

Maven Intermediate Concepts

Customizing the build process



Overview



- ▶ Maven plugins, and customizing the build lifecycle using plugins
- ▶ Customizing your build process with properties and filters
- ▶ Using build profiles

Properties and Profiles

Properties and Profiles



- ▶ Using properties and build profiles to customize your build
 - ▶ Properties let you parameterize and refactor your builds
 - ▶ Build profiles let you adapt your build process to different circumstances

Defining properties



- ▶ Properties can be defined in many places
 - ▶ Maven project properties
 - ▶ Maven settings properties
 - ▶ Java system properties
 - ▶ Environment properties
 - ▶ Arbitrary properties

Using POM variables



- ▶ Properties can refer to elements in the POM file itself
 - ▶ For example
 - ▶ *Maven coordinates:* \${project.groupId}, \${project.artifactId}, \${project.version},...
 - ▶ *Project name and description:* \${project.name}, \${project.description}
 - ▶ *Build directories:*
 - ▶ \${project.build.sourceDirectory} and \${project.build.testSourceDirectory},
 - ▶ \${project.build.outputDirectory} and \${project.build.testOutputDirectory}
 - ▶ ...

```
<dependencies>
  <dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>killerapp-core</artifactId>
    <version>${project.version}</version>
  </dependency>
</dependencies>
```

Properties in POMs



- ▶ You can use properties in your POM files to customize your build
 - ▶ Refactor and centralize frequently-used variables
 - ▶ Access environment variables in your build script
 - ▶ Refer to elements in the POM itself
- ▶ Properties use the familiar \${...} syntax:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency>
```

Property value

Using settings.xml variables



- ▶ Properties can be read from the settings.xml file, e.g.

settings.xml property	Description
settings.localRepository	Path to the local repository
settings.offline	Is Maven running in offline mode
settings.proxies	A list of the proxies defined in the settings.xml file
settings.servers	A list of the servers defined in the settings.xml file
settings.mirrors	A list of the mirrors defined in the settings.xml file
settings.profiles	A list of the profiles defined in the settings.xml file
...	

Using settings.xml variables



- ▶ Getting properties from the settings.xml file - some examples
- ▶ Passing the local repository path to an Ant script

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <id>process-widgets</id>
      <phase>validate</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <tasks>
          <ant target="process-widgets">
            <property name="localRepository"
                      value="${settings.localRepository}" />
          </ant>
        </tasks>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Pass the local repository path in the settings.xml file to an ant script

Using settings.xml variables



- ▶ Getting properties from the settings.xml file - some examples
- ▶ Retrieving the username and password from a server definition

```
<plugin>
  <groupId>org.codehaus.groovy.maven</groupId>
  <artifactId>gmaven-plugin</artifactId>
  <version>1.0-rc-5</version>
  <executions>
    <execution>
      <id>process-db-scripts</id>
      <phase>pre-integration-test</phase>
      <goals>
        <goal>execute</goal>
      </goals>
      <configuration>
        <source>
          def server = settings.servers.find{ it.id.equals('staging-database') }
          """ant -Ddb.username=${server.username}
          -Ddb.password=${server.password}""".execute()
        </source>
      </configuration>
    </execution>
  </executions>
</plugin>
```

```
<settings>
  ...
  <servers>
    ...
    <server>
      <id>staging-database</id>
      <username>scott</username>
      <password>tiger</password>
    </server>
  </servers>
</settings>
```

settings.xml

Fetch the 'staging-database' server entry in settings.xml

Pass the username and password defined here to an Ant script

Java System properties



- ▶ You can also access Java System Properties, e.g.

System property	Description
java.version	Java Runtime Environment version
user.name	User's account name
user.home	User's home directory
user.dir	User's current working directory
os.name	Operating system name
java.io.tmpdir	Directory for temporary files
...	

System properties



- ▶ Command-line system properties are passed in using the -D option

```
$ mvn deploy -Dwar.version=1.2.4
```

```
<dependencies>
  <dependency>
    <groupId>com.training.killerapp</groupId>
    <artifactId>killerapp-war</artifactId>
    <version>${war.version}</version>
    <type>war</type>
  </dependency>
</dependencies>
```

User-defined properties



- ▶ User-defined properties can be set up in a POM or in settings.xml
- ▶ Properties are defined in the <properties> section
- ▶ Often used to centralize version numbers used in several places

```
<dependencies>
    ...
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>${junit.version}</version>
        <scope>test</scope>
    </dependency>
</dependencies>
...
<properties>
    <junit.version>3.8.1</junit.version>
</properties>
```

Properties are defined here

Using environment variables



- ▶ Environment variables can be accessed using the “env” prefix
- ▶ \${env.HOME}, \${env.JAVA_HOME},...
- ▶ Environment variables can be used to hide sensitive data:

```
$ export DB_USERNAME=root  
$ export DB_PASSWORD=s3cr3t  
$ mvn deploy
```

```
<configuration>  
  ...  
  <username>${env.DB_USERNAME}</username>  
  <password>${env.DB_PASSWORD}</password>  
</configuration>
```



mvn help:system

Using Java version: 1.6

[INFO]

NOTE: Maven is executing in offline mode. Any artifacts not already in your local repository will be inaccessible.

[INFO] Scanning for projects...

[INFO] Searching repository for plugin with prefix: 'help'.

[INFO] -----

[INFO] Building Maven Default Project

[INFO] task-segment: [help:system] (aggregator-style)

[INFO] -----

[INFO] [help:system {execution: default-cli}]

[INFO]

==== Platform Properties Details ====

System Properties

java.runtime.name=Java(TM) SE Runtime Environment

sun.boot.library.path=/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Libraries

java.vm.version=14.3-b01-101

awt.nativeDoubleBuffering=true

gopherProxySet=false

mrj.build=10M3025

java.vm.vendor=Apple Inc.

java.vendor.url=http://www.apple.com/

path.separator=:

java.vm.name=Java HotSpot(TM) 64-Bit Server VM

file.encoding.pkg=sun.io

sun.java.launcher=SUN_STANDARD

user.country=US

:

```
mrj.version=1060.1.6.0_17-248
```

```
sun.cpu.isalist=
```

Environment Variables

```
DISPLAY=/tmp/launch-Z3AIY3/:0
GRAILS_HOME=/Applications/Dev/grails
SOAPUI_HOME=/Applications/Dev/soapui-2.5
SSH_AUTH_SOCK=/tmp/launch-YN4iEc/Listeners
NEXUS_HOME=/Applications/Dev/nexus/nexus-webapp
MYCOURSES=/Users/mccm06/Documents/Teach/Courses
CXF_HOME=/Applications/Dev/apache-cxf-2.2
PATH=/opt/local/bin:/opt/local/sbin:/System/Library/Frameworks/JavaVM.framework/Versions/1.6/Commands
:/Applications/Dev/btrace/bin:/opt/local/bin:/opt/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local
/bin:/usr/X11/bin:./:/Users/mccm06/scripts:/Applications/Dev/apache-ant/bin:/Applications/Dev/apache-
maven/bin:/Applications/Dev/groovy/bin:/Applications/Dev/grails/bin:/Applications/Dev/nexus/nexus-web
app/bin/jsw/macosx-universal-32:/Applications/Dev/jakarta-jmeter/bin:/Applications/Dev/jruby/bin:/App
lications/Dev/soapui-2.5/bin:/Applications/Dev/appengine-java-sdk/bin:/Applications/Dev/hadoop-family
/hadoop-0.20.1/bin:/Applications/Dev/hadoop-family/pig-0.5.0/bin:/Applications/Dev/hadoop-family/avro
-1.2.0/bin:/Applications/Dev/hadoop-family/chukwa-0.3.0/bin:/Applications/Dev/hadoop-family/hbase-0.2
0.2/bin:/Applications/Dev/hadoop-family/hive-0.4.1-bin/bin:/Applications/Dev/hadoop-family/zookeeper-
3.2.2/bin:/Applications/Dev/apache-tomcat/bin:/Applications/Dev/ec2-api-tools-1.3-46266/bin:/Applicat
ions/Dev/apache-tomcat-6.0.20-ibean/mule-ibean/bin:/Applications/Dev/sonar/bin/macrosx-universal-64:
/Applications/Dev/Bamboo:/Applications/Dev/mvnsh/bin:/Applications/Dev/pmvn/bin:/Library/PostgreSQL
8/bin
LOGNAME=mccm06
SHLVL=1
JAVA_MAIN_CLASS_58079=org.codehaus.classworlds.Launcher
TERM_PROGRAM=Apple_Terminal
ANT_HOME=/Applications/Dev/apache-ant
MYPROCESS=/Users/mccm06/Documents/Temp/Process
CLOJURE_HOME=/Applications/Dev/clojure
:|
```



filters

Using filters on Resources



- ▶ What are resources?
 - ▶ Resources are non-Java files to be bundled with the application
 - ▶ Properties files
 - ▶ Configuration files
 - ▶ Message bundles
 - ▶ Templates
 - ▶ ...
 - ▶ Resources are placed in `src/main/resources`
 - ▶ When packaged, they are placed on the application classpath

Using filters on Resources

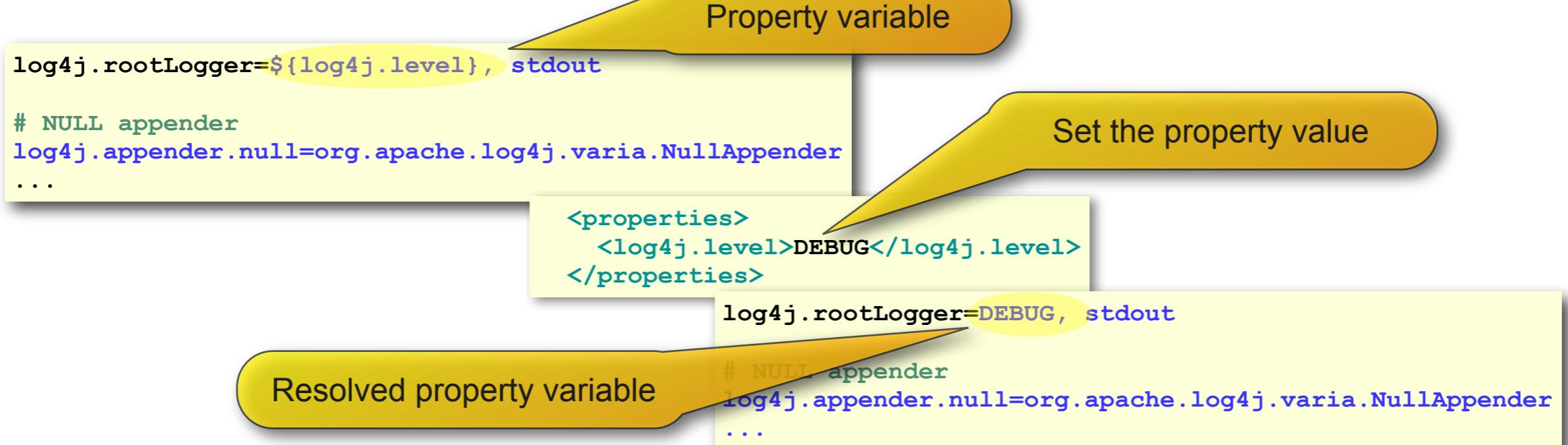


- ▶ What are test resources?
 - ▶ Resources required for unit and integration tests
 - ▶ Placed in `src/test/resources`
 - ▶ Not deployed with the application
 - ▶ Can be used to override application resources during testing

Using filters on Resources



- ▶ Filtering resources
 - ▶ You can embed properties in your resources
 - ▶ Property values come from the usual places
 - ▶ pom.xml, settings.xml, environment variables,...
 - ▶ Or from a filter properties file
 - ▶ They will be resolved during the *process-resources* phase



Using filters on Resources



- ▶ Enabling filtering
- ▶ Filtering disabled by default
- ▶ You need to activate it in your POM file

```
<project>
  ...
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <filtering>true</filtering>
      </resource>
    </resources>
    <testResources>
      <testResource>
        <directory>src/test/resources</directory>
        <filtering>true</filtering>
      </testResource>
    </testResources>
  ...

```

Enable filtering on the main resource directory

Enable filtering on the test resource directory

An alternate approach

- ▶ Enabling filtering
- ▶ Create a new folder specifically for filtering

```
<project>
  ...
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <filtering>false</filtering>
      </resource>
      <resource>
        <directory>src/main/filtered-resources</directory>
        <filtering>true</filtering>
      </resource>
    </resources>
  ...

```

Leave Filtering disabled
on the main resource

Enable filtering on the
filtered - resource directory



labs

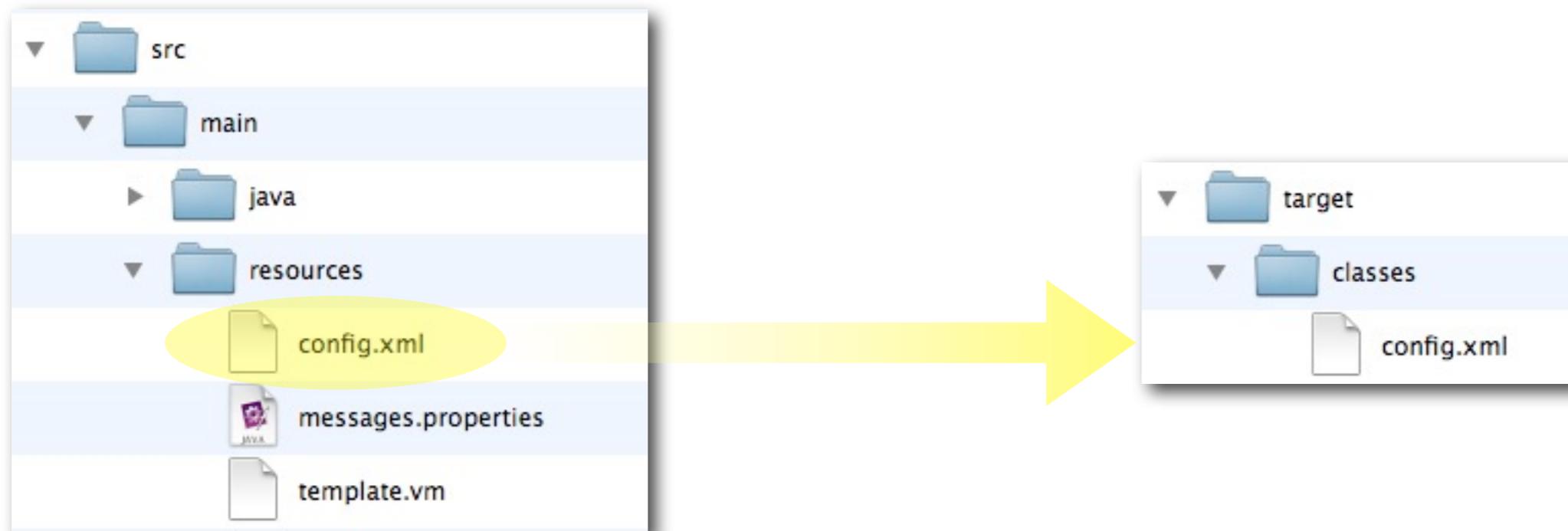
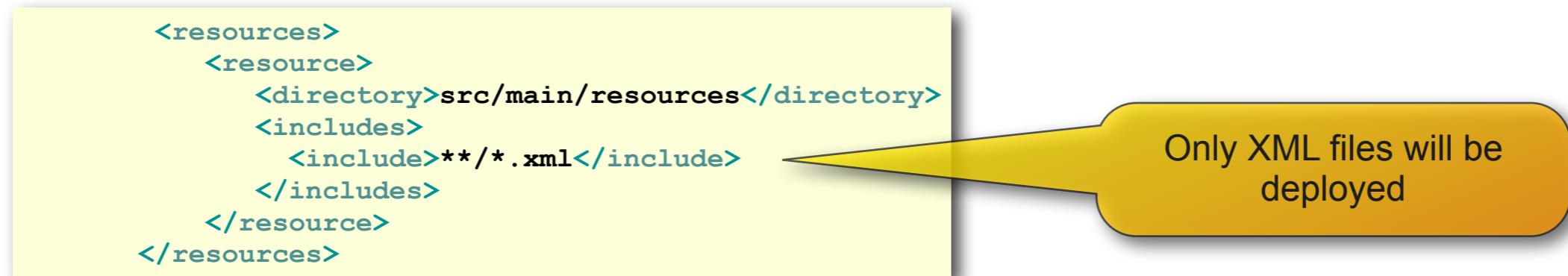
- ▶ Lab 10 - Working with properties and filters



Including & Excluding Resources

Including/excluding Resources

- ▶ What files do you want to include?
- ▶ Include and exclude certain files in the resources



Including/excluding Resources



- ▶ Selective filtering
- ▶ Attention: here, only the included files will be deployed

```
<project>
  ...
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
        <filtering>true</filtering>
      </resource>
    </resources>
  ...

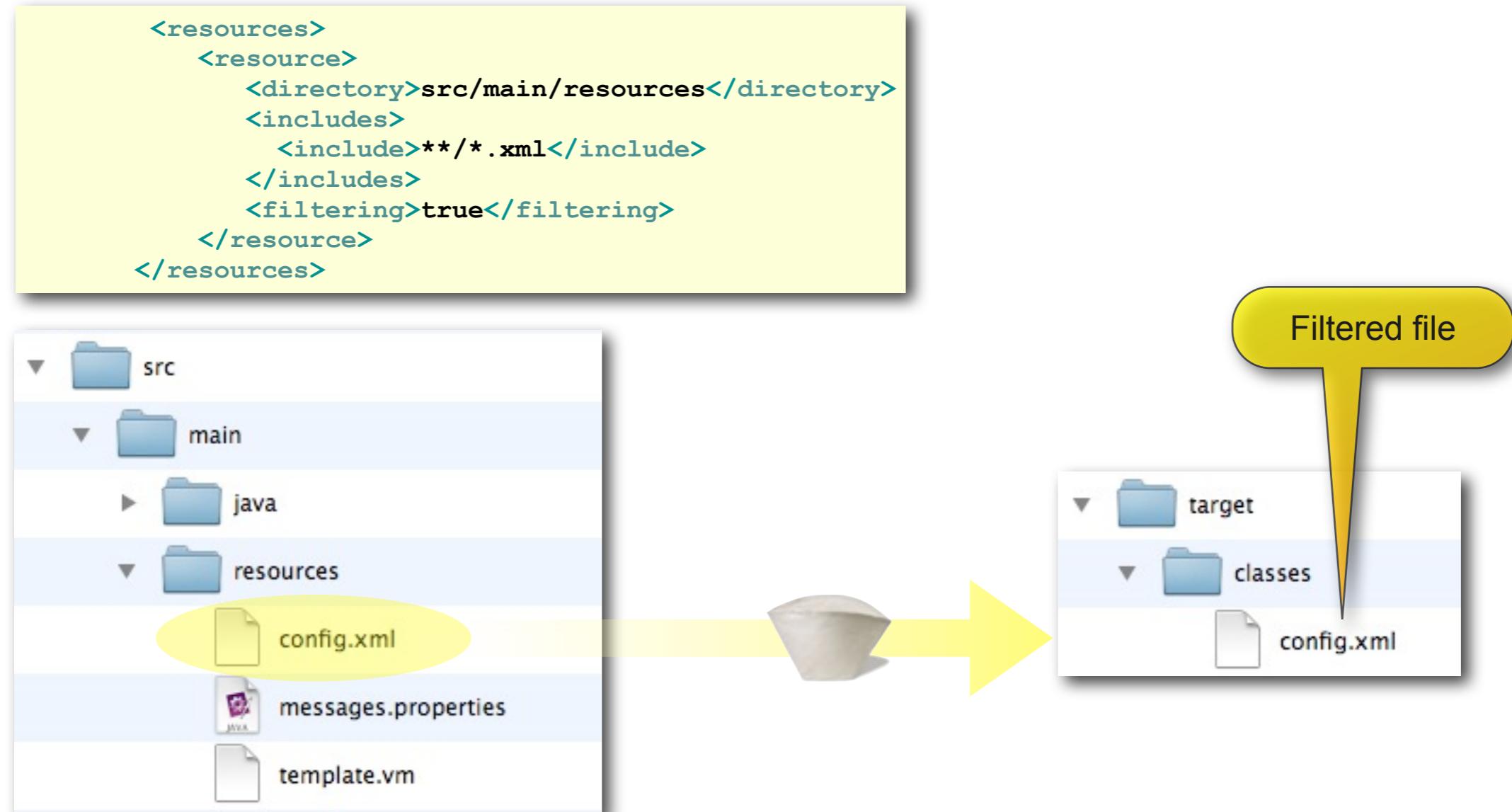
```

Only XML files will be deployed

Including/excluding Resources



- ▶ Selective filtering
- ▶ Attention: here, only the included files will be deployed



Including/excluding Resources

- ▶ What files do you want to filter?
- ▶ Selectively filtering certain files

```
<project>
  ...
  <build>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
        <filtering>true</filtering>
      </resource>
      <resource>
        <directory>src/main/resources</directory>
        <excludes>
          <exclude>**/*.xml</exclude>
        </excludes>
        <filtering>false</filtering>
      </resource>
    </resources>
  ...

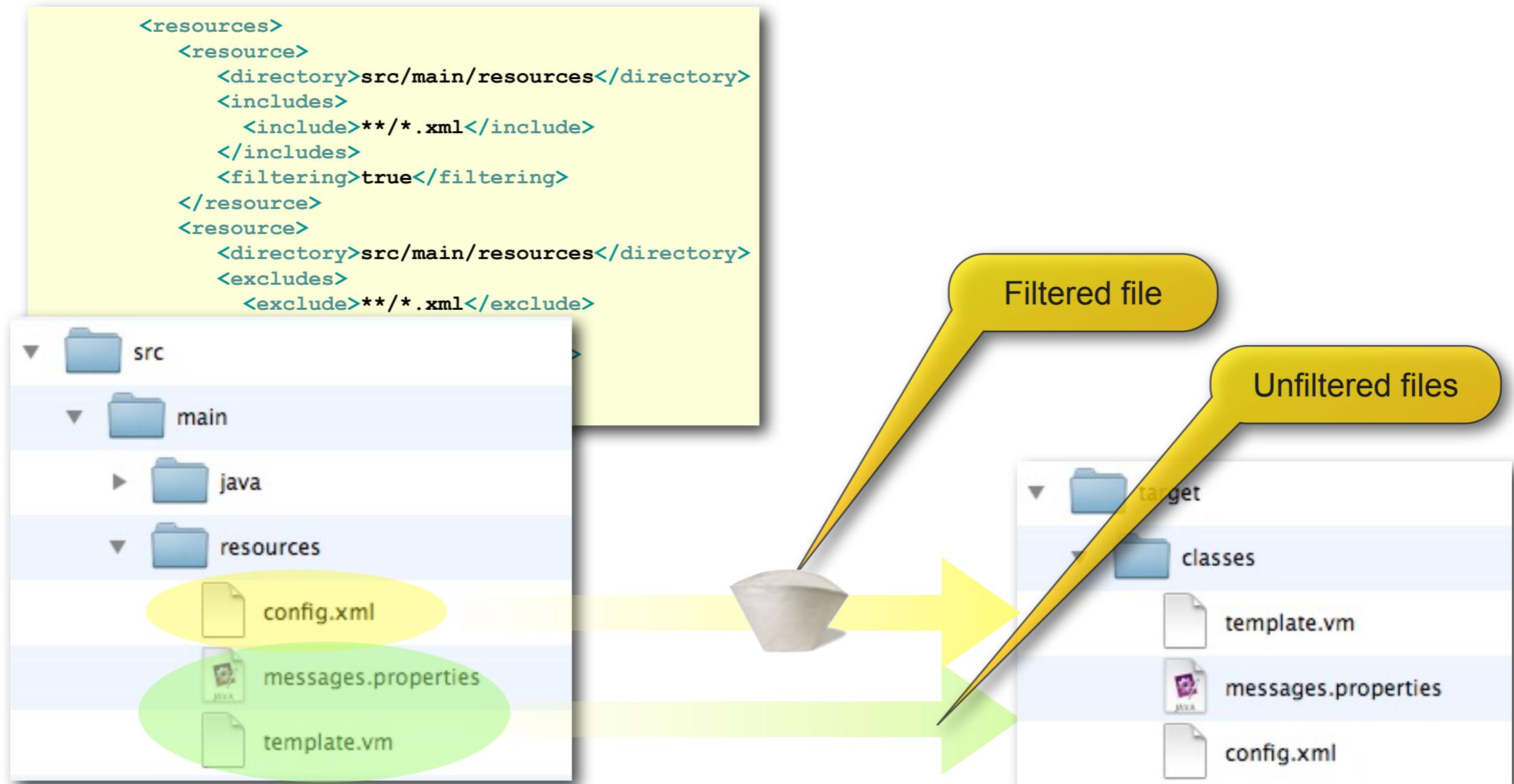
```

Only XML files will be filtered

Turn filtering off for all the non-XML files

Including/excluding Resources

- ▶ What files do you want to filter?
- ▶ Selectively filtering certain files





Profiles

Working with build profiles



- ▶ What are build profiles?
 - ▶ Customize the build process for different environments or circumstances
 - ▶ Override default behavior defined in your POM
 - ▶ Define specific property values
 - ▶ Define specific behavior (by configuring plugins)
- ▶ Where do you put them?
 - ▶ In your POM...
 - ▶ ...or in a settings.xml file

Working with build profiles



- ▶ What can you override
 - ▶ Almost everything!
 - ▶ Build configuration
 - ▶ FinalName
 - ▶ Resources and Test Resources
 - ▶ Plugins
 - ▶ Reporting
 - ▶ Modules
 - ▶ Dependencies and DependencyManagement sections
 - ▶ Repositories and Plugin Repositories
 - ▶ Properties

Working with build profiles



▶ Some simple profiles

```
<profiles>
  <!-- Development environment -->
  <profile>
    <id>development</id>
    <properties>
      <!-- The development platform -->
      <log4j.level>DEBUG</log4j.level>
    </properties>
  </profile>
</profiles>
```

Every profile needs a unique identifier

This property will only be defined for this profile

```
<profile>
  <id>production</id>
  <properties>
    <log4j.level>WARNING</log4j.level>
  </properties>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <configuration>
            <debug>false</debug>
            <optimize>true</optimize>
          </configuration>
        </plugin>
      </plugins>
    <pluginManagement>
    </build>
  </profile>
```

This profile uses a different property value

Customize the compiler plugin

Working with build profiles



- ▶ Activating a profile
 - ▶ Profiles need to be activated to take effect
 - ▶ You can activate a profile in several ways:
 - ▶ By defining a default profile
 - ▶ From the command line using the -P option and the profile id
 - ▶ By specifying conditions (environment variables, JDK versions, OS...) in which this profile should be activated
 - ▶ From the settings
 - ▶ Several profiles can be activated at once.

Working with build profiles



- ▶ Setting up a default profile
- ▶ A default profile will be activated if no other profiles (in the same pom) are activated
- ▶ Use a default profile to define sensible default values for any properties used elsewhere

```
<profile>
    <id>development</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
        <!-- The development platform -->
        <log4j.debug>DEBUG</log4j.debug>
    </properties>
</profile>
```

Use the 'development' profile if no others are activated

Working with build profiles



- ▶ Defining default profiles in the `settings.xml` file
- ▶ You can also define active profiles using the `<activeProfiles>` entry:

```
<settings>
  <activeProfiles>
    <activeProfile>development</activeProfile>
    <activeProfile>windows</activeProfile>
  </activeProfiles>
</settings>
```

Use the 'development' and
'windows' profiles

Working with build profiles



- ▶ Activating a profile from the command line
- ▶ Just use the -P option

```
$ mvn deploy -P production
```

Use the 'production' profile

```
$ mvn deploy -P production,unix
```

You can activate several profiles at once

- ▶ Deactivating a default profile
- ▶ Use '!'

```
$ mvn deploy -P !development
```

Deactivate the 'development' profile

Working with build profiles



- ▶ Activating a profile using properties or environment variables
- ▶ Activate a profile if a property or environment variable is set to a certain value
- ▶ Properties can be provided on the command line using -D
- ▶ Environment variables can be accessed using the “env.” prefix

```
<profile>
  <id>development</id>
  <activation>
    <property>
      <name>env.PLATFORM</name>
      <value>dev</value>
    </property>
  </activation>
  ...
</profile>
```

Activated if an environment variable called PLATFORM is set to “dev”

```
$ export PLATFORM=dev
$ mvn deploy
```

```
<profile>
  <id>development</id>
  <activation>
    <property>
      <name>platform</name>
      <value>dev</value>
    </property>
  </activation>
  ...
</profile>
```

Activated if a property called ‘platform’ is set to “dev”

```
$ mvn deploy -Dplatform=dev
```

Working with build profiles



- ▶ Activating a profile using environment conditions
- ▶ Define profiles that will be activated for certain operating systems or Java versions

```
<profile>
  <id>windows-test</id>
  <activation>
    <os>
      <name>Windows XP</name>
      <family>Windows</family>
      <arch>x86</arch>
      <version>5.1.2600</version>
    </os>
  </activation>
  ...
</profile>
```

Only activate on Windows environments

Working with build profiles



▶ Activating a profile using JDK

```
<profile>
  <id>jdk-14-test</id>
  <activation>
    <jdk>1.4</jdk>
  </activation>
  ...
</profile>
```

```
<profile>
  <id>jdk-14-test</id>
  <activation>
    <jdk>[1.3,1.6)</jdk>
  </activation>
  ...
</profile>
```

Brackets in version ranges
are end inclusive

Parenthesis in version
ranges are end exclusive

This range includes 1.3,
1.4, 1.5 but not 1.6

Working with build profiles



- ▶ Activating a profile using a file's existence

```
<profile>
  <id>jdk-14-test</id>
  <activation>
    <file>
      <missing>target/generated-sources/something</missing>
    </file>
  </activation>
  ...
</profile>
```

Profile Inheritance



- ▶ Profiles themselves are not inherited
 - ▶ The results of a profile are inherited -- properties set, dependencies added, etc
- ▶ The only net impact of this is that profiles defined in a parent may not be activated by a profile set in the child.

labs

- ▶ Lab 11 - Working with build profiles



Repositories

Maven Repositories



- ▶ Maven downloads artifacts from a remote repository
 - ▶ Keeps the size of the Maven core small and modular
- ▶ The default remote repository is known as “Central”
 - ▶ <http://repo1.maven.org/maven2/>
 - ▶ Defined by the superpom
- ▶ Maven Repositories are defined by Structure
 - ▶ A collection of artifacts organized to be easily understood by Maven
 - ▶ Directory hierarchy defined by Maven coordinates
 - ▶ <groupId>/<artifactId>/<version>/<artifactId>-<version>. <packaging>

Maven Repositories



- ▶ Artifacts are downloaded from remote repos to the local repo
 - ▶ The local repo serves as a cache of artifacts
- ▶ Local repo is located in your home dir
 - ▶ **Linux/Unix/Mac** - `~/.m2/repository/`
 - ▶ **Windows XP** - `C:\Documents and Settings\<USERNAME>\.m2\repository`
 - ▶ **Windows Vista** - `C:\Users\<USERNAME>\.m2\repository`
- ▶ Artifacts built locally can be installed into the local repo
 - ▶ `mvn install`
- ▶ Artifacts can also be copied to a remote repository
 - ▶ `mvn deploy`

Maven Repositories



- ▶ Some examples:
 - ▶ JUnit - junit:junit:4.4
 - ▶ <http://repo1.maven.org/maven2/junit/junit/4.4/junit-4.4.jar>
 - ▶ Castor - org.codehaus.castor:castor:1.2
 - ▶ <http://repo1.maven.org/maven2/org/codehaus/castor/castor/1.2/castor-1.2.jar>
 - ▶ Maven - org.apache.maven:maven-core:2.2.1
 - ▶ <http://repo1.maven.org/maven2/org/apache/maven/maven-core/2.2.1/maven-core-2.2.1.jar>

Maven Repositories



- ▶ Installation into the local repo allows other projects to access the artifacts
- ▶ Example output during a Maven run:

```
$ mvn install  
...  
[INFO] [install:install]  
[INFO] Installing /Users/someuser/Desktop/eclipse-3.4/  
workspace/simple/target/simple-0.0.1-SNAPSHOT.jar to /Users/  
someuser/.m2/repository/com/sonatype/maven/simple/0.0.1-  
SNAPSHOT/simple-0.0.1-SNAPSHOT.jar  
...
```

Maven Repositories



- ▶ All non-SNAPSHOT artifacts are cached (found or not found)
- ▶ Force an update from remote repos via the -U flag
 - ▶ For example, ran a build while off the ‘net on an airplane
 - ▶ Useful for when a “failure to find an artifact” is cached
 - ▶ Once back on the internet, run your goal with -U to retrieve the plugin or dependency

```
$ mvn package -U
```

```
...
```

```
<pluginRepositories>
  <pluginRepository>
    <id>apache-plugin-snapshots</id>
    <name>Apache Maven Plugins Snapshot Repository</name>
    <url>http://people.apache.org/maven-snapshot-repository</url>
    <snapshots>
      <enabled>true</enabled>
      <updatePolicy>always</updatePolicy>
      <checksumPolicy>fail</checksumPolicy>
    </snapshots>
    <releases>
      <enabled>false</enabled>
    </releases>
  </pluginRepository>
</pluginRepositories>
```

or...

```
<repositories>
  <repository>
    ...
  </repository>
</repositories>
```

updatePolicy can be "always", "daily" (default), "interval:XXX" (in minutes) or "never".

checksumPolicy for failure of artifact checksum verification can be "fail" or "warn".

Maven Intermediate Concepts

Consistent Development Environments



maven ant Tasks



Overview

Introduction

Download
Installation
Usage
Release Notes
FAQ

Examples

dependencies
install, deploy
pom
writepom
mvn

Reference

dependencies
install, deploy
install-provider
localRepository
remoteRepository
pom
writepom
versionMapper

Project Documentation

Project Information

About

Continuous
Integration
Dependencies
Issue Tracking
Mailing Lists
Plugin Management
Project License
Project Plugins
Project Summary
Project Team
Source Repository

Maven Ant Tasks

The Maven Ant Tasks allow several of Maven's artifact handling features to be used from within an Ant build. These include:

- *Dependency management* - including transitive dependencies, scope recognition and SNAPSHOT handling
- *Artifact deployment* - deployment to a Maven repository (file integrated, other with extensions)
- *POM processing* - for reading and writing a Maven 2 pom.xml file

The Ant tasks can be downloaded from the [download page](#).

System Requirements

JDK JDK version 1.5 or above is required (This is due to dependencies on internal Maven libraries.)

Ant Ant version 1.6.x or 1.7.x is required

Usage

General instructions for installing and using the Maven Ant Tasks can be found on the [installation page](#) and the [usage page](#) respectively. Some more specific use cases are described in the examples given below. Last but not least, users occasionally contribute additional examples, tips or errata to the [project's wiki page](#) .

Examples

Several common usage examples are provided for the following tasks:

- Dependencies task
- Install and Deploy tasks
- Pom task
- WritePom task
- Mvn task

What are Maven Ant Tasks?



- ▶ Ant plugin (tasks) that interoperate with Maven repositories
- ▶ Dependency management
 - ▶ Transitive dependencies, scope recognition and SNAPSHOT handling
- ▶ Artifact deployment
 - ▶ Deployment to a Maven repository (file integrated, other with extensions)
- ▶ POM processing
 - ▶ Reading and writing a Maven 2 pom.xml file

Maven Ant Task References



- ▶ Project Homepage
 - ▶ <http://maven.apache.org/ant-tasks/index.html>
- ▶ Managing dependencies
 - ▶ <http://maven.apache.org/ant-tasks/examples/dependencies.html>
 - ▶ Adds dependencies to pathId
- ▶ Install an Ant artifact to a Maven Repository
 - ▶ <http://maven.apache.org/ant-tasks/examples/install-deploy.html>
- ▶ POM Management
 - ▶ <http://maven.apache.org/ant-tasks/examples/pom.html>

Consistency Approaches

Consistent Development Environments



- ▶ Make it easy to get Maven working.
- ▶ Make it easy to do the right thing with the build.
- ▶ Set a default goal.
- ▶ Create safety nets for “late night” mistakes.
 - ▶ JDK
 - ▶ Plugins
 - ▶ Maven version

Consistent
maven Version

Consistent environments

- ▶ Create a “standard” Maven install
- ▶ Use the global settings.xml file to define enterprise repositories and mirrors or a Nexus instance.
- ▶ Store a copy of the standard install in your SCM
- ▶ Provide instructions on the project/company wiki



Consistent environments



- ▶ Require team-wide consistent version of Maven
- ▶ Significant improvements in behavior
 - ▶ Less plugin, text encoding consistency in **<= 2.0.8**
 - ▶ Core plugin definitions in **>= 2.0.9**
 - ▶ Multithreaded downloads in **>= 2.1.0**
 - ▶ Command and default plugin execution IDs **>= 2.2.0**
 - ▶ <http://maven.apache.org/guides/mini/guide-default-execution-ids.html>
 - ▶ Requires JDK 1.5 **>= 2.2.0**
 - ▶ Pluggable providers per wagon protocol **>= 2.2.1**
- ▶ Full Release Notes

minimum Maven Version



```
pom.xml
1 <?xml version="1.0"?>
2 <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/maven-v4_0_0.xsd">
3     <modelVersion>4.0.0</modelVersion>
4     <groupId>com.ambientideas</groupId>
5     <artifactId>sample-svn-scm</artifactId>
6     <packaging>jar</packaging>
7     <version>1.9-SNAPSHOT</version>
8     <name>Sample Project - Subversion SCM</name>
9     <url>http://github.com/matthewmcullough/maven-training/tree/master</url>
10
11     <prerequisites>
12       <maven>2.0.9</maven>
13     </prerequisites>
14
15     <scm>
16
17       <connection>scm:svn:http://ambientideas.unfuddle.com/svn/ambientideas_sample11
·           -svnscm</connection>
18
19       <url>http://ambientideas.unfuddle.com/svn/ambientideas_sample11-svnscm</url>
20     </scm>
```

The code block shows a Maven POM XML file. A red oval highlights the `<prerequisites>` section, which contains a single entry for Maven 2.0.9. The entire file is color-coded for syntax highlighting, with tags in blue and text in various shades of green, yellow, and white.

Rule Enforcement

Standard rules



Enforcer Rules – Standard Rules

http://maven.apache.org/enforcer/enforcer-rules/index.html

Google

Enforcer Rules – Standard Rules

 Apache Maven Project <http://maven.apache.org>

Apache > Maven > Enforcer > Enforcer Rules

Last Published: 2009-02-25 | Version: 1.0-beta-1

Overview

Introduction

Rule API

Maven Enforcer Plugin

Custom Rules

Writing a custom rule

Project Documentation

▶ Project Information

▶ Project Reports

Maven Projects

Ant Tasks

Doxia

JXR

Maven 1.x

Maven 2

Mercury

Plugins

SCM

Shared Components

Surefire

Standard Rules

The following standard rules ship along with the enforcer plugin:

- [alwaysPass](#) - Always passes... used to test plugin configuration.
- [alwaysFail](#) - Always fail... used to test plugin configuration.
- [bannedDependencies](#) - enforces that excluded dependencies aren't included.
- [evaluateBeanshell](#) - evaluates a beanshell script.
- [requireReleaseDeps](#) - enforces that no snapshots are included as dependencies.
- [requireReleaseVersion](#) - enforces that the artifact is not a snapshot.
- [requireMavenVersion](#) - enforces the Maven version.
- [requireJavaVersion](#) - enforces the JDK version.
- [requireOS](#) - enforces the OS / CPU Architecture.
- [requirePluginVersions](#) - enforces that all plugins have a specified version.
- [requireProperty](#) - enforces the existence and values of properties.
- [requireFilesNotExist](#) - enforces that the list of files do not exist.
- [requireFilesExist](#) - enforces that the list of files do exist.
- [requireFileSize](#) - enforces that the list of files exist and are within a certain size range.

You may also create and inject your own custom rules by following the [maven-enforcer-rule-api](#) instructions.

Rule enforcement



- ▶ Ensure that your project uses a specific Maven version
- ▶ Similar to the core Maven version enforcement, but profile-selectable

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0-beta-1</version>
  <executions>
    <execution>
      <id>enforce-versions</id>
      <goals>
        <goal>enforce</goal>
      </goals>
      <configuration>
        <rules>
          <requireMavenVersion>
            <version>2.0.10</version>
          </requireMavenVersion>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Requires Maven 2.0.10 or higher

Consistent
JPK

Consistent version of Java



- ▶ Ensure that all developers are using the same JDK
- ▶ Motivations
 - ▶ Bytecode produced is not identical between JDKs, even with the same target platform specified
 - ▶ XML parsers that ship with the JDK are different
 - ▶ Have you tried to use JAXB with Java 6?
 - ▶ Have you compiled something with Java 5 and distributed it to someone running 1.4?
 - ▶ Core Maven and its plugins are affected by the current JDK

Rule enforcement



- ▶ Ensure that your project uses a specific Maven version and JDK

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0-beta-1</version>
  <executions>
    <execution>
      <id>enforce-versions</id>
      <goals>
        <goal>enforce</goal>
      </goals>
      <configuration>
        <rules>
          <requireJavaVersion>
            <version>[1.5.0,1.6.0)</version>
          </requireJavaVersion>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Java 1.5.0 or greater, but less than Java 1.6.0

**Consistent
Plugin Versions**

Specify plugin versions



- ▶ In Maven 2.0.x, you can use a plugin with no version number
- ▶ In Maven 3.0 missing version specifications outputs a warning to the console
- ▶ Undeclared versions are never a good idea
 - ▶ Changes can break your build
 - ▶ Builds can become inconsistent among developers
 - ▶ Builds may not be repeatable over time
- ▶ Lock down all your plugin versions



Reminder: From Maven 2.0.9 onward, default plugins such as `compiler` and `resource` are set to specific versions in the super POM.

Super POM



[Apache-SVN] Contents of /maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project/pom-4.0.0.xml

1P + http://svn.apache.org/viewvc/maven/components/tags/maven-2.0.9/maven-project/src/main/resource Google

[Apache-SVN] Contents of /maven...

/[Apache-SVN]/maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project/pom-4.0.0.xml

Contents of /maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project/pom-4.0.0.xml

 viewvc

[Parent Directory](#) | [Revision Log](#)

Revision [645582](#) - ([show annotations](#)) ([download](#)) ([as text](#))
Mon Apr 7 16:02:54 UTC (22 months, 1 week ago) by brianf
File MIME type: text/xml
File size: 6401 byte(s)

```
[maven-release-plugin] copy for tag maven-2.0.9

1 <!--
2 Licensed to the Apache Software Foundation (ASF) under one
3 or more contributor license agreements. See the NOTICE file
4 distributed with this work for additional information
5 regarding copyright ownership. The ASF licenses this file
6 to you under the Apache License, Version 2.0 (the
7 "License"); you may not use this file except in compliance
8 with the License. You may obtain a copy of the License at
9
10  http://www.apache.org/licenses/LICENSE-2.0
```

Super POM - Plugin Versions



[Apache-SVN] Contents of /maven/components/tags/maven-2.0.9/maven-project/src/main/resources/org/apache/maven/project/pom-4.0.0.xml
http://svn.apache.org/viewvc/maven/components/tags/maven-2.0.9/maven-project/src/main/resource Google

[Apache-SVN] Contents of /maven...

```
69 </testResources>
70 <pluginManagement>
71   <plugins>
72     <plugin>
73       <artifactId>maven-antrun-plugin</artifactId>
74       <version>1.1</version>
75     </plugin>
76     <plugin>
77       <artifactId>maven-assembly-plugin</artifactId>
78       <version>2.2-beta-2</version>
79     </plugin>
80     <plugin>
81       <artifactId>maven-clean-plugin</artifactId>
82       <version>2.2</version>
83     </plugin>
84     <plugin>
85       <artifactId>maven-compiler-plugin</artifactId>
86       <version>2.0.2</version>
87     </plugin>
88     <plugin>
89       <artifactId>maven-dependency-plugin</artifactId>
90       <version>2.0</version>
91     </plugin>
92     <plugin>
93       <artifactId>maven-deploy-plugin</artifactId>
94       <version>2.3</version>
95     </plugin>
96     <plugin>
97       <artifactId>maven-ear-plugin</artifactId>
98       <version>2.3.1</version>
```

Specify plugin versions



- ▶ For predictable builds, avoid SNAPSHOT plugins
- ▶ Release plugin will check no SNAPSHOT dependencies are used
- ▶ What about plugins?
 - ▶ Enforcer plugin can verify no SNAPSHOT versions are used
 - ▶ <http://maven.apache.org/enforcer/enforcer-rules/requirePluginVersions.html>

Enforce plugin versions



- ▶ Ensure that plugin version numbers are declared
- ▶ Forbid SNAPSHOT, LATEST and RELEASE versions

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0-beta-1</version>
  <executions>
    <execution>
      <id>enforce-versions</id>
      <goals>
        <goal>enforce</goal>
      </goals>
      <configuration>
        <rules>
          <requirePluginVersions/>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

All plugins must have stable
version numbers

Enforce plugin versions



Enforcer Rules – Require Plugin Versions

http://maven.apache.org/enforcer/enforcer-rules/requirePluginVersions.html

maven enforcer plugin

Enforcer Rules – Require Plugin ...

Apache Maven Project
http:// maven.apache.org

Apache > Maven > Enforcer > Enforcer Rules

Last Published: 2009-02-25 | Version: 1.0-beta-1

Require Plugin Versions

This rule enforces that all plugins have a version defined, either in the plugin or pluginManagement section of the pom or a parent pom.

The following parameters are supported by this rule:

- message - an optional message to the user if the rule fails.
- banLatest - disallow any use of "LATEST" as a version for any plugin. Default = true.
- banRelease - disallow any use of "RELEASE" as a version for any plugin. Default = true.
- banSnapshots - disallow any use of SNAPSHOT plugins. Default = true.
- banTimestamps - disallow any use of snapshot plugins with timestamp version (only enabled when banSnapshots is true). Default = true.
- phases - The comma separated list of phases that should be used to find lifecycle plugin bindings. The default value is "clean,deploy,site".
- additionalPlugins - A list of additional plugins to enforce have versions. These are plugins that may not be in the poms but are used anyway, like help, eclipse etc. The plugins should be specified in the form: group:artifactId.
- unCheckedPluginsList - A comma separated list of plugins to skip version checking. Ie allow no version, or snapshots, etc. The plugins should be specified in the form: group:artifactId.

Questions



- ▶ What could you enforce?
- ▶ Are your builds platform dependent?
- ▶ Do you have a top level “corporate” pom?
- ▶ Great place for team-wide rules
- ▶ Ever been frustrated by moving SNAPSHOTs?

Maven Intermediate Concepts

Part 6

Deeper Dependency Management



Optional Dependencies

Optional Dependencies

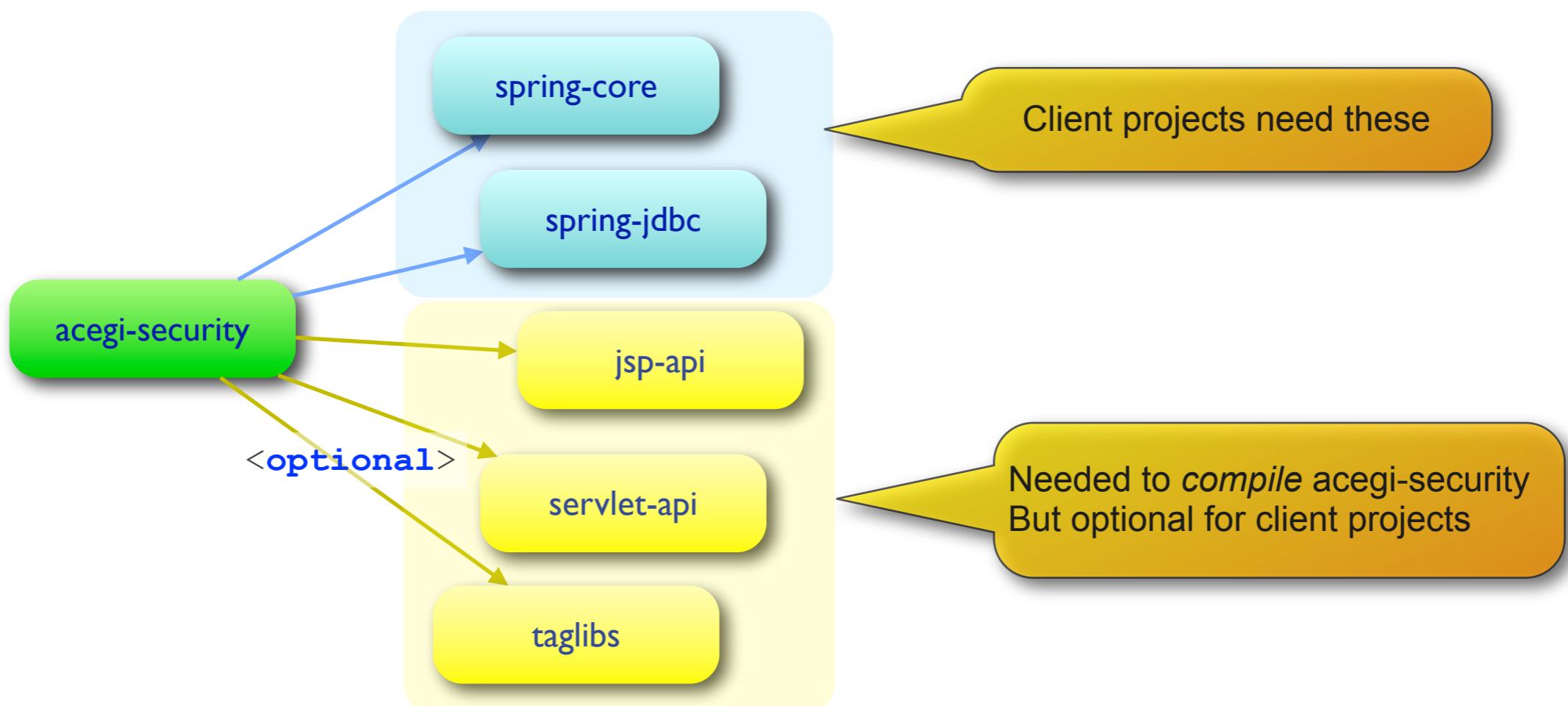


- ▶ Dependencies can be marked as optional
 - ▶ Provide choices to library users (e.g. supported databases, plugins,...)
 - ▶ Libraries that are only required for certain (optional) features
 - ▶ You need all of them to compile the project
 - ▶ But client projects only need to declare the ones they use
- ▶ Why use optional dependencies?
 - ▶ Avoid unnecessary API bloat
 - ▶ Clearly define what dependencies are required to use a project

Optional Dependencies



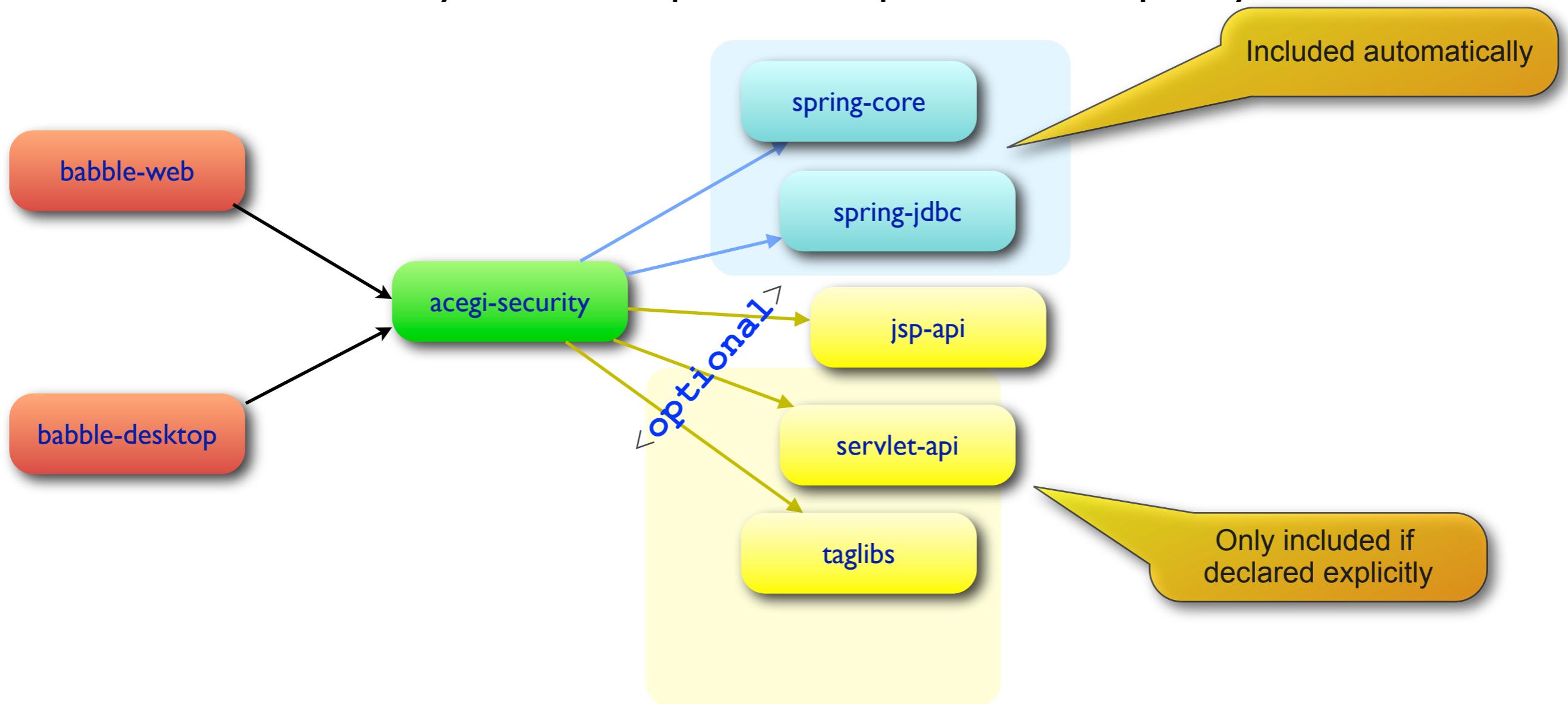
- ▶ Dependencies can be marked as optional
- ▶ Provide choices to library users (e.g. supported databases, plugins,...)
- ▶ You need all of them to compile the project
- ▶ Client projects only need to declare the ones they use



Optional Dependencies



- ▶ Dependencies can be marked as optional
- ▶ By default, client projects only get the *required* libraries
- ▶ Must declare any desired <optional> dependencies explicitly



Optional Dependencies



acegi-security

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
  </dependency>
  ...
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.0</version>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.0.6</version>
    <optional>true</optional>
  </dependency>
  ...
  <dependencies>
  ...
</project>
```

All are needed to compile this project

But these ones are optional for client projects

We use this library in our project

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.sonatype.mavenbook</groupId>
  <artifactId>babble-web</artifactId>
  <version>1.0.0</version>
  <dependencies>
    <dependency>
      <groupId>org.acegisecurity</groupId>
      <artifactId>acegi-security</artifactId>
      <version>1.0.7</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.0</version>
    </dependency>
    ...
  </dependencies>
</project>
```

We only declare the optional libraries that this project uses

version Ranges

Version Ranges



- ▶ Infrequently used, but powerful
- ▶ Designed for widely used APIs
 - ▶ log4j to indicate compatible library ranges
 - ▶ JUnit

Version Ranges



- ▶ Number parsed if using standard pattern
 - ▶ A.B.C-buildnum or A.B.C-qualifier
 - ▶ 1.0.9 < 1.0.12
- ▶ Compared as strings if non-standard pattern
 - ▶ 1.0.0.9 > 1.0.0.12
- ▶ Range specification pages
 - ▶ <http://docs.codehaus.org/display/MAVEN/Versioning>
 - ▶ <http://jira.codehaus.org/browse/MNG-3010>
 - ▶ <http://www.sonatype.com/books/mvnref-book/reference/pom-relationships-sect-version-ranges.html>

Version Ranges



- ▶ You can specify a version range instead of an specific version
 - ▶ Exclusive quantifiers - (,)
 - ▶ Inclusive quantifiers - [,]
 - ▶ Examples
 - ▶ Include JUnit 3.8 or greater, but less than 4.0

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[3.8,4.0)</version>
  <scope>test</scope>
</dependency>
```

- ▶ No higher than JUnit 3.8.1

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[,3.8.1]</version>
  <scope>test</scope>
</dependency>
```

Version Ranges



- ▶ Using snapshots in version ranges
 - ▶ Ranges don't include SNAPSHOT versions unless explicitly defined
 - ▶ Examples
 - ▶ Use a SNAPSHOT version until the 1.0.1 release is available

```
<dependency>
  <groupId>com.acme.killerapp</groupId>
  <artifactId>core</artifactId>
  <version>[1.0.1-SNAPSHOT,1.0.1]</version>
  <scope>test</scope>
</dependency>
```

Version Ranges



- ▶ JUnit 4.0 or greater

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[4.0,)</version>
  <scope>test</scope>
</dependency>
```

- ▶ Only JUnit 3.8.1

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[3.8.1]</version>
  <scope>test</scope>
</dependency>
```

The build will fail if this version is not available

- ▶ Allow any, but prefer JUnit 3.8.1

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
</dependency>
```

If a conflict is detected, a different version might be used

Maven Intermediate Concepts

Part 7

Site Generation



Objectives



- ▶ Learn about Maven site and documentation mechanism
- ▶ Learn how to generate a site from an archetype
- ▶ Learn Best Practices such as:
 - ▶ the default site directory layout
 - ▶ the site lifecycle
 - ▶ internationalize documentation
- ▶ Learn what is the purpose of - and how to configure - the site descriptor
- ▶ Learn how to write documents in different markup languages
 - ▶ how to extend the available markup languages beyond the default
 - ▶ using and creating an alternate look-and-feel to the site



maven sites

Site Generation and Reporting



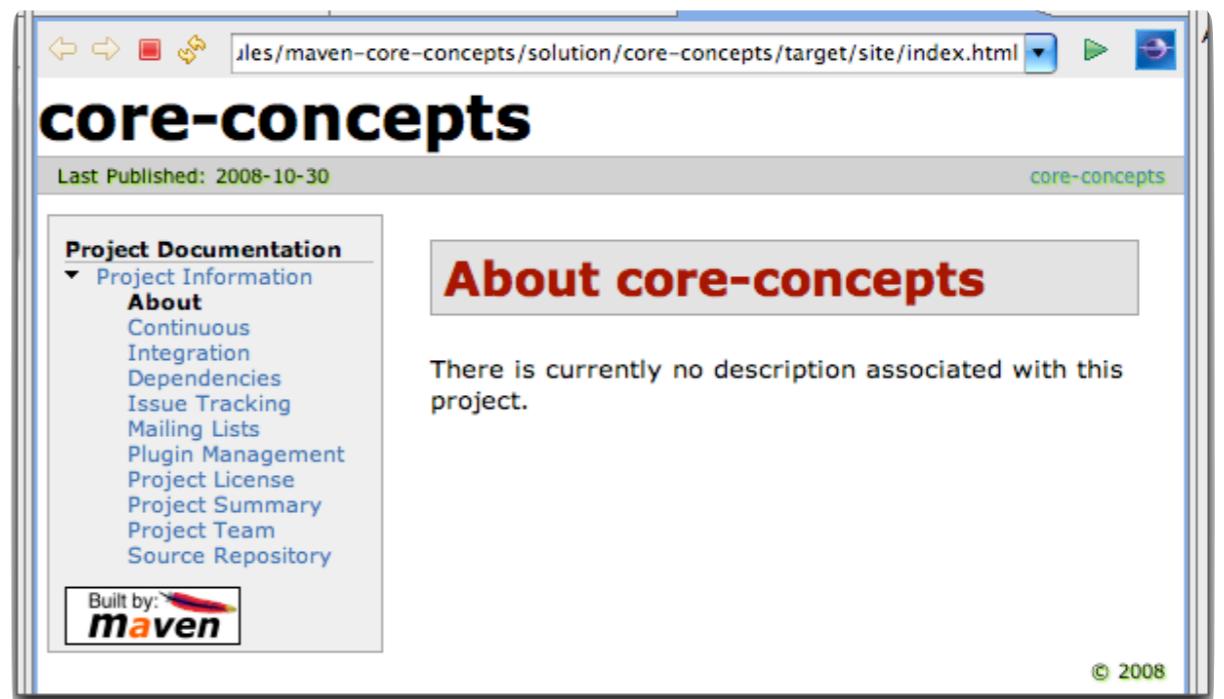
- ▶ Maven can generate documentation and reports

- ▶ Best demonstrated through a quick example

Demonstration



- ▶ Generate and view some reports
 - ▶ Right-click on the project
 - ▶ Go to Run As->Maven build...
 - ▶ Enter 'site' into the Goals field and click Run
 - ▶ Press F5 to refresh the project
 - ▶ Expand the target/site directory
 - ▶ Right-click the index.html file
 - ▶ Go to Open With->Web Browser
 - ▶ Inspect the site info



The Maven Site



- ▶ The Maven Site: a communication tool
 - ▶ General information
 - ▶ SCM, team details, issue management, CI...
 - ▶ Reports
 - ▶ Dependency graphs
 - ▶ Test results
 - ▶ Javadoc
 - ▶ Code and quality and code coverage metrics
 - ▶ ...

The Maven Site



► Generating the Maven site

- ▶ Generate the site using the **mvn site** command
- ▶ Basic configuration generates human-readable project information
 - ▶ Project description
 - ▶ Source code repository
 - ▶ Issue tracking
 - ▶ Continuous Integration
 - ▶ Dependencies
 - ▶ Team members

Project Information

This document provides an overview of the various documents and links that are part of this project's general information. All of this content is automatically generated by Maven on behalf of the project.

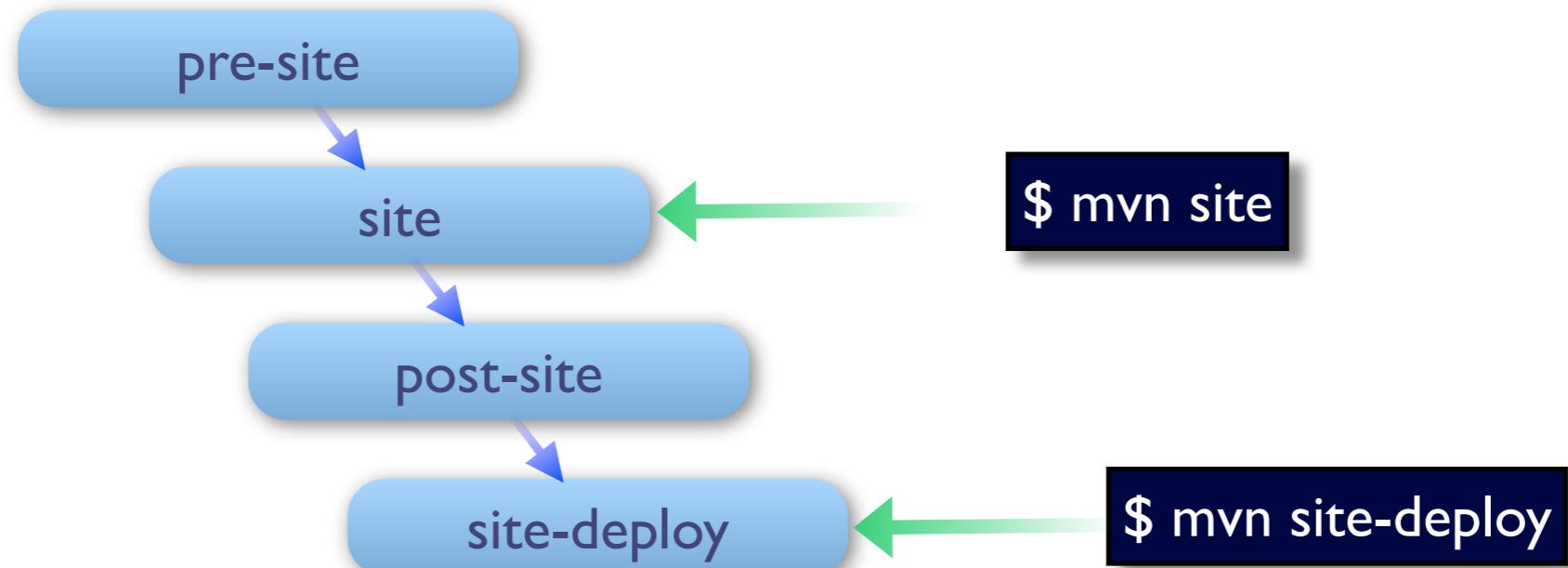
Overview

Document	Description
Continuous Integration	This is a link to the definitions of all continuous integration processes that builds and tests code on a frequent, regular basis.
Dependencies	This document lists the projects dependencies and provides information on each dependency.
Issue Tracking	This is a link to the issue management system for this project. Issues (bugs, features, change requests) can be created and queried using this link.
Mailing Lists	This document provides subscription and archive information for this project's mailing lists.
Project License	This is a link to the definitions of project licenses.
Project Summary	This document lists other related information of this project.
Project Team	This document provides information on the members of this project. These are the individuals who have contributed to the project in one form or another.
Source Repository	This is a link to the online source repository that can be viewed via a web browser.

The Site lifecycle



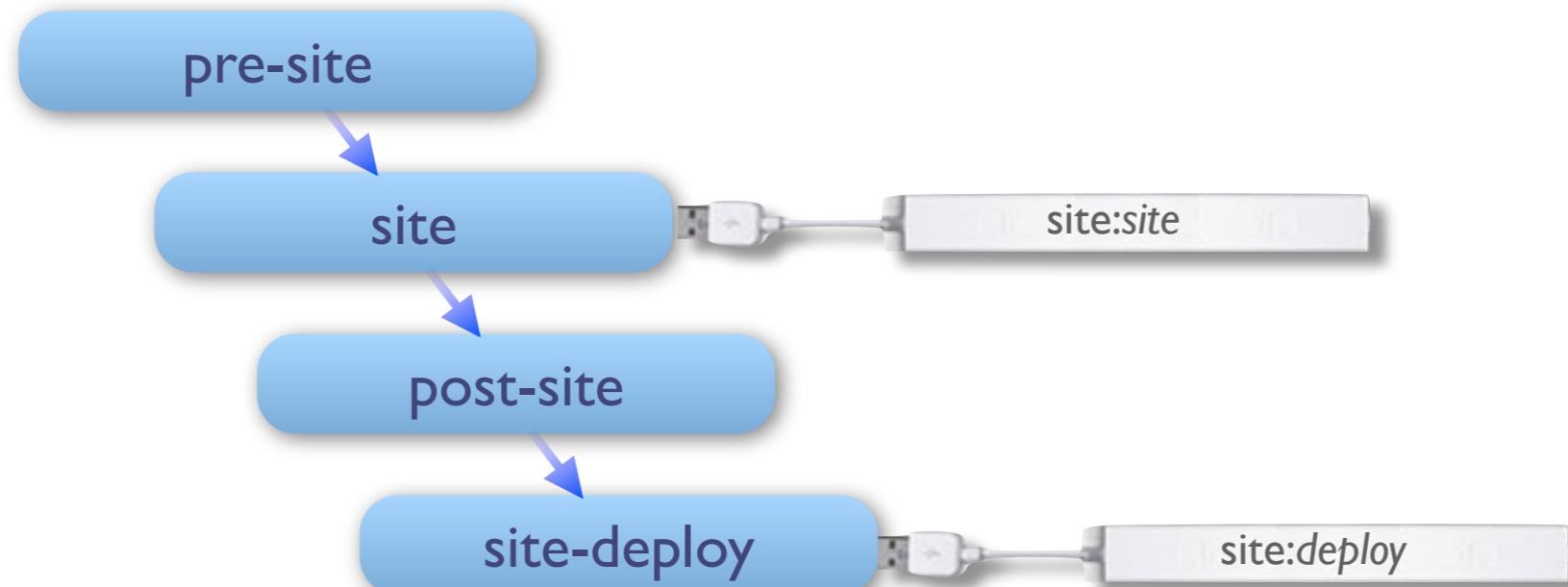
- ▶ Site generation happens outside the normal development lifecycle
- ▶ The Maven site lifecycle:



The Site lifecycle



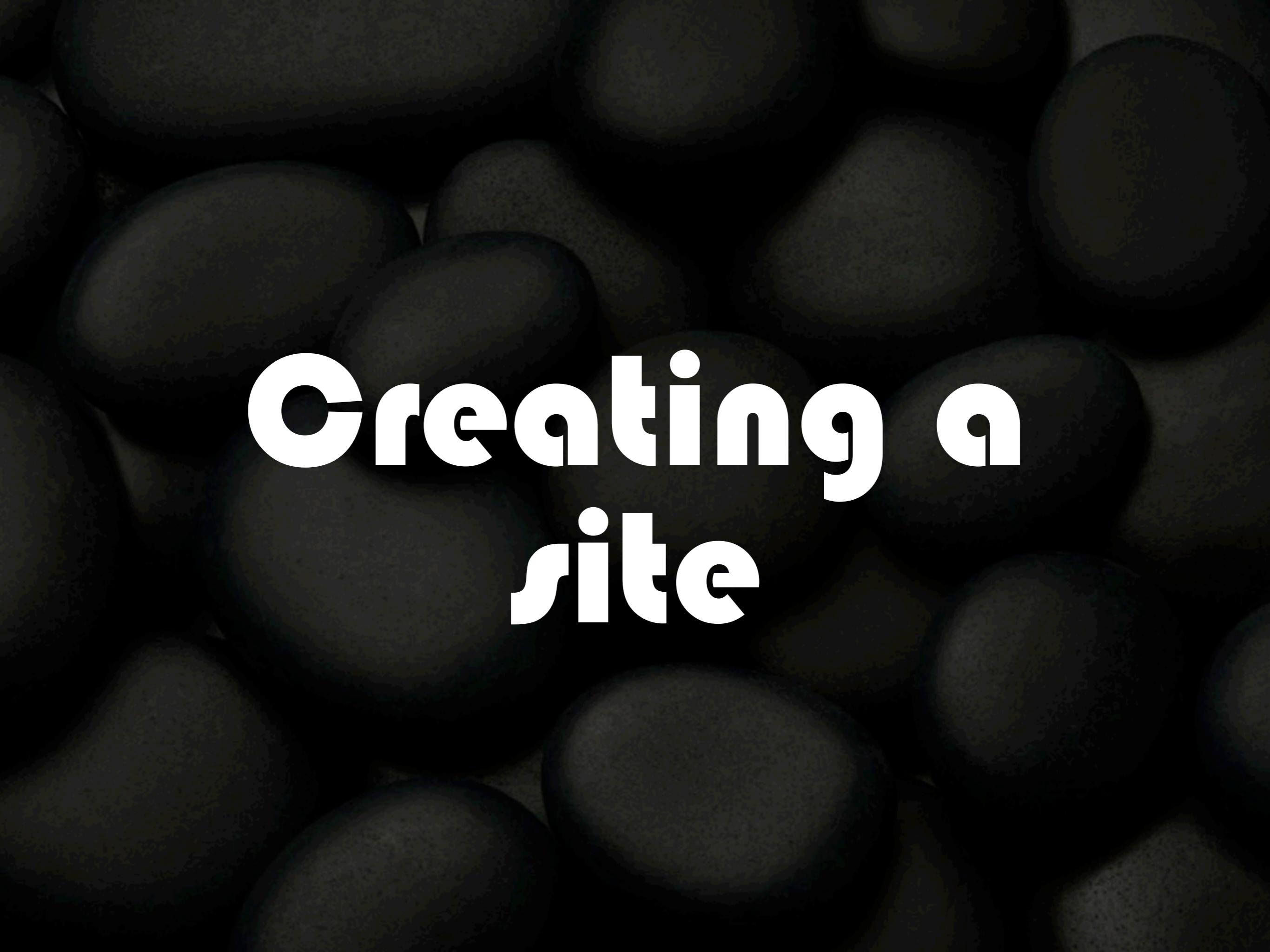
- ▶ The Maven site plugin
 - ▶ Renders documents
 - ▶ Generates reports



Default site layout



- ▶ You can add your own content to your Maven site
 - ▶ General project information
 - ▶ Architecture documents
 - ▶ Configuration notes
 - ▶ Other project documentation
- ▶ Your content can be in several formats
 - ▶ APT - *a simple, Wiki-like format*
 - ▶ XDoc - *older format used in Maven 1*
 - ▶ FML - *used for FAQ*
 - ▶ Pick whatever suits you!



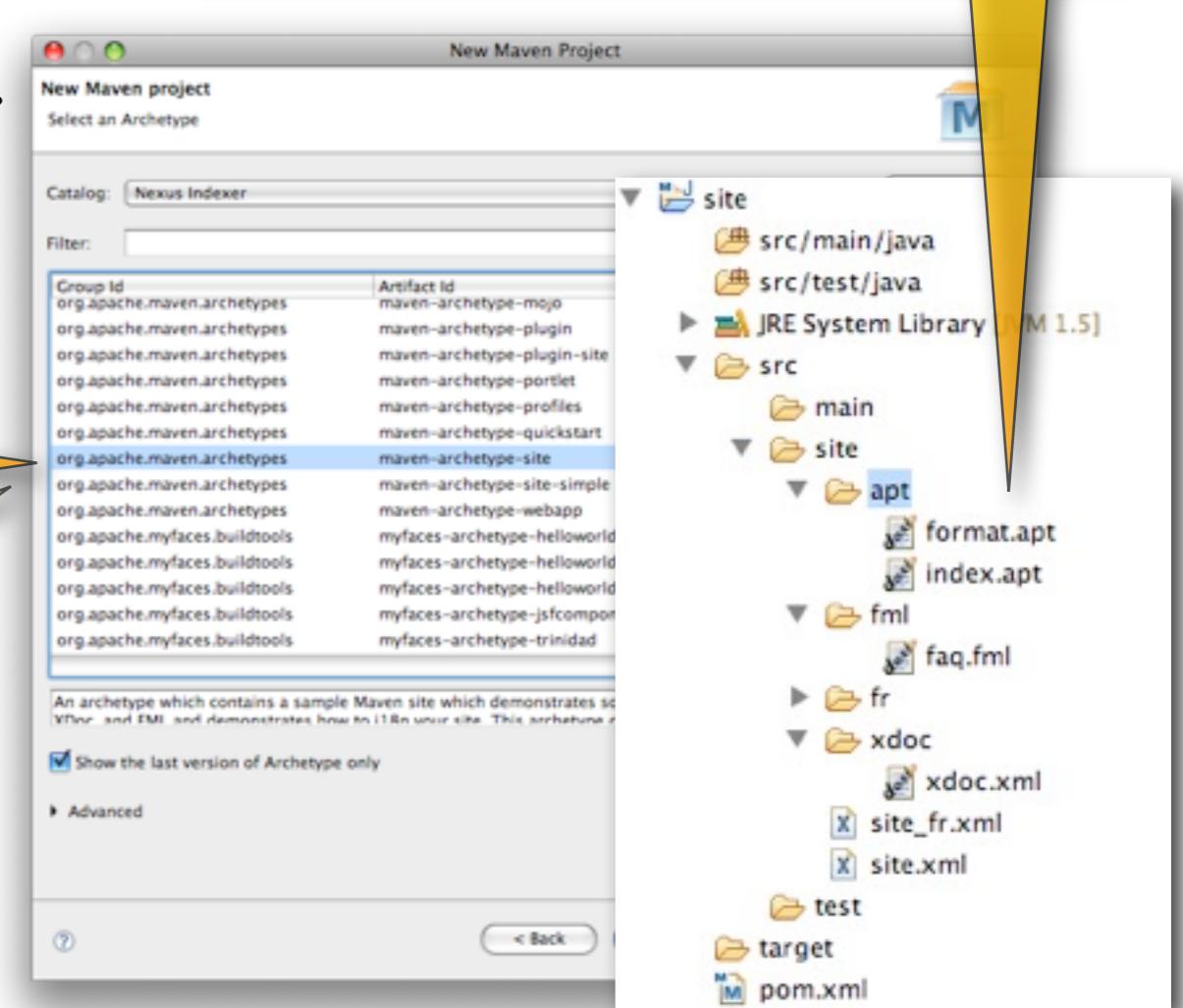
**Creating a
site**

Creating your site project



- ▶ You can define a site:
 - ▶ As part of an existing Maven project...
 - ▶ Or as a separate project...
 - ▶ The site archetype provides a project structure and sample files.

Comes with lots of formatting examples



Use the maven-archetype-site archetype to get started

You can generate this archetype on top of an existing project(!)

(but not from within an IDE)

Creating your site project



- ▶ You can also add a site to an existing project

```
$ mvn archetype:create -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-site  
-DgroupId=com.sonatype.training -DartifactId=babble  
...  
[INFO] -----  
[INFO] Building Maven Default Project  
[INFO]   task-segment: [archetype:create] (aggregator-style)  
[INFO] -----  
[INFO] ...  
[INFO] -----  
[INFO] Using following parameters for creating OldArchetype: maven-archetype-site:RELEASE  
[INFO] -----  
[INFO] Parameter: groupId, Value: com.sonatype.training  
[INFO] Parameter: packageName, Value: com.sonatype.training  
[INFO] Parameter: basedir, Value: /Users/someuser/Projects/sonatype/sonatype-training/lab-solutions  
[INFO] Parameter: package, Value: com.sonatype.training  
[INFO] Parameter: version, Value: 1.0-SNAPSHOT  
[INFO] Parameter: artifactId, Value: babble  
[INFO] ***** End of debug info from resources from generated POM *****  
[INFO] OldArchetype created in dir: /Users/someuser/Projects/sonatype/sonatype-training/lab-solutions/babble  
[INFO] -----  
[INFO] BUILD SUCCESSFUL  
[INFO] -----  
[INFO] Total time: 11 seconds  
[INFO] Finished at: Fri Mar 27 10:42:06 NZDT 2009  
[INFO] Final Memory: 8M/15M  
[INFO] -----
```

**\$ mvn archetype:create
-DarchetypeGroupId=org.apache.maven.archetypes
-DarchetypeArtifactId=maven-archetype-site
-DgroupId=com.sonatype.training -DartifactId=babble**

Adds a site template to an existing project

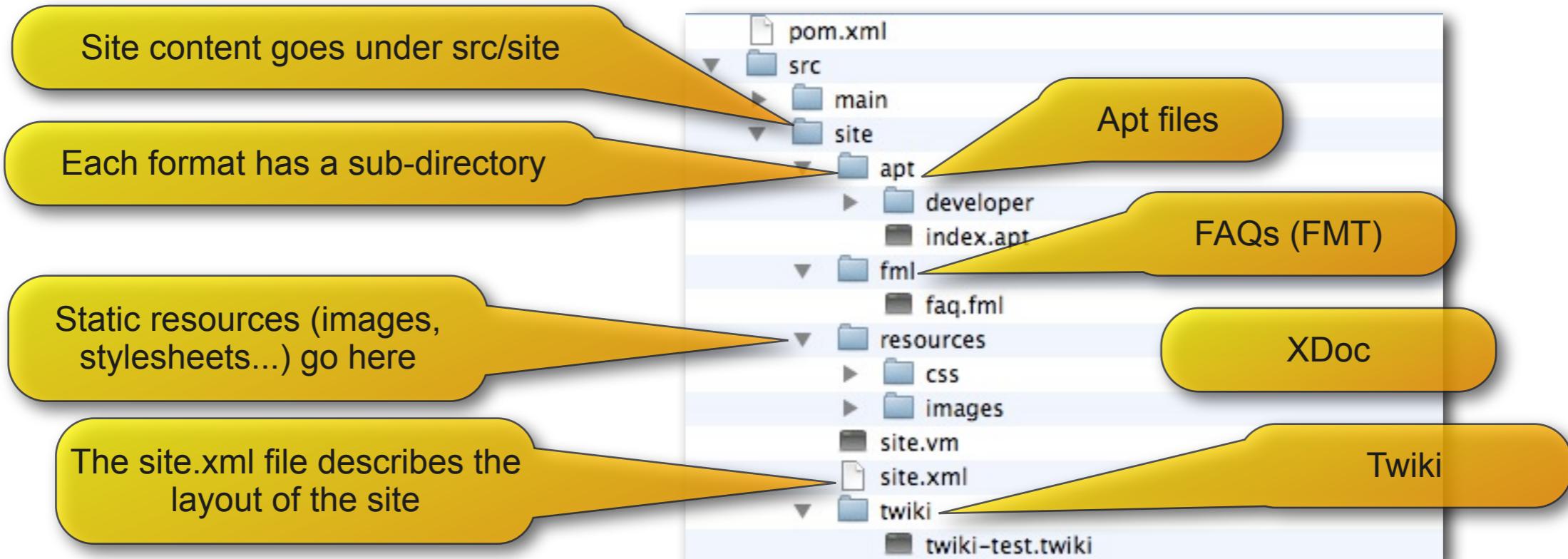
For multi-module projects, use the aggregator project

Run this command from a project's parent directory

Default site layout



- ▶ The site directory layout
 - ▶ A main layout definition file
 - ▶ And a folder for each format



Organizing your site



- ▶ Defining the site outline
- ▶ The site.xml file defines the overall structure of the site

Left and right banners

Links at the top of the screen

Menus to content

Reports menu goes here

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="Maven">
  <bannerLeft>
    <name>Maven</name>
    <src>http://maven.apache.org/images/apache-maven-project.png</src>
    <href>http://maven.apache.org/</href>
  </bannerLeft>
  <bannerRight>
    <src>http://maven.apache.org/images/maven-small.gif</src>
  </bannerRight>
  <body>
    <links>
      <item name="Apache" href="http://www.apache.org/" />
      <item name="Maven 1.0" href="http://maven.apache.org/" />
      <item name="Maven 2" href="http://maven.apache.org/maven2/" />
    </links>
    <menu name="Maven 2.0">
      <item name="APT Format" href="format.html" />
      <item name="FAQ" href="faq.html" />
      <item name="Xdoc Example" href="xdoc.html" />
    </menu>
    <menu ref="reports" />
  </body>
</project>
```

Organizing your site



- ▶ Customizing the site outline
- ▶ It is easy to personalize the site outline

Add your own logo

Customize the links

Customize the menus

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="Maven">
  <bannerLeft>
    <name>Sonatype</r
    <src>images/logo.
    <href>http://www.
  </bannerLeft>
  <body>
    <links>
      <item name="Ap
      <item name="Mav
      <item name="Mav
    </links>
    <menu name="Getti
      <item name="Ins
      <item name="FAQ
    </menu>
    <menu name="Archit
      <item name="High
      <item name="Des
    </menu>
    <menu ref="report
      </body>
</project>
```

Maven - The Site

Sonatype the maven company

Last Published: 2009-03-30

Apache | Maven 1.0 | Maven 2

Section title

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

Sub-section title

Section titles are not indented. A sub-section title begins with one asterisk (*), a sub-sub-section title begins with two asterisks (**), and so forth up to four sub-section levels.

Sub-sub-section title

This is a sub-sub-section.

Sub-sub-sub-section title

And so forth

You can configure this class in Spring as follows:

```
<bean id="taxCalculator" class="com.sonatype.training.TaxCalculatorImpl">
```

Adding content to your site



► The Xdoc format

- XML, very HTML-like format
- An older format, used in Maven 1
- Largely replaced by Apt in Maven 2

XML-based format

Structured in sections and subsections

Actual content can be any valid XHTML

Including source code

```
<?xml version="1.0"?>
<document>
  <properties>
    <title>Welcome</title>
    <author email="dev@maven.apache.org">
  </properties>
  <body>
    <section name="section 1">
      <p>Hi!</p>
      <subsection name="subsection 1">
        <p>Subsection text...</p>
      </subsection>
    </section>
    <section name="other section">
      <source>
        public class Foo {
          private int bar;

          public Foo() {
        }
      </source>
    </section>
  </body>
</document>
```

The screenshot shows a web browser window titled "Maven - Welcome" displaying the Apache Maven Project website. The page includes a sidebar with links for Maven 2.0, APT Format, FAQ, Xdoc Example, Project Documentation, Project Information, and Project Reports. The main content area features sections for "section 1" and "other section", each containing the generated XHTML content from the Xdoc XML. The "other section" also contains the source code for the Java class "Foo".

Adding content to your site



- ▶ The APT format
- ▶ Introduced in Maven 2
- ▶ Light-weight wiki-style format
- ▶ Relatively easy to learn and write
- ▶ Powerful formatting features

Adding content to your site



- ▶ The APT format - structuring your document
- ▶ APT documents are plain-text documents using a wiki-style syntax

The APT format

In the following section, boxes containing text in typewriter-like font are examples of APT source.

* Document structure

A short APT document is contained in a single text file. A longer document may be contained in an ordered list of text files. For instance, first text file contains section 1, second text file contains section 2, and so on.

A file contains a sequence of paragraphs and ``displays'' (non paragraphs such as tables) separated by open lines.

...

The APT format

Dashed underlines are optional

A document title is a line with no indentation

Sub-titles start with asterisks

Paragraph contents are indented

Paragraphs are separated by a blank line

Document structure

A short APT document is contained in a single text file. A longer document may be contained in an ordered list of text files. For instance, first text file contains section 1, second text file contains section 2, and so on.

Adding content to your site



► The APT format - structuring your document

► Asterisks for section titles

Section titles have no indentation

Sub-sections start with asterisks

Section title

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

* Sub-section title

Section titles are not indented. A sub-section title begins with one asterisk (<<<*>>>), a sub-sub-section title begins with two asterisks (<<<**>>>), and so forth up to four sub-section levels.

** Sub-sub-section title

This is a sub-sub-section.

*** Sub-sub-sub-section title

And so forth

Section title

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

Sub-section title

Section titles are not indented. A sub-section title begins with one asterisk (*), a sub-sub-section title begins with two asterisks (**), and so forth up to four sub-section levels.

Sub-sub-section title

This is a sub-sub-section.

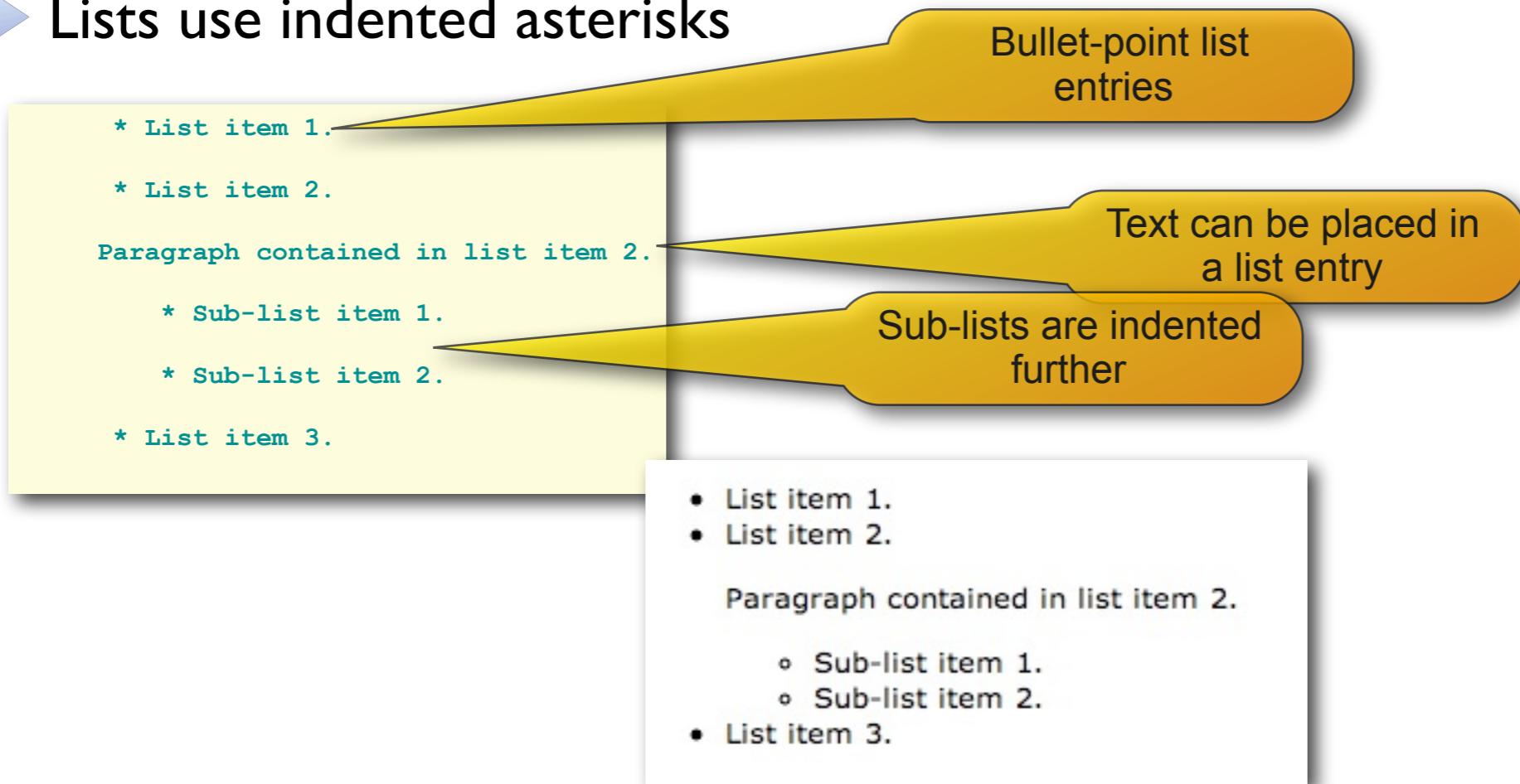
Sub-sub-sub-section title

And so forth

Adding content to your site



- ▶ The APT format - lists
- ▶ Lists use indented asterisks



Adding content to your site



- ▶ The APT format - code snippets
- ▶ You can include code samples between lines of dashes

You can configure this class in Spring as follows:

```
<bean id="taxCalculator" class="com.sonatype.training.TaxCalculatorImpl">
  ...
</bean>
```

A code snippet

Anything here will be displayed “verbatim”

You can configure this class in Spring as follows:

```
<bean id="taxCalculator" class="com.sonatype.training.TaxCalculatorImpl">
  ...
</bean>
```

Adding content to your site



- ▶ The APT format - tables
- ▶ A simple table layout format

The diagram illustrates the APT table format and its visual representation. On the left, a code snippet shows the APT table definition:

```
*-----+-----+
|| Header 1 || Header 2 |
*-----+-----+
| Cell 1   | Cell 2   |
*-----+-----+
```

The word "Caption" is written below the code. To the right, a yellow callout points to the first row of the code with the text "Headers start with '|'".

Below the code, a yellow callout points to the entire row structure with the text "Cells demarcated with dashes and bars".

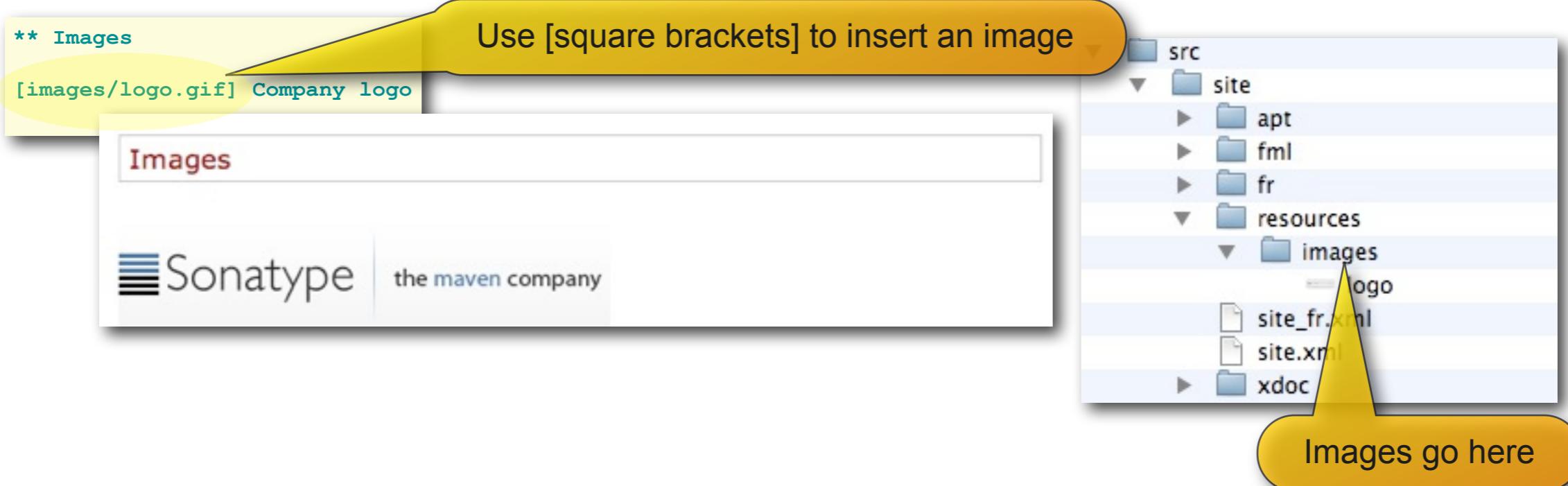
At the bottom, a visual representation of the table is shown with two columns labeled "Header 1" and "Header 2". The first row contains "Cell 1" and "Cell 2". The second row contains "Caption" above the table, indicating where the table caption should be placed.

Header 1	Header 2
Cell 1	Cell 2

Adding content to your site



- ▶ The APT format - images
- ▶ Place any images you need in the resources directory



Adding content to your site



- ▶ Adding FAQ using the FML format
- ▶ What is FML? An XML format for question-answer structured documents

```
<?xml version="1.0"?>
<faqs id="General FAQ">
  <part id="General">
    <faq id="where">
      <question>Where did Maven come from?</question>
      <answer>
        <p>
          Maven was created by a group of software developers who were tired
          of wasting their time fiddling around with builds and wanted to get
          down to brass tacks and actually develop software!
        </p>
      </answer>
    </faq>
    <faq id="why">
      <question>Why is maven so wildly popular?</question>
      <answer>
        <p>
          Maven saves you so much time in your software development efforts that
          you will have time to learn a second language, relax ten hours a day, and train for that
          marathon you've always wanted to run!
        </p>
      </answer>
    </faq>
  </part>
</faqs>
```

FAQs can be organized into “parts”

Each FAQ has a question and an answer

FAQ

- 1. Where did Maven come from?
- 2. Why is maven so wildly popular?

Where did Maven come from?

Maven was created by a group of software developers who were tired of wasting their time fiddling around with builds and wanted to get down to brass tacks and actually develop software!

[\[top\]](#)

Why is maven so wildly popular?

Maven saves you so much time in your software development efforts that you will have time to learn a second language, relax ten hours a day, and train for that marathon you've always wanted to run!

[\[top\]](#)

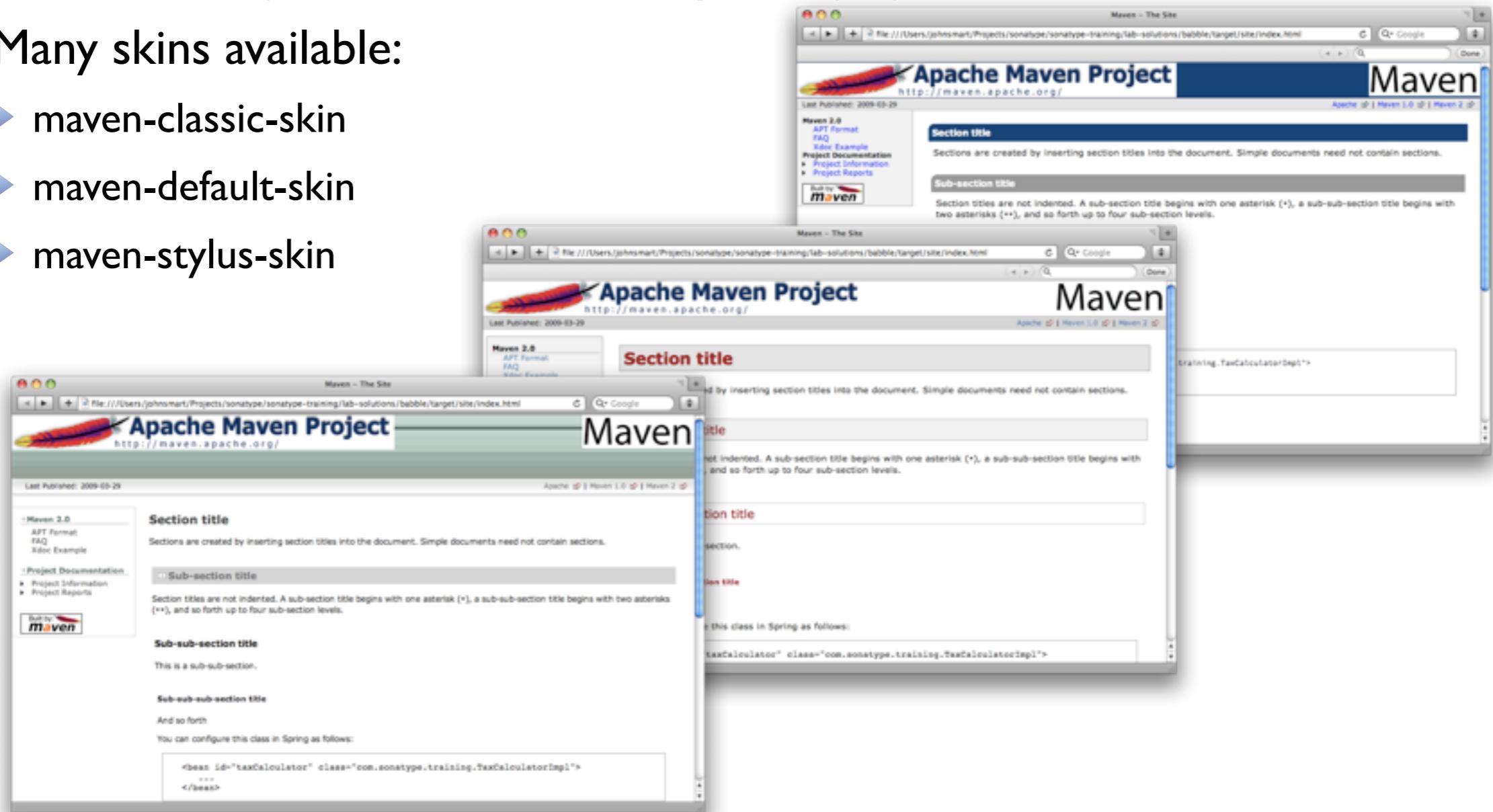


**Site
look & feel**

Changing the look and feel



- ▶ Using an alternate skin
- ▶ You can change the look and feel by changing skins
- ▶ Many skins available:
 - ▶ maven-classic-skin
 - ▶ maven-default-skin
 - ▶ maven-stylus-skin



Changing the look and feel



- ▶ Using an alternate skin
- ▶ Change the skin in the site.xml file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="Maven">
  <skin>
    <groupId>org.apache.maven.skins</groupId>
    <artifactId>maven-stylus-skin</artifactId>
    <version>1.1</version>
  </skin>
  <bannerLeft>
    <name>Maven</name>
    <src>http://maven.apache.org/images/apac...
    <href>http://maven.apache.org/</href>
  </bannerLeft>
  <bannerRight>
    <src>http://maven.apache.org/images/mave...
  </bannerRight>
  <body>
    <links>
      <item name="Apache" href="http://www.a...
      <item name="Maven 1.0" href="http://ma...
      <item name="Maven 2" href="http://mav...
    </links>
    <menu name="Maven 2.0">
      <item name="APT Format" href="format.h...
      <item name="FAQ" href="faq.html"/>
      <item name="Xdoc Example" href="xdoc.h...
    </menu>
    <menu ref="reports" />
  </body>
</project>
```

Use the “stylus” skin

Maven – The Site

Apache Maven Project
http://maven.apache.org/

Last Published: 2009-03-29

Apache | Maven 1.0 | Maven 2

Section title

Sections are created by inserting section titles into the document. Simple documents need not contain sections.

Sub-section title

Section titles are not indented. A sub-section title begins with one asterisk (*), a sub-sub-section title begins with two asterisks (**), and so forth up to four sub-section levels.

Sub-sub-section title

This is a sub-sub-section.

Sub-sub-sub-section title

And so forth

You can configure this class in Spring as follows:

```
<bean id="taxCalculator" class="com.sonatype.training.TaxCalculatorImpl">
  ...
</bean>
```



**Running the
site**

Developing your site



- ▶ Use **mvn site:run** for fast turn-around
 - ▶ Run the site using Jetty
 - ▶ Get quick feedback on content changes

\$ mvn site:run
...
[INFO] Starting Jetty on <http://localhost:8080>/

2009-03-29 22:11:09.078::INFO: jetty-6.1.5
2009-03-29 22:11:09.458::INFO: NO JSP Supp
2009-03-29 22:11:09.660::INFO: Started Sel

Apache Maven Project
<http://maven.apache.org/>

Maven - The Site

Section title

Sub-section title

Sub-sub-section title

This is a sub-sub-section.

Sub-sub-sub-section title

And so forth

One error in opening the page. For more information, choose Window > Activity.

Site runs on <http://localhost:8080>

Deploying your site



- ▶ Distributing your site
 - ▶ Copy site contents to a remote server
 - ▶ Configured in the `<distributionManagement>` section of your `pom.xml` file

```
<project>
  ...
  <distributionManagement>
    <site>
      <id>internal.website</id>
      <url>scp://devserver.mycompany.com/var/www/projects/babble</url>
    </site>
  </distributionManagement>
  ...
</project>
```

Uses common protocols such as
FTP, SCP and DAV

Define username and password
details in the `settings.xml` file

```
<settings>
  ...
  <servers>
    <server>
      <id>internal.website</id>
      <username>apache</username>
      <password>t0ps3cr3t</password>
    </server>
    ...
  </servers>
  ...
</settings>
```



site PDFs

PDF Generation



- ▶ PDF generating standalone plugin, reporting plugin
- ▶ Can generate an aggregate of all submodule sites
- ▶ `mvn site pdf:pdf`
or
`<reports>`
 `<report>maven-pdf-plugin</report>`
`<reports>`



**Sample Project - Wicket With
Unneeded Dependencies**

v. 1.0-SNAPSHOT

Project Documentation

mccm06

2010-06-28

Maven Intermediate Concepts

Part 8

Reporting



Site Reporting



- ▶ Generating Reports with Maven
 - ▶ New reports are easy to integrate using the reporting plugins
 - ▶ Many reporting plugins available
 - ▶ Javadoc
 - ▶ Test results
 - ▶ Code quality (Checkstyle, PMD, Findbugs...)
 - ▶ Code coverage (Cobertura, Emma, Clover...)
 - ▶ Change logs
 - ▶ TODO lists
 - ▶ ..
 - ▶ Report data can also be used by other tools like Hudson or Sonar

Technical Reports

Technical documentation



- ▶ Automated technical documentation in Java
 - ▶ Javadoc
 - ▶ UMLGraph - *UML embedded in your Javadocs*
 - ▶ DOxygen - *a more complete alternative to Javadoc*

Technical documentation



- ▶ Integrating Javadoc
- ▶ Simply add the maven-javadoc-plugin:

```
<project...>
  <reporting>
    <plugins>
      <plugin>
        <artifactId>maven-javadoc-plugin</artifactId>
      </plugin>
      ...
    </plugins>
  </reporting>
</project>
```

The screenshot shows a Java API documentation interface. On the left, there's a sidebar with navigation links: 'All Classes', 'Packages' (listing com.wakaleo.tax.taxcalculator, com.wakaleo.tax.taxcalculator, com.wakaleo.tax.taxcalculator), 'Interfaces' (listing TaxCalculator), and 'Exceptions' (listing InvalidYearException). The main content area has a header bar with tabs: Overview, Package, Class (which is selected and highlighted in blue), Use, Tree, Deprecated, Index, Help. Below the tabs, there are links for PREV CLASS, NEXT CLASS, SUMMARY, NESTED, FIELD, CONSTR, and METHOD. There are also buttons for FRAMES, NO FRAMES, and DETAIL. The main content area displays the 'Interface TaxCalculator' with its methods: calculateGST and calculateIncomeTax.

com.wakaleo.tax.taxcalculatorcore.service

Interface TaxCalculator

All Known Implementing Classes:
[TaxCalculatorImpl](#)

public interface TaxCalculator

Method Summary

double	calculateGST(double price)
double	calculateIncomeTax(double income, int year)

Method Detail

Technical documentation



- ▶ More advanced Javadoc
- ▶ Javadoc with multi-module projects

