Intro to Maven

http://github.com/ostewart/maven-training

Oliver Stewart

Contact

Oliver Stewart

oliver@trailmagic.com

@sintactic

http://github.com/ostewart

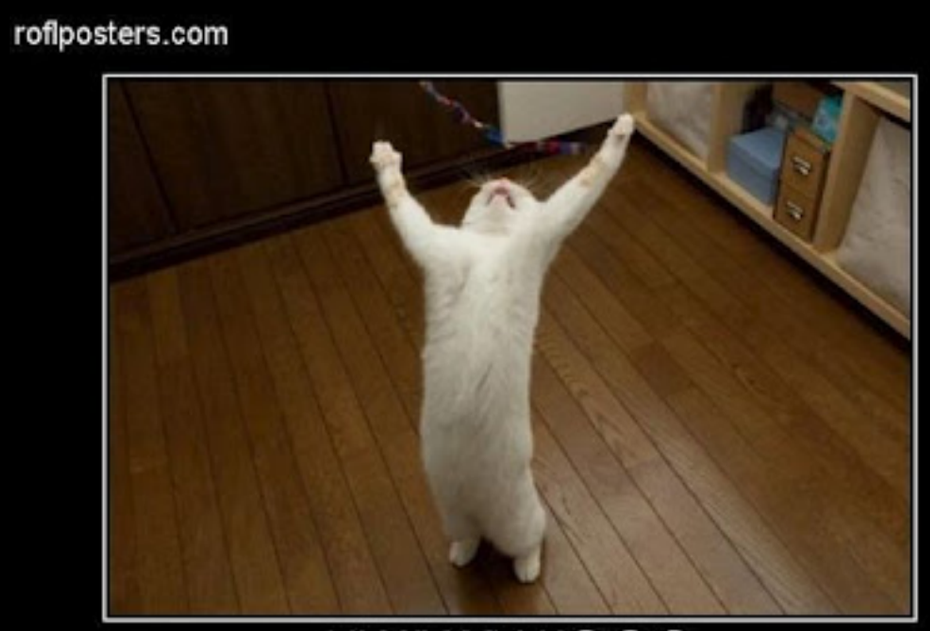
Agenda

- Why Maven?
- The Maven Way
- Maven Basics
- Multi-Module Projects
- Web Projects
- Integration Testing
- Clean/Advanced Maven

What Is Maven?

- Build tool (focused on JVM)
- Convention over configuration
- Structured project model
- Build code reuse

Why Maven?



Shanty Town Builds

- Rarely get focused attention
- Haphazard areas of focus
- Each is unique

http://dornob.com/urban-shanty-town-photo-collages-of-sao-paolo/#axzz2aADkkp00

Regular Structure

- Approachable
- Machine-readable
- Codifies good practice
- Supports collective ownership

Photo: Russ Allison Loar
http://bit.ly/14flwzd

Good Practices

- Separate code, tests, resources
- Platform independence
- Declarative dependency management
- Share resources via dependencies
- Predictable directory structure
- Test built artifacts

Why Not Maven?

- General automation
- Legacy code with intricate build incantations
- Might be able to bridge with AntRun or custom plugins

The Maven Way

- Don't fight Maven. You'll lose.
- Maven is a box that turns source code into build artifacts
- First, try to see how your problem fits into Maven's worldview
- Next, you might want to write a plugin or use AntRun
- If you're still struggling, you may want to consider another tool

Maven Basics

Creating a New Project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example.maventraining</groupId>
  <artifactId>simple</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Running Maven

```
mvn <phase or goal> [<phase or goal>...]
```

Maven Lifecycles

- Clean
- Default
- Site

Default Lifecycle (Abridged)

- Validate
- Compile
- Test
- Package
- Integration-test
- Verify
- Install
- Deploy

Dependency Management

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Coordinates: groupId, artifactId, version, and optional scope

Dependency Scopes

- compile (default)
- provided
- runtime
- test
- system
- import

Snapshot Versions

- Good for intra-project dependencies during development
- Will check for updates each time you build
- Will break your build eventually outside a single source tree
- x.x-SNAPSHOT

Release versions

- Generated by the release plugin
- Assumed immutable (cached aggressively)

Directory Layout

`src/main/java`	Application/Library sources
`src/main/resources`	Application/Library resources
`src/main/webapp`	Web application sources
`src/test/java`	Test sources
`src/test/resources`	Test resources
`target`	Build output

Super POM

- The POM from which all other POMs inherit
- Look here for defaults
- http://maven.apache.org/ref/3.0.4/maven-model-builder/super-pom.html

Image source: examiner.com

Exercise

Build a simple library that outputs "Hello, World!". Write a HelloWorld class that outputs a String and a test that invokes it. Build and test your project with Maven.

