- Technical Architect at Universal Mind
- Adobe Community Professional
- Lead Author, Adobe AIR 1.5 Cookbook (O'Reilly)
- Author, AIR for Flash Developers (Lynda.com)
- Author, RIA's that Rock Column (Adobe's NewsFlash Newsletter)
- My Fourth Adobe Max
  - Max Master 2010
  - 2 Time Max Award Finalist
  - 3 Time Speaker



Goal

The goal of today's session is to present you with **introductory** knowledge of **Maven**, **Flex-mojos**, and **Jenkins.** At the end of the session you should know how to:

- Setup multiple types of Flex projects with Maven
- Run unit tests with Maven
- Build a Flex application with Maven
- Integrate your Maven project with Jenkins for continuous integration

Prerequisites

- Intermediate Flex Knowledge
  - We are going to deal with pre-existing code for all of the samples today. I will not be explaining how this code was created or how it works.
- Intermediate FlashBuilder Knowledge
  - I'll be covering how to integrate with Maven, but I'll only briefly discuss how this integrates with FlashBuilder. This will be mainly left up to the attendees to determine.

Source Code

All source code from today's session can be found here:

https://github.com/davidtucker/Adobe-Max-2011-Maven-Samples

Agenda

- What is Enterprise?
- Introduction to Maven and Flex-Mojos
- Introduction to Continuous Integration with Jenkins
- Available Resources
- Conclusion

What is Enterprise?

What is Enterprise?

- Enterprise development is different:
  - It requires processes that are repeatable, defined, and self-analyzing
  - It includes many different people in the process from developers, architects, designers, UX specialists, QA engineers, project managers, product managers, executives, customers, etc....
  - It includes multiple departments which may be in multiple locations building different solutions on a single platform
  - It can include life or death implementations (for example – healthcare related applications)

Introduction to Maven

What is Maven?

- Dependency Management
  - Easily manage all of the dependencies on your project (such as your framework SWC)
- Build Automation
  - Follows a standard process for building and structuring applications
- Project Management
  - Uses a standard approach for all projects which can be used to enforce standards across an organization

Maven vs. Ant

- Maven
  - Provides a standard approach and structure for all applications
  - Requires the creation of your POM file per project (and per module)
  - No configuration required for the build process
  - Easily extensible using third-party or custom plugins
  - Includes support for unit testing
- Ant
  - Provides complete customization over the process
  - Requires the creation and maintenance of ANT build scripts for your project
  - Extensibility is available but will need to be written into existing scripts
  - Requires unit testing to be written into build scripts

Why Avoid Maven?

- You need to not use Maven if:
  - You want complete control every aspect of the build process
  - You have your own set of standards in the project structure that you are not willing to change
  - You are not willing to work with the Maven repository and dependency management

Maven Glossary

- A file produced by a Maven build is referred to as an **artifact.** Every artifact is identified by four different properties:
  - **groupId** – Generally follow the reverse-DNS style naming. For example `net.davidtucker.max2011`
  - **artifactId** – The name of the specific artifact. For example, `utility-lib`
  - **version** – The version number of the artifact. Development builds usually have the `-SNAPSHOT` suffix
  - **packaging** – The type of the artifact. Types include swc, swf, jar, pom, war, etc...
- These values (and many others) are defined in a **POM**, Project Object Model, for each artifact.
- Artifacts are stored in a **repository.** A repository can be remote or local.

The Project Object Model (POM)

- The POM is the XML configuration file that tells Maven about your project. It tells Maven:
  - How your project should be built
  - How your project files are structured
  - How it relates to other local projects
  - What dependencies your project has
  - What Maven plugins are used for your project
  - What configuration options are used for your project

What is Flex-Mojos?

Flex-Mojos provides Maven integration for Flex projects. It is a Maven plugin that takes the configuration options from your POM and builds your application using its own version of the Flex SDK.

Website: http://flexmojos.sonatype.org/

Setting Up Your Machine for Flex-Mojos

- Add your Flash Player Debug Standalone instance to your PATH (Required for unit testing integration)
  - Download: http://www.adobe.com/support/flashplayer/downloads.html
- Add the Flex-mojos repository to your Maven `settings.xml` file (OPTIONAL)
  - Sample Settings file can be found in the Github code samples

Using Archetypes

Creating an application from an archetype is as simple as executing a command on the command line:

Generating an Application Project:

```
mvn archetype:generate \
-DarchetypeRepository=http://repository.sonatype.org/content/groups/flexgroup/ \
-DarchetypeGroupId=org.sonatype.flexmojos \
-DarchetypeArtifactId=flexmojos-archetypes-application \
-DarchetypeVersion=3.9
```

Generating a Library Project:

```
mvn archetype:generate \
-DarchetypeRepository=http://repository.sonatype.org/content/groups/flexgroup/ \
-DarchetypeGroupId=org.sonatype.flexmojos \
-DarchetypeArtifactId=flexmojos-archetypes-library \
-DarchetypeVersion=3.9
```

DEMO 1: Hello World with Flex-Mojos

Demo 1 Summary

- In this example, we did the following:
  - Used an archetype to create a project from scratch
  - Examined and modified the created POM for the project
    - Changed the SDK version for our project
    - Set the `flashPlayer.command` value
  - Used the `mvn clean install` command to build our project