```
<project...>
  <reporting>
    <plugins>
      <plugin>
        <artifactId>maven-javadoc-plugin</artifactId>
        <configuration>
          <aggregate>true</aggregate>
        </configuration>
      </plugin>
      ...
    </plugins>
  </reporting>
</project>
```

Combine Javadoc from all
the modules into one site

Browsable source code



- ▶ Integrating JXR
- ▶ Cross-referenced HTML source code
- ▶ Useful for integration with other reports

```
<project...>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jxr-plugin</artifactId>
        <configuration>
          <aggregate>true</aggregate>
        </configuration>
      </plugin>
      ...
    </plugins>
  </reporting>
</project>
```

```
All Classes
Packages
com.wakaleo.tax.taxcalculator
com.wakaleo.tax.taxcalculator
com.wakaleo.tax.taxcalculator

All Classes
App
InvalidYearException
TaxCalculator
TaxCalculatorImpl
TaxRate

View Javadoc
package com.wakaleo.tax.taxcalculatorcore.service.impl;

import java.util.ArrayList;
import java.util.List;

import org.apache.log4j.Logger;
import org.joda.time.DateTime;

import com.wakaleo.tax.taxcalculator;
import com.wakaleo.tax.taxcalculatorcore;
import com.wakaleo.tax.taxcalculatorcore.service;
import com.wakaleo.tax.taxcalculatorcore.service.impl;

/**
 * @opt inferretype
 */
public class TaxCalculatorImpl implements TaxCalculator {

    public static final List<TaxRate> TAX_RATES = new ArrayList<TaxRate>();
    private static final double GST_RATE = 0.125;

    static {
        TAX_RATES.add(new TaxRate(0, 38000, 0.195));
        TAX_RATES.add(new TaxRate(38000, 60000, 0.33));
        TAX_RATES.add(new TaxRate(60000, 0, 0.39));
    }

    private List<TaxRate> taxRates;

    public List<TaxRate> getTaxRates() {
        if (taxRates == null) {
            taxRates = TAX_RATES;
        }
        return taxRates;
    }

}
```

Links to Javadoc

Links to other classes

Displaying test results



- ▶ The surefire reports plugin
 - ▶ Default configuration good for most situations
 - ▶ You need to configure for multi-module projects

```
<project...>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-report-plugin</artifactId>
        <configuration>
          <aggregate>true</aggregate>
        </configuration>
      </plugin>
      ...
    </plugins>
  </reporting>
</project>
```

The screenshot shows a web browser window titled "Maven - Surefire Report". The URL is <http://taronga/projects/babble/surefire-report.html>. The page is branded with the Sonatype logo and "the maven company". It includes a sidebar with links like "Getting started", "Architecture and design", and "Project Documentation". The main content area is titled "Surefire Report" and contains a "Summary" section with a table:

Tests	Errors	Failures	Skipped	Success Rate	Time
5	0	0	0	100%	0.079

A note below the table states: "Note: failures are anticipated and checked for with assertions while errors are unanticipated."

Below the summary is a "Package List" section with a table:

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.sonatype.training.babble.domain	4	0	0	0	100%	0.045
com.sonatype.training.babble.services	1	0	0	0	100%	0.034

A note below the package list states: "Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers."

Under the package list, there are two expanded sections: "com.sonatype.training.babble.domain" and "com.sonatype.training.babble.services", each with its own table:

com.sonatype.training.babble.domain

Class	Tests	Errors	Failures	Skipped	Success Rate	Time
BabblerTest	4	0	0	0	100%	0.045

com.sonatype.training.babble.services

Class	Tests	Errors	Failures	Skipped	Success Rate	Time
BabbleManagerTest	1	0	0	0	100%	0.034

Code Quality metrics

Code quality metrics

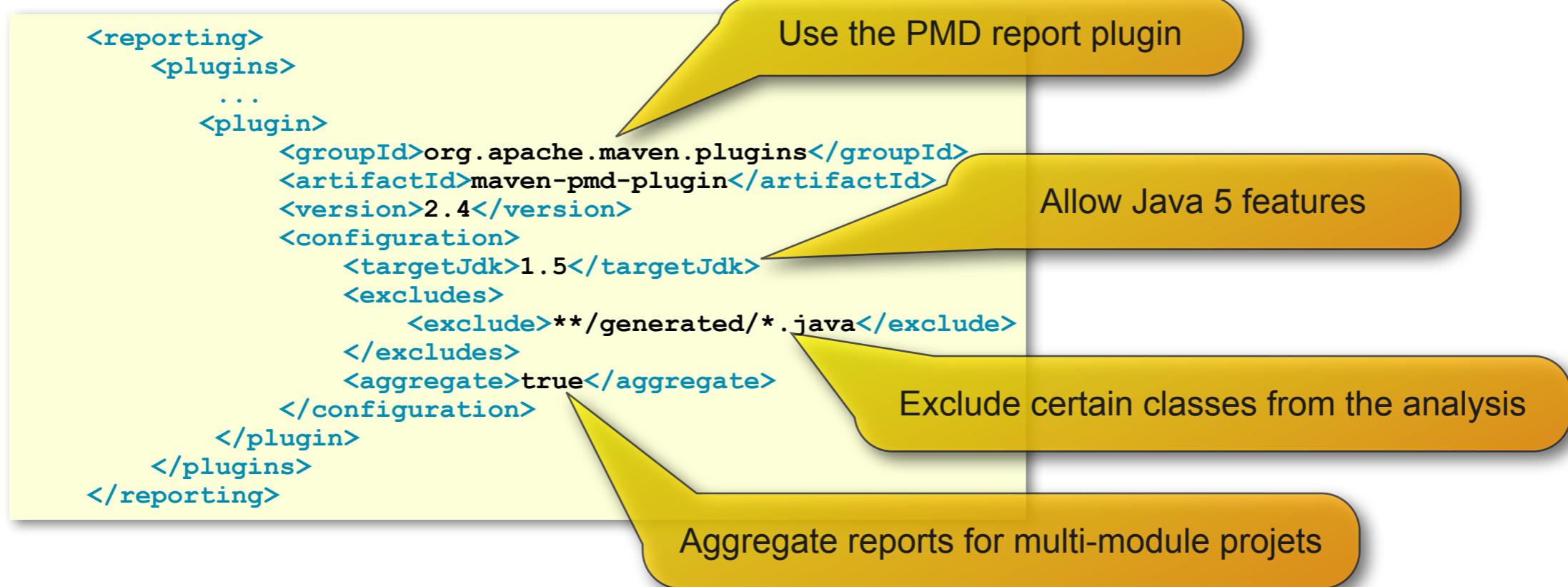


- ▶ Many code quality and code coverage metrics tools available
 - ▶ Checkstyle
 - ▶ PMD
 - ▶ FindBugs
 - ▶ Crap4j (Change Risk Analysis and Predictions)
 - ▶ Clover
 - ▶ Cobertura
 - ▶ ...
- ▶ Most generate results in HTML and XML
- ▶ XML data can be used by other tools (e.g. Hudson)

Code quality metrics



- ▶ Code quality reports - some examples
- ▶ Enforcing best practices - PMD



Code quality metrics



- ▶ Code quality reports - some examples
- ▶ Enforcing best practices - PMD

The screenshot shows a Maven build interface with two main windows:

- Maven - PMD Results (Left Window):** A web browser window displaying the Sonatype Maven project's PMD results. The title bar says "Maven - PMD Results". The URL is "http://taronga/projects/babble/pmd.html". The page header includes the Sonatype logo and navigation links for "Home", "Installing the project", "FAQ", "High-level architecture", "Design notes", "Project Information", "Project Reports", "CPD Report", "JavaDocs", "PMD Report", "Source Xref", "Surefire Report", "Test JavaDocs", and "Test Source Xref". A "Built by maven" badge is visible. The main content area is titled "PMD Results" and states "The following document contains the results of PMD 4.2.2." It lists violations under "Files" and "Violation".

JXR links to source code

Aggregated PMD report
- BabbleManager xref (Right Window):** A code editor window showing the Java source code for "BabbleManager". The title bar says "BabbleManager xref". The URL is "http://taronga/projects/babble/xref/com/sonatype/training/babble/services/BabbleManager.html#8". The code is:

```
/*
 * To change this template, choose Tools / Templates
 * and open the template in the editor.
 */
package com.sonatype.training.babble.services;

import com.sonatype.training.babble.domain.Babble;
import com.sonatype.training.babble.domain.Babbler;
import java.util.List;
import java.util.ArrayList;

/**
 * A service class that acts as a broker for babble messages.
 * @author johnsmart
 */
public class BabbleManager {
    private List<Babbler> registeredBabblers = new ArrayList<Babbler>();
    public Babbler register(String name) throws NameAlreadyExistsException{
        return new Babbler(name);
    }
}
```

This page was automatically generated by Maven

Code quality metrics



- ▶ Code quality reports - some examples
- ▶ Enforcing coding standards - Checkstyle

```
<reporting>
  <plugins>
    ...
    <plugin>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <configuration>
        <configLocation>
          http://svnserver/config/company-coding-standards.xml
        </configLocation>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

A checkstyle report

NOTE: Checkstyle reports are not aggregated

Enterprise-specific rules

The screenshot shows a web-based Checkstyle report titled "Checkstyle Results". It includes a summary table showing 2 files, 3 Infos, 12 Warnings, and 42 Errors. Below the summary are sections for "Files" and "Rules". The "Files" section lists two Java files: "com/sonatype/training/babble/domain/Babble.java" and "com/sonatype/training/babble/domain/Babbler.java", each with its respective error counts. The "Rules" section lists specific Java code style violations, such as "LeftCurly" and "RightCurly", along with their violation counts and severities.

Files	Infos ⓘ	Warnings⚠	Errors✖
2	3	12	42

Rules	Violations	Severity
LeftCurly • option: "s1"	14	Error
RightCurly • option: "alone"	0	Error

Overview**Introduction**

Goals

Usage

Examples

Report Formats

Sample Cobertura

Report

Project Documentation

▶ Project Information

▶ Project Reports



Cobertura Maven Plugin

This plugin provides the features of [Cobertura](#) within the Maven 2 environment.

The report generated by this plugin is the result of executing the Cobertura tool against your compiled classes to help you determine how well the unit testing efforts have been, and can then be used to identify which parts of your Java program are lacking test coverage.

Goals Overview

- [cobertura:check](#) Check the Last Instrumentation Results.
- [cobertura:clean](#) Clean up rogue files that `cobertura` maven plugin is tracking.
- [cobertura:dump-datafile](#) Cobertura Datafile Dump Mojo.
- [cobertura:instrument](#) Instrument the compiled classes.
- [cobertura:cobertura](#) Instruments, Tests, and Generates a Cobertura Report.

Usage

Instructions on how to use the Cobertura Maven Plugin can be found on the [usage page](#).

Examples

To provide you with better understanding of some usages of the Cobertura Maven Plugin, you can take a look into the following example:

- [Report Formats](#)

Code quality metrics



- ▶ Code quality reports - some examples
- ▶ Measuring code coverage - Cobertura

```
<reporting>
  <plugins>
    ...
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>cobertura-maven-plugin</artifactId>
      <version>2.2</version>
    </plugin>
  </plugins>
</reporting>
```

The screenshot shows a 'Coverage Report' window for a Maven project. On the left, a sidebar displays the Maven configuration for the Cobertura plugin. The main area shows a hierarchical tree of packages and classes, with coverage percentages. A yellow callout points to the package 'com.sonatype.training.babble.domain' with the text 'Project-level coverage (no aggregation)'. Another yellow callout points to a table of class-level coverage data with the text 'Drill-down test coverage metrics'.

Package	# Classes	Line Coverage	Branch Coverage	Complexity
com.sonatype.training.babble.domain	2	58% / 14/24	N/A	1
Classes in this Package	Line Coverage	Branch Coverage	Complexity	
Babble	50% / 6/12	N/A	1	
Babbler	67% / 8/12	N/A	0	

Report generated by Cobertura 1.9 on 3/30/09 11:52 AM.

Project-level coverage (no aggregation)

Drill-down test coverage metrics

Cobertura



- ▶ Plugin Homepage
 - ▶ <http://mojo.codehaus.org/cobertura-maven-plugin/>
- ▶ Usage
 - ▶ <http://mojo.codehaus.org/cobertura-maven-plugin/usage.html>

Sonar &
maven

Sonar



- ▶ Complementary tool to Maven Reports, Hudson
- ▶ Adds time dimension to statistics
- ▶ Effortless integration with Maven builds
- ▶ Scalable from 1 project to 1000 projects

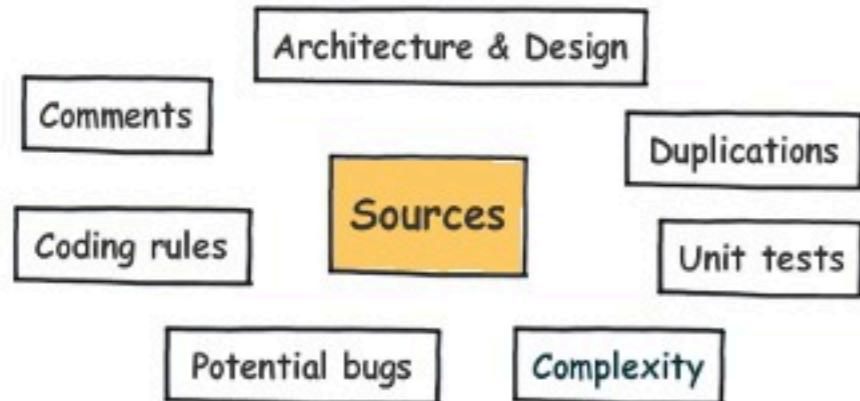
Put your technical debt under control

Productivity is falling ?
Confess your source code to clean it up !

► [Mission](#) ► [Platform](#) ► [Figures](#)

All in one

Sonar is an open platform to manage code quality. As such, it covers the 7 axes of code quality:



Extend with plugins

Covering new languages, adding rules engines, computing advanced metrics can be done through a powerful extension mechanism. More than [30 plugins](#) are already available.

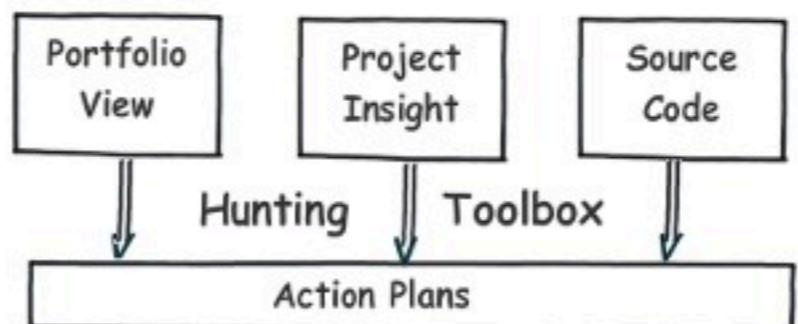
Languages covered

Java is built in. Plugins enable to cover [PL/SQL](#) and C++.



In 3 clicks

Sonar has got a very efficient way of navigating, a balance between high-level view, dashboard, TimeMachine and defect hunting tools. This enables to quickly uncover projects and / or components that are in Technical Debt to establish action plans.



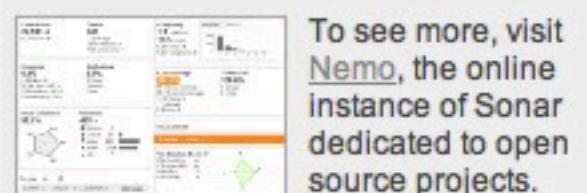
Quality is central

Sonar is a web-based application. Rules, alerts, thresholds, exclusions, settings... can be configured online. By leveraging its database, Sonar not only allows to combine metrics altogether but also to mix them with historical measures.

Get started

1. [Download latest version \(2.0 on March 10, 2010\)](#)
2. Unzip and start
3. [Analyze projects](#)
4. Ready to improve quality

Sonar in action



To see more, visit [Nemo](#), the online instance of Sonar dedicated to open source projects.

Blog

Subscribe to our blog by [email](#) or directly through the [feed](#)

- [Fight Back Design Erosion by Breaking Cycles with Sonar](#)
- [Sonar 2.0 in screenshots](#)
- [Sonar in the news](#)

Sonar Execution



- ▶ Start Sonar
 - ▶ sonar.sh start
- ▶ Clean execution to ensure Sonar analysis will succeed
 - ▶ mvn clean install -Dtest=false -DfailIfNoTests=false
- ▶ Sonar analysis
 - ▶ mvn sonar:sonar

```
[INFO] Execute maven plugin maven-pmd-plugin done: 2089 ms
[INFO] Sensor PmdSensor...
[INFO] Sensor PmdSensor done: 237 ms
[INFO] Sensor ProfileSensor...
[INFO] Sensor ProfileSensor done: 22 ms
[INFO] Sensor ProjectLinksSensor...
[INFO] Sensor ProjectLinksSensor done: 43 ms
[INFO] Sensor VersionEventsSensor...
[INFO] Sensor VersionEventsSensor done: 111 ms
[INFO] Sensor Maven dependencies...
[INFO] Sensor Maven dependencies done: 110 ms
[INFO] Sensor SurefireSensor...
[INFO] parsing /Users/mccm06/Documents/Teach/Courses/Mastering-Maven-2.0/examples/maven-training_git/sample02-dependency/target/surefire-reports
[INFO] Sensor SurefireSensor done: 16 ms
[INFO] Execute decorators...
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000
[INFO] Database optimization...
[INFO] Database optimization done: 381 ms
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 minute 13 seconds
[INFO] Finished at: Sun Jun 27 23:32:46 MDT 2010
[INFO] Final Memory: 42M/89M
[INFO] -----
[sample02-dependency (master)]>
```

The Sonar Dashboard

► <http://localhost:9000>



Projects

Motion chart

Radiator

Views



Name	Lines of code	Technical Debt ratio	Coverage	Duplicated lines (%)	Build time	JBoss Application Server	OpenEJB	OpenJPA	Apache Tomcat	Hibernate Core	JOnAS
Name	Lines of code	Technical Debt ratio	Coverage	Duplicated lines (%)	Build time	Parent	POM	Parent	Tomcat	Core	Aggregation
MasterProject	5,983,917	17.0%	18.4%	6.6%	2010-03-17	Despot	Esper	Wicket	Camel	Struts 2	Apache Cocoon
AisLib application framework	12,187	9.6%	38.6%	1.1%	2010-03-13	OpenNMS	Apache Felix	JBoss Seam	Hudson	Maestro	Earl Grey
Tapestry 5 Project	57,213 ▲	6.7%	51.8%	0.2%	2010-03-15	Apache Jackrabbit	Squibbler	JBoss Seam	Pluto	Apache Axis	grizzly
Commons Collections	20,901	10.1% ▲	79.8%	3.3%	2010-03-14	ActiveMQ	Silverpeafowl	JBoss Seam	Apache Axis	Apache Axis 2	Apache Axis 5
MicroEmulator	28,344	11.7% ▲		2.3%	2010-03-14	Jetspeed 2	Apache Sling	jBPM	Apache Axis 2	Apache Axis 2	Apache Axis 2
Apache Jackrabbit	206,604 ▲	16.6%	26.4%	4.6%	2010-03-14	Jahia Project Root	Jetty ::	Apache Axis 2	Apache Axis 2	Apache Axis 2	Apache Axis 2
Units	18,585	14.9%	2.8%	3.8%	2010-03-15	XWiki Platform	Sonar Nexus	Apache Axis 2	Apache Axis 2	Apache Axis 2	Apache Axis 2
javagit	6,127	11.1% ▲	61.2%	0.3%	2010-03-14	Apache CXF	Castor fujitsu	Apache Axis 2	Apache Axis 2	Apache Axis 2	Apache Axis 2
Wicket Parent	104,703	9.3%	44.7%	1.1%	2010-03-15						
Commons Chain	3,901	14.4% ▼	64.5%	21.9%	2010-03-14						
Commons IO	6,779 ▲	5.2% ▲	82.0%	2.3%	2010-03-14						
Commons SCXML	7,331	9.8% ▲	69.8%	7.2%	2010-03-14						
Sonar	31,558 ▲	6.2%	68.6%	0.0%	2010-03-17						

Size

Lines of code

Color 0.0% 100.0%

Rules compliance

Projects

Motion chart

Radiator

Views



Period: Six months

Lin

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

▶

nemo



- ▶ Public instance, analyzing open source projects
- ▶ <http://nemo.sonarsource.org/>

The screenshot shows the Sonar web interface with the URL <http://nemo.sonarsource.org/> in the address bar. The page title is "Sonar". On the left, there's a sidebar with links for Home, Search, and a "MasterProject" section. The "MasterProject" section lists various open-source projects with their names, lines of code, technical debt ratio, coverage, and duplicated lines percentage. A large heatmap matrix on the right displays the size and rules compliance for many of these projects. The heatmap uses a color scale from green (low) to red (high).

Name	Lines of code	Technical Debt ratio	Coverage	Duplicated lines (%)	Build time
MasterProject	6,144,324	16.9%	19.0%	6.4%	2010-06-08
AisLib application framework	12,187	9.7%	38.6%	1.1%	2010-06-26
Tapestry 5 Project	60,824 ▲	6.9%	50.4%	0.2%	04:18
Commons Collections	20,852 ▼	10.1%	79.9%	3.3%	2010-06-27
MicroEmulator	28,344	11.8%		2.3%	2010-06-27
Apache Jackrabbit	209,864 ▲	16.7%	26.5%	4.6%	2010-06-27
Units	19,276 ▲	14.7%	2.7%	3.7%	04:45
Javagit	6,127	11.1%	61.2%	0.3%	2010-06-27
Wicket Parent	104,900	9.8%	36.9%	1.1%	05:01
Commons Chain	3,901	14.4%	64.5%	21.9%	2010-06-27
Commons IO	6,827	5.5%	82.3%	3.1%	2010-06-27
Commons SCXML	7,331	9.9%	69.8%	7.2%	2010-06-27
Sonar	36,952 ▲	6.9%	65.8%	0.0%	01:37
Apache Velocity	29,094	11.1%		5.1%	04:53
Apache Abdera	49,960 ▼	8.9%		2.9%	2010-06-26
Castor	112,001 ▼	9.1%		6.9%	2010-05-16
Commons BeanUtils	11,374	15.0%	62.8%	6.6%	2010-06-27
JEuclid	12,664	4.1%		0.5%	2010-06-27
Jackcess	11,747 ▲	3.6% ▼	86.1%	0.1%	2010-06-27
jcrom	3,958	9.5%	76.5%	2.9%	2010-06-27
Apache Tomcat	162,143 ▲	11.6%		8.4%	04:36
Commons Logging	2,695	28.2%	2.0%	6.7%	2010-06-27
Commons Digester	5,591	6.7%	70.5%	0.7%	2010-06-27
Struts 2	101,245 ▲	14.6%	33.4%	3.4%	03:40
GreenPepper Open	12,875 ▲	6.7%	75.0% ▲	1.5%	2010-06-27
Archiva	22,154 ▲	9.2%	57.7%	2.0%	2010-06-26
Apache Cocoon (build root)	100,867	16.9%	10.2%	2.6%	2010-06-27
Logback-Parent	20,377	9.0%	52.3%	1.0%	2010-06-27

Sonar References



- ▶ Homepage
 - ▶ <http://sonar.codehaus.org/>
- ▶ Usage Guide
 - ▶ <http://docs.codehaus.org/display/SONAR/Collect+data>

labs

- ▶ Lab 12 - Adding reports to your site



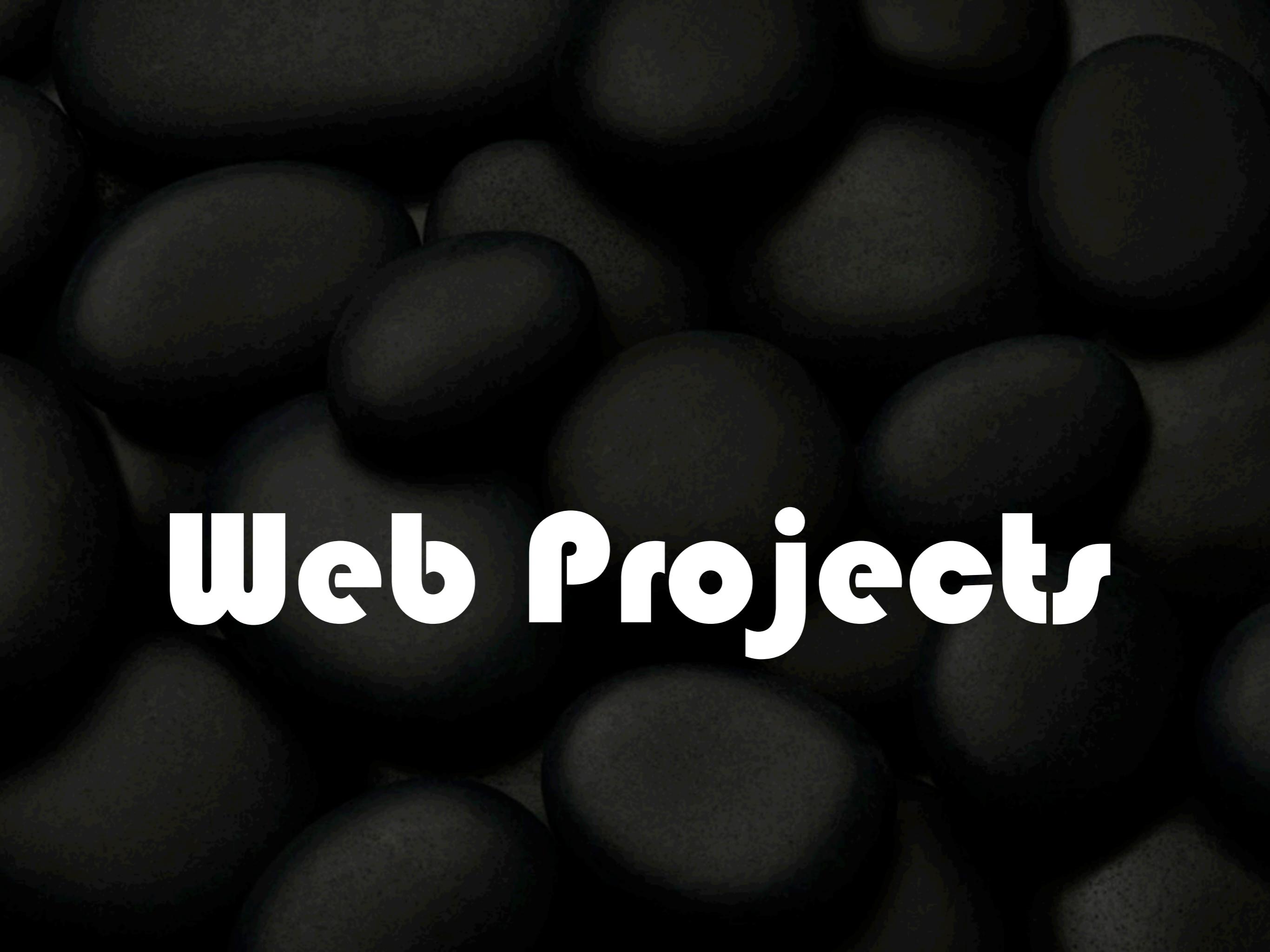


Maven Intermediate Concepts

Part 9

*Developing Web
Applications with Maven*



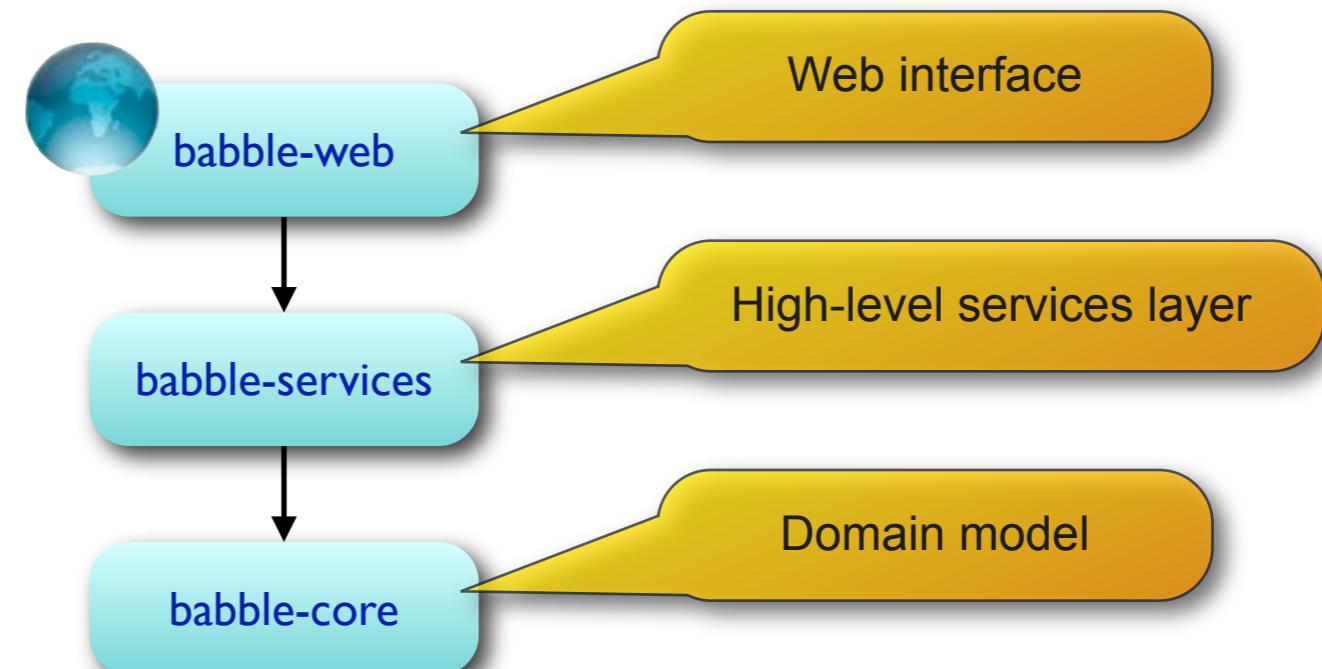


web Projects

Web projects in Maven



- ▶ Writing web applications in Maven
 - ▶ Use a separate web app module
 - ▶ Separation of concerns
 - ▶ Isolate the display technology from the business code
 - ▶ Cleaner and more flexible architecture

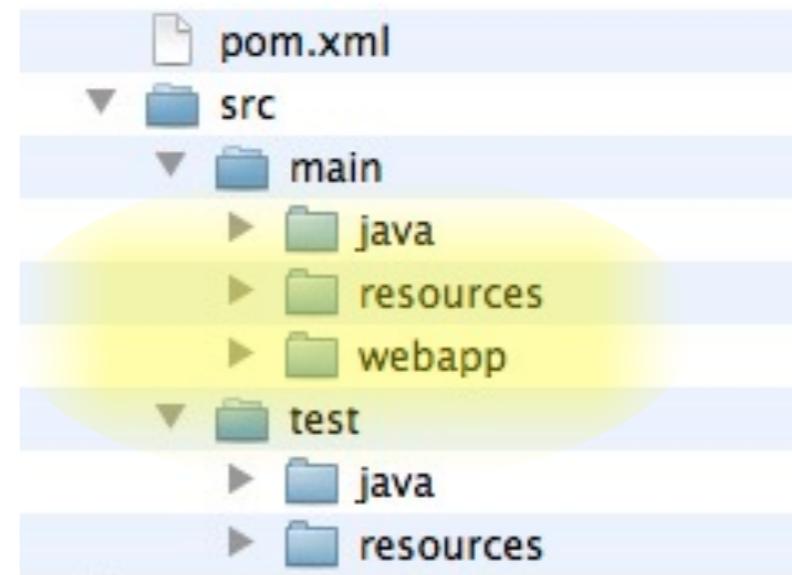


Web projects in Maven



- ▶ The Maven Webapp Directory Structure
- ▶ Production source code goes in src/main

src/main/java	Java source code
src/main/resource	Other resources your application needs
src/main/webapp	The web application directory for a WAR project

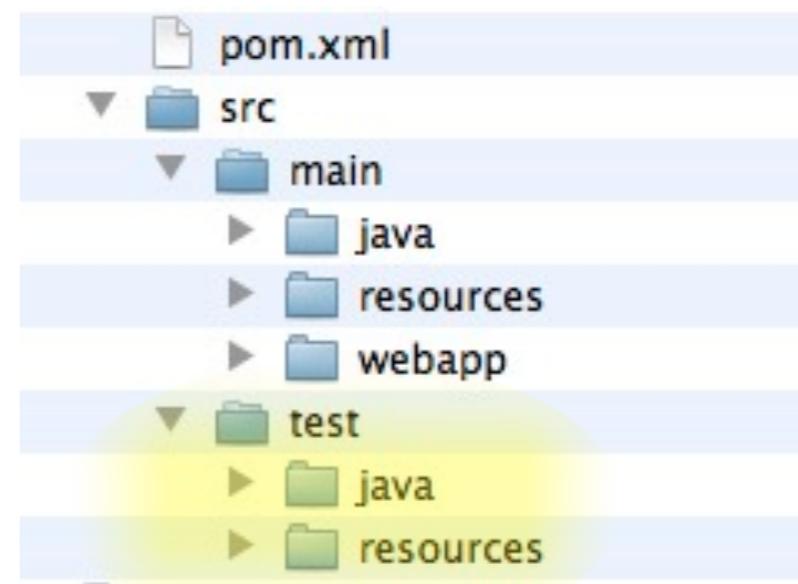


Web projects in Maven



- ▶ The Maven Webapp Directory Structure
- ▶ Test source code goes in `src/test`

<code>src/test/java</code>	Test classes source code
<code>src/test/resource</code>	Resources used for testing
<code>src/test/filters</code>	Filters to be used for testing purposes

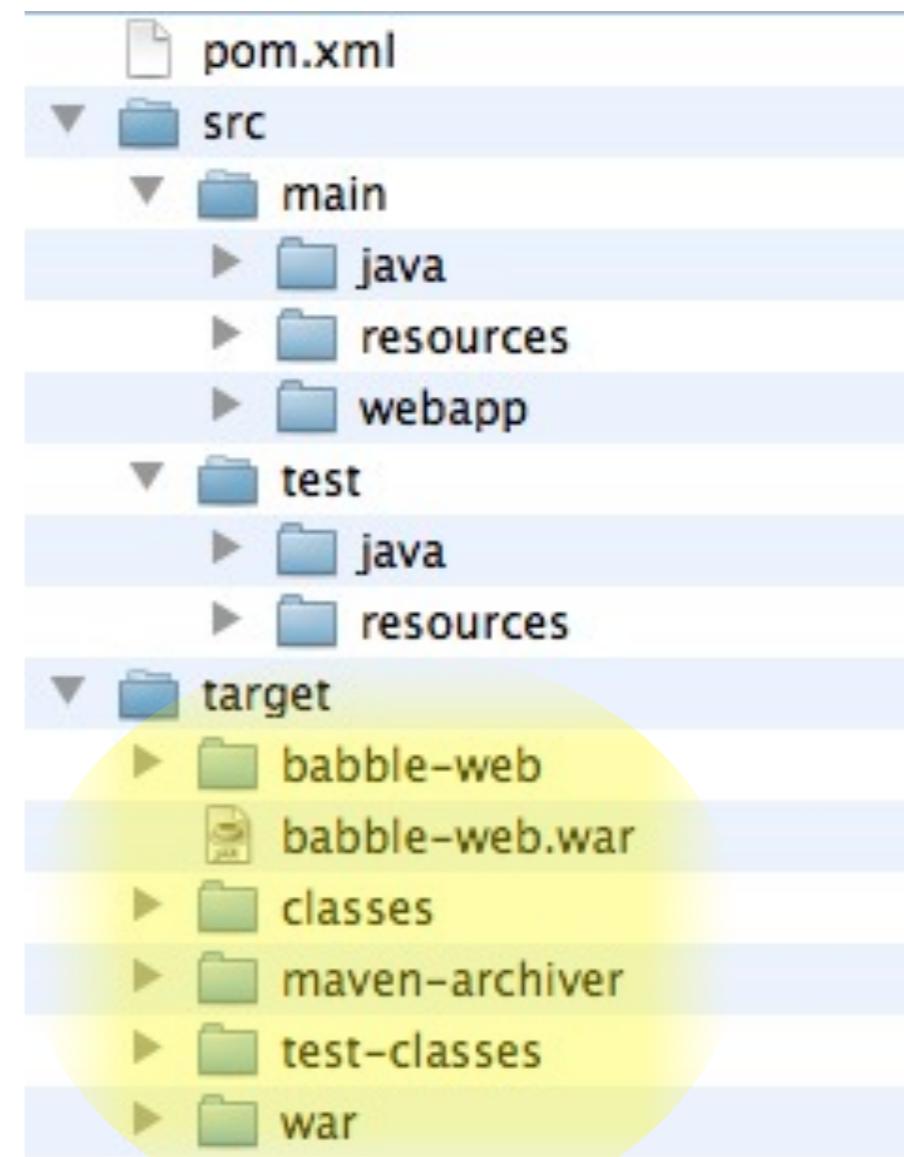


Web projects in Maven



- ▶ The Maven Webapp Directory Structure
- ▶ Compiled code and reports go in target

target/classes	Compiled Java classes
target/test-classes	Compiled test classes
target/surefire-reports	Unit test reports
target/war	Working directory
target/babble-web	Exploded web application
target/babble-web.war	Packaged web application



Web projects in Maven



- ▶ Building a web application in Maven

- ▶ Generate a WAR file

```
$ mvn war:war or $ mvn package
```

- ▶ Generate an 'exploded' WAR file

```
$ mvn war:exploded
```

Web projects in Maven

- ▶ Testing your Maven web application
- ▶ Jetty - an open source embedable web server
 - ▶ Runs stand-alone or as an embedded server
 - ▶ Can be run from within Maven
 - ▶ Light-weight and easy to configure





Jetty

Working with Jetty



- ▶ Enabling Jetty in your web project
- ▶ Jetty is simple to configure:

```
<project>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.mortbay.jetty</groupId>
          <artifactId>maven-jetty-plugin</artifactId>
          <version>6.1.15</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
```

Use the Jetty plugin

Working with Jetty



- ▶ Enabling Jetty in your web project
- ▶ Jetty is simple to configure:

```
<project>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.mortbay.jetty</groupId>
          <artifactId>maven-jetty-plugin</artifactId>
          <version>6.1.15</version>
          <configuration>
            <scanIntervalSeconds>5</scanIntervalSeconds>
            <contextPath>/mywebapp</contextPath>
            <connectors>
              <connector implementation="org.mortbay.jetty.nio.SelectChannelConnector">
                <port>8081</port>
                <maxIdleTime>60000</maxIdleTime>
              </connector>
            </connectors>
          </configuration>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
```

Port

More advanced configuration

Context path

Working with Jetty



- ▶ Running Jetty
 - ▶ Start Jetty and leave it running
 - ▶ Web page modifications are available immediately
 - ▶ Code changes are automatically redeployed
 - ▶ Even code changes in other modules are detected!

Working with Jetty



- ▶ Running Jetty
- ▶ An example

\$ mvn jetty:run

```
$ mvn jetty:run
...
[INFO] Preparing jetty:run
...
[INFO] [jetty:run]
[INFO] Configuring Jetty for project: Tax Calculator web application
[INFO] Webapp source directory = C:\dev\work\tax-calculator\tax-calculator-webapp\src\main\webapp
[INFO] web.xml file = C:\dev\work\tax-calculator\tax-calculator-webapp\src\main\webapp\WEB-INF\web.xml
[INFO] Classes = C:\dev\work\tax-calculator\tax-calculator-webapp\target\classes
2008-04-26 17:14:55.152::INFO: Logging to STDERR via org.mortbay.log.StdErrLog
[INFO] Context path = /tax-calculator-webapp
[INFO] Tmp directory = determined at runtime
[INFO] Web defaults = org/mortbay/jetty/webapp/webdefault.xml
[INFO] Web overrides = none
[INFO] Webapp directory = C:\dev\work\tax-calculator\tax-calculator-webapp\src\main\webapp
[INFO] Starting jetty 6.1.15 ...
2008-04-26 17:14:56.226::INFO: jetty-6.1.15
2008-04-26 17:14:56.539::INFO: Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
```

Working with Jetty



- ▶ Running Integration tests on Jetty
 - ▶ Use for web tests (Selenium, JWebUnit,...)
 - ▶ Start Jetty just before the integration-test phase
 - ▶ Shut it down afterwards

```
<plugin>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>maven-jetty-plugin</artifactId>
  <version>6.1.18</version>
  <configuration>
    <scanIntervalSeconds>5</scanIntervalSeconds>
    <stopPort>9966</stopPort>
    <stopKey>foo</stopKey>
  </configuration>
  ...
</plugin>
```

Start Jetty before the integration tests

Run Jetty in the background

Shut down Jetty afterwards

Configure Jetty

```
...
<executions>
  <execution>
    <id>start-jetty</id>
    <phase>pre-integration-test</phase>
    <goals>
      <goal>run</goal>
    </goals>
    <configuration>
      <daemon>true</daemon>
    </configuration>
  </execution>
  <execution>
    <id>stop-jetty</id>
    <phase>post-integration-test</phase>
    <goals>
      <goal>stop</goal>
    </goals>
  </execution>
</executions>
</plugin>
```

labs

- ▶ Lab 13 - Running your webapp in Jetty



Maven Intermediate Concepts

Part 10

Scripting in Maven



Scripting in Maven

- ▶ Why run scripts in Maven?
 - ▶ Low level or specific procedural tasks
 - ▶ Insert custom scripting at particular points in the lifecycle
- ▶ Groovy is a great way to script Maven builds
 - ▶ Concise and readable scripting language
 - ▶ Close to Java
 - ▶ Access to the Maven Project variables



Scripting in Maven



- ▶ Groovy in Maven
- ▶ Embedding Groovy scripting

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.groovy.maven</groupId>
      <artifactId>gmaven-plugin</artifactId>
      <version>1.0-rc-5</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>execute</goal>
          </goals>
          <configuration>
            <source>
              println "Hi there I am in ${project.name}"
            </source>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

The gmaven plugin

Run this script at the
'verify' phase

Inline Groovy scripting

```
$ mvn verify
...
[INFO] [groovy:execute {execution: default}]
Hi there I am in ebank-core version 1.0.0-SNAPSHOT
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
```

Scripting in Maven



- ▶ Groovy in Maven
- ▶ GMaven gives you full access to the Maven project properties

```
...  
<configuration>  
  <source>  
    println "I am in ${project.name} version ${project.version}"  
    project.dependencies.each {  
      println "Group: ${it.groupId}; Artifact: ${it.artifactId}; Version: ${it.version}"  
    }  
  </source>  
</configuration>  
...
```

```
$ mvn verify  
...  
[INFO] [groovy:execute {execution: default}]  
I am in ebank-core version 1.0.0-SNAPSHOT  
Group: hsqldb; Artifact: hsqldb; Version: 1.8.0.7  
Group: junit; Artifact: junit; Version: 4.5  
[INFO] -----  
[INFO] BUILD SUCCESSFUL  
[INFO] -----
```

Scripting in Maven



- ▶ Groovy in Maven
- ▶ You can also call external Groovy scripts

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.groovy.maven</groupId>
      <artifactId>gmaven-plugin</artifactId>
      <version>1.0-rc-5</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>execute</goal>
          </goals>
          <configuration>
            <source>${project.basedir}/src/main/script/myscript.groovy</
source>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
```

Run a groovy script in your project

Scripting in Maven



- ▶ Groovy in Maven
- ▶ You can also call external Groovy scripts

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.groovy.maven</groupId>
      <artifactId>gmaven-plugin</artifactId>
      <version>1.0-rc-5</version>
      <executions>
        <execution>
          <phase>verify</phase>
          <goals>
            <goal>execute</goal>
          </goals>
          <configuration>
            <source>http://mydomain.com/myscript.groovy</source>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Or run a remote
Groovy script

Scripting in Maven



▶ Groovy in Maven

▶ To use Java libraries, you need to declare them in your dependencies

```
<dependencies>
  ...
  <dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.4</version>
  </dependency>
</dependencies> <configuration>
  <classpath>
    <element>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
    </element>
  </classpath>
  <source>
    import org.apache.commons.lang.SystemUtils
    println SystemUtils.IS_OS_MAC OSX ? "I'm a Mac" : "I'm not"
  </source>
</configuration>
```

Declare your dependencies in your `<dependencies>` section

Then add the dependencies to the `<classpath>`

Then import classes as usual

Maven Intermediate Concepts

Part 11

*Setting up a Maven
Repository Manager*



Enterprise Repositories

Enterprise Repositories

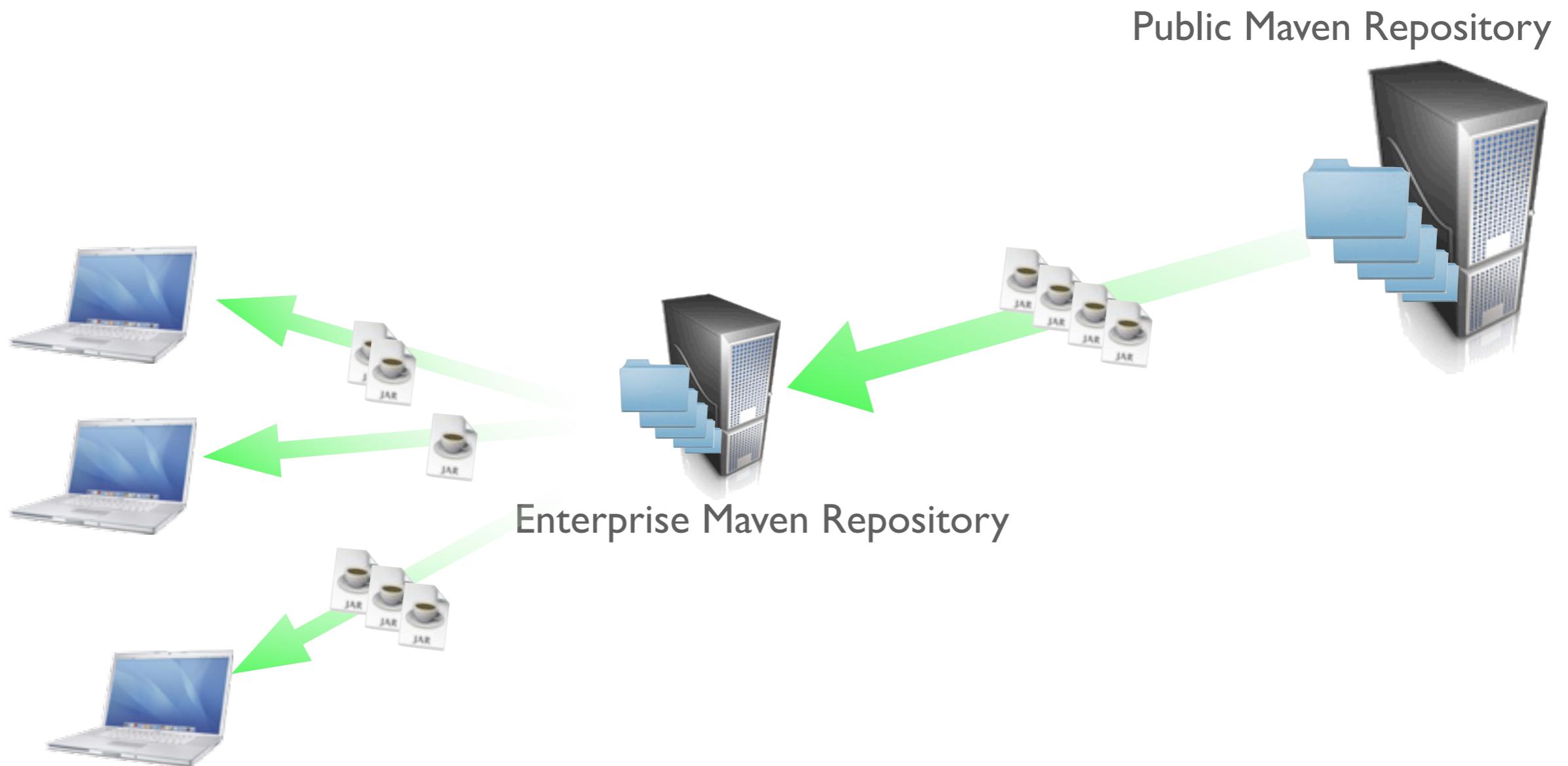


- ▶ Why use an Enterprise Repository?
 - ▶ Proxy/cache for downloading JARs from public repositories
 - ▶ Improve speed and reliability
 - ▶ Saves bandwidth
 - ▶ Allows better control over what is downloaded
 - ▶ Store and share enterprise libraries
 - ▶ Store and distribute proprietary third-party libraries

Enterprise Repositories



- ▶ How does an Enterprise Repository work?



Enterprise Repositories

- ▶ Nexus – an Enterprise Repository Manager
 - ▶ Easy to install
 - ▶ Easy to configure (web interface)
 - ▶ Easy to use







Installing
nexus

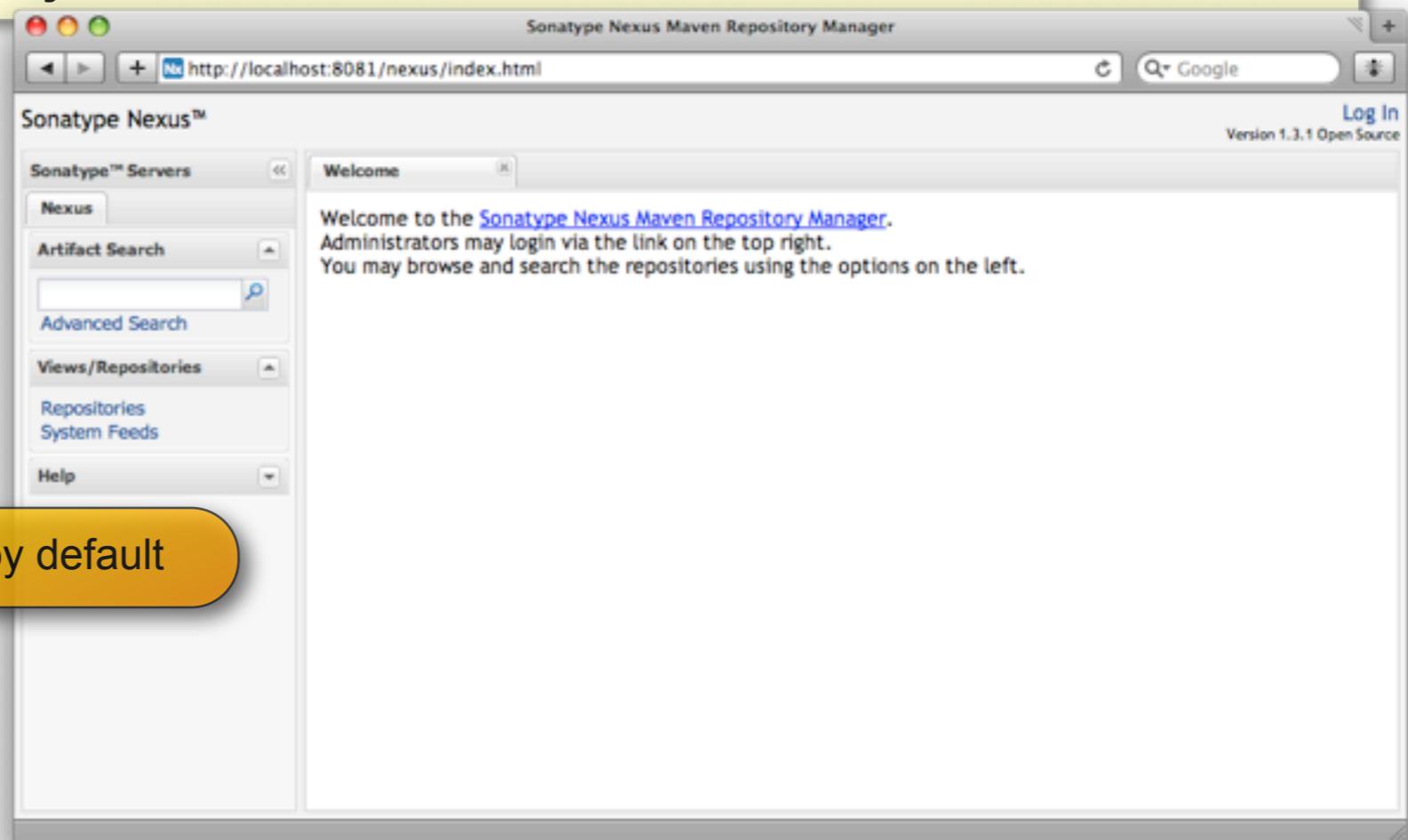
Installing Nexus



- ▶ Installing and running Nexus
 - ▶ Download and unzip the Nexus binary
 - ▶ Nexus can be run out-of-the-box (runs on Jetty)

```
$ sudo /usr/local/nexus-webapp-1.3.1/bin/jsw/macosx-universal-32/nexus start
Starting Sonatype Nexus Repository Manager...
Started Sonatype Nexus Repository Manager.
```

Choose the script for your OS



Installing Nexus

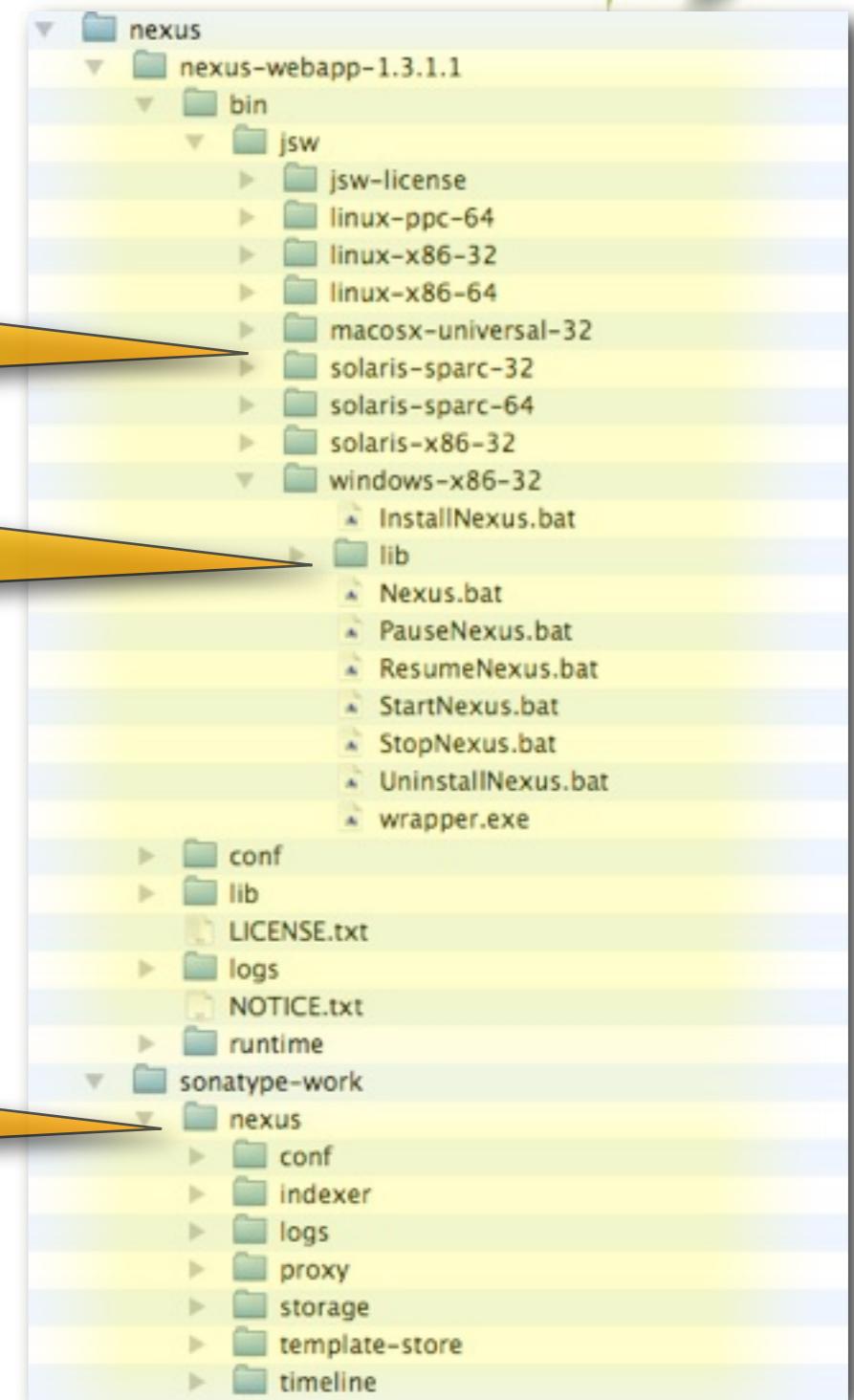


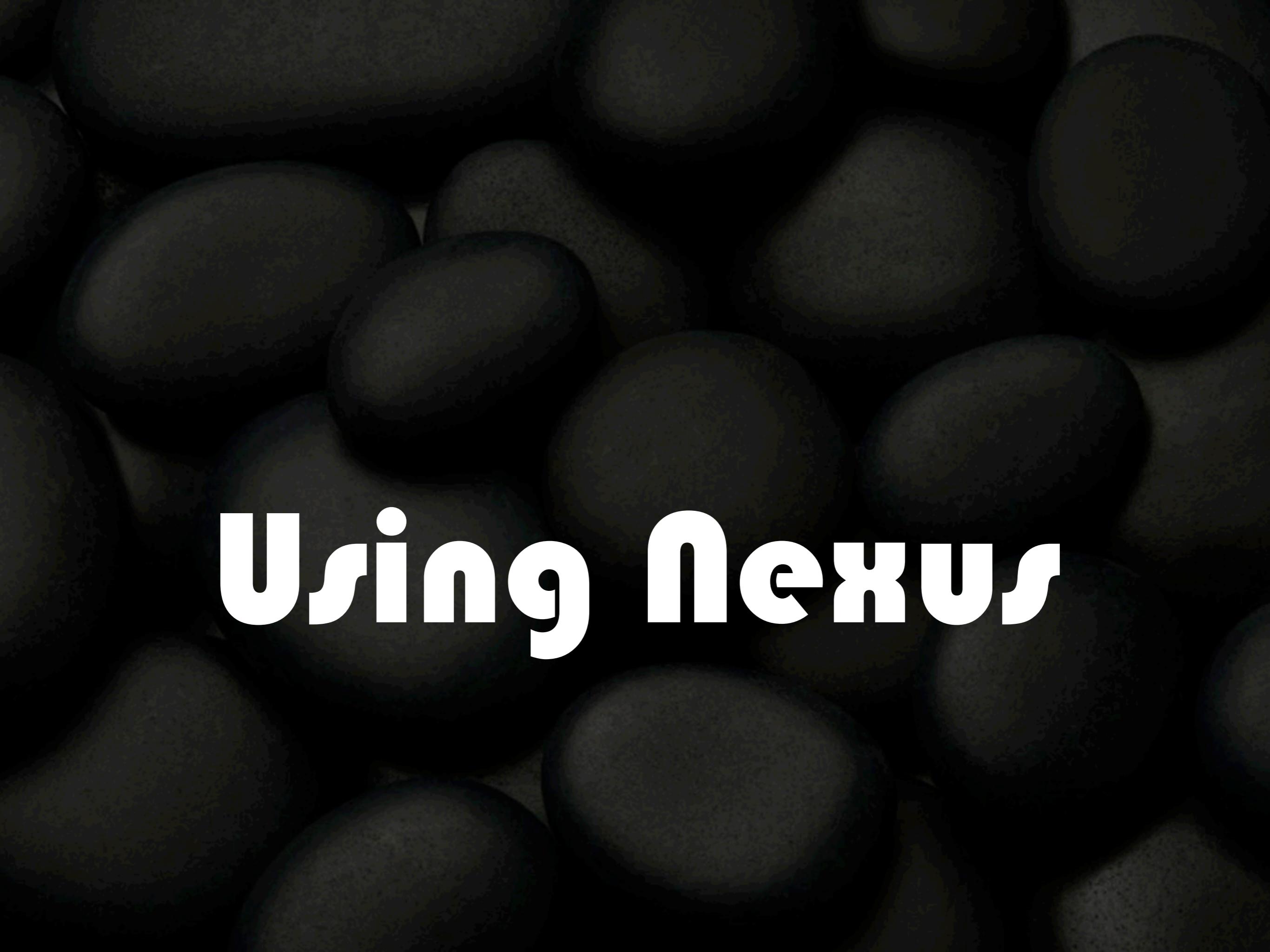
- ▶ Installing and running Nexus
- ▶ The Nexus directory structure

Nexus binaries

Scripts for each O/S

Local workspace





Using Nexus

Using Nexus



- ▶ Browsing the Nexus repositories
- ▶ Nexus comes with many repositories pre-configured

The screenshot shows the Sonatype Nexus Maven Repository Manager interface. The title bar reads "Sonatype Nexus Maven Repository Manager". The URL in the address bar is "http://localhost:8081/nexus/index.html". On the right, there is a "Log In" link and a note "Version 1.3.1 Open Source". The main content area has tabs for "Welcome" and "Repositories". The "Repositories" tab is selected and displays a table of managed repositories:

Repository	Type	Format	Policy	Repository Status	Repository Path
Public Repositories	group	maven2		Releases, Snapshots, 3rd party, Ma...	http://localhost:8081/nexus/content/groups/public
Public Snapshot Repositories	group	maven2		Apache Snapshots, Codehaus Snap...	http://localhost:8081/nexus/content/groups/public-snapshots
3rd party	hosted	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	maven1		In Service	http://localhost:8081/nexus/content/repositories/central-m1
Codehaus Snapshots	proxy	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snapshots
Maven Central	proxy	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/central
Releases	hosted	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots

Below the table, a message says "Select a record to view the details."

Using Nexus



▶ Repository Groups

- ▶ Repositories are organized into groups
- ▶ Two groups are predefined: Public Repositories and Public Snapshot Repositories