More Basics

Plugins
Packaging

Plugins

- Maven's main unit of modularity
- How most of Maven's functionality is implemented and configured

Plugin Configuration

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Built-In Plugins

- maven-compiler-plugin
- maven-surefire-plugin
- maven-dependency-plugin
- maven-release-plugin

Common Plugins

- AntRun
- Cargo
- Jasmine
- Exec
- PMD
- Cobertura
- Findbugs

Packaging

- Defines the artifact type
- Binds goals to phases
- Core values: pom, jar, maven-plugin, ejb, war, ear, rar, par
- Default is jar

Packaging Binds Goals to Phases

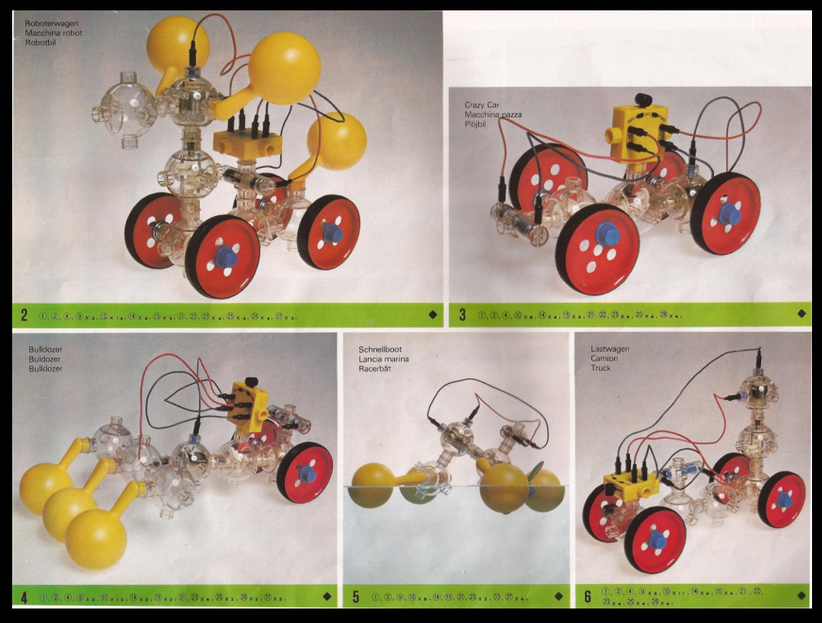
Phase	Goal
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar
install	install:install
deploy	deploy:deploy

Exercise

Add isEmpty check on a String, set source/ target compiler versions to 1.6

Multi-Module Maven

- Why?
- How?
- Dependence vs. Inheritance
- Naming



Why Multiple Modules?

- Group related functionality
- Isolate unrelated functionality
- Organize dependencies
- Unit for sharing code
- Unit of packaging

Multi-Modules: How

- Top-level pom packaging with modules
- Optional parent in sub-modules

Modules

```
<modules>
  <module>hello-util</module>
  <module>hello-service</module>
</modules>
```

Parent

```
<parent>
    <groupId>com.example.maventraining</groupId>
    <artifactId>hello</artifactId>
    <version>1.0-SNAPSHOT</version>
</parent>
```

Dependence vs. Inheritance

- Use dependencies to share code, resources
- Use inheritance to aggregate common settings

Naming

- "There are only two hard things in Computer Science: naming, cache invalidation, and off-by-one errors." - almost Phil Karlton

- All modules are top-level, referenced by groupId-artifactId

- http://maven.apache.org/guides/mini/guide-naming-conventions.html

Exercise: Multi-Module

Now that we have a Hello utility, we'd like to use it to greet people on our roster of users. For each name on our roster, output "Hello, <name>!". While the Hello utility might be generally useful, this greeting service is specialized functionality, so it should go in its own module.

Bonus

- Web modules
- Integration testing with Cargo
- Dependency management
- Properties
- Profiles

Web Modules

- war packaging
  - Static resources in `src/main/webapp`
  - `web.xml` in `src/main/webapp/WEB-INF`

Exercise: Hello Web

We'd like to expose our HelloWorld utility as a web service. Write a simple servlet that extracts the name from the URL and serves the hello response as plain text.