Adding in Unit Testing with FlexUnit 4

- Flex-mojos integrates the existing unit testing support in Maven with FlexUnit
  - This integration is automatic. If properly setup there will be no need to create a test runner or manually include libraries. It works in the Maven way.
- **Advanced Tip:** Newer versions of Flex-mojos (currently the 4.0-SNAPSHOT support code coverage reporting utilizing Apparat and not Flex Cover). **We will not cover this today.**

DEMO 2: Adding Unit Testing to a Single Project

Demo 2 Summary

- In this example, we did the following:
  - Modified the existing POM file to add support for FlexUnit 4
  - Placed our FlexUnit4 Unit Tests in the `/src/test/flex` directory
  - Placed our test resources in `/src/test/resources` directory
  - Used the `mvn clean test` command to run our unit tests

Multi-Module Projects

- Maven has support for multi-module projects (think generic modules not Flex modules in this case)
- POM's can have a hierarchical structure with a parent POM controlling a set of child module POM's
- This structure can work for building library projects with application projects
- Maven builds the child modules in the correct order based on how you define the dependencies

DEMO 3: Creating a Library Project for our Existing Application

Demo 3 Summary

- In this example, we did the following:
  - Moved our reusable elements of code into a separate project
  - Created a parent POM to manage the child modules
  - Modified the current POM file for each of the child modules to reference the parent POM
  - Used the `mvn clean install` command to test and build our application

Continuous Integration

Why Continuous Integration?

- Automated Builds
  - Manual builds have an entire set of problems. In an ideal world, you always take the human factor out of the build process.
- Automated Testing
  - Unit tests (and functional tests if possible) should be run continually. In an ideal world this happens everything a developer checks in code.
- Automated Analytics
  - Tools like FlexPMD, FlexCPD, and FlexMetrics (as well as non-Flex analytics tools) can help you get an insight into your code especially as you monitor the changes over time.

Continuous Integration Options

- Many options existing for Continuous Integration servers:
  - **Jenkins (and also Hudson)** – An open-source Java based continuous integration server with a healthy plugin development community
  - **Atlassian Bamboo** – A commercial server with a large collection of plugins which can also tie directly into other Atlassian tools such as JIRA for Release Management
  - **Cruise Control** – An older open-source CI server which works well but requires a lot of manual configuration to get your projects up and running
  - **Team City** – A commercial solution from JetBrains (the company who creates IntelliJ)
- I have used Jenkins and Bamboo extensively and would recommend both of them.

Why Jenkins?

Jenkins (formerly Hudson) provides an easy entry point into Continuous Integration because of it open-source model and extensive plugin library. It addition, it comes with Maven integration.

Site: http://jenkins-ci.org/

Plugins: https://wiki.jenkins-ci.org/display/JENKINS/Plugins

Platform Considerations

- Flex-mojos requires that the machine have the Flash Player debug version to run unit testing
- The Flash Player requires a display to run
- Virtual displays can be used on headless Linux boxes (but that is beyond the scope of this talk)
- Adobe has announced that after AIR 2.6, you won't be able to fully utilize the AIR SDK on Linux
- **Recommendation:** Using Windows for your build / CI server with a cloud service like Amazon EC2 is the easiest way to get up and running for your organization. It will also work fine for AIR and Flex applications.

DEMO 4: Automatically Building and Testing our Flex Application

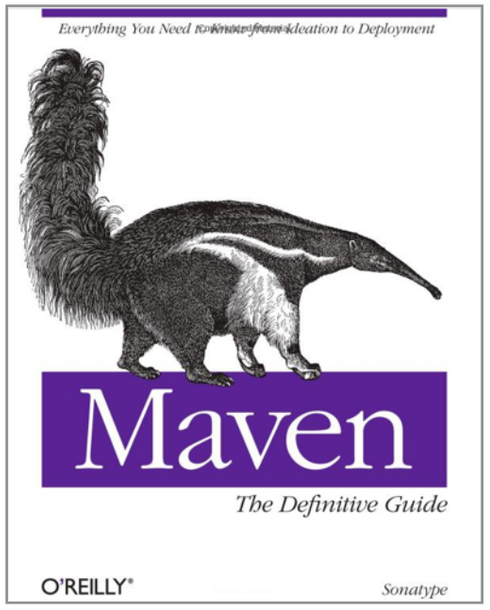
Demo 4 Summary

- In this example, we did the following:
  - Installed Jenkins using the Native Package installer
  - Installed a few plugins:
    - Git plugin – Provides Git Integration
    - Green Balls – Fixes an annoying design aspect of Jenkins
  - Created a build job that hooks into the existing Github repository
  - Configured the build job to point to the parent POM for our project
  - Configured the build job to keep the needed artifacts from the build proces

Resources

Maven Resources

- Flex and Maven with Flex-Mojos
  - Authored By Justin Moses
  - Link: http://www.adobe.com/devnet/flex/articles/flex-maven-flexmojos-pt1.html
- Maven: The Complete References
  - Link: http://www.sonatype.com/books/mvnref-book/reference/
- Flex-Mojos
  - Link: https://docs.sonatype.org/display/FLEXMOJOS/Home



Jenkins / Continuous Integration Resources

- Jenkins: Meet Jenkins
  - Link: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins
- Continuous Integration
  - Link: http://martinfowler.com/articles/continuousIntegration.html

MAX

CONNECT. DISCOVER. INSPIRE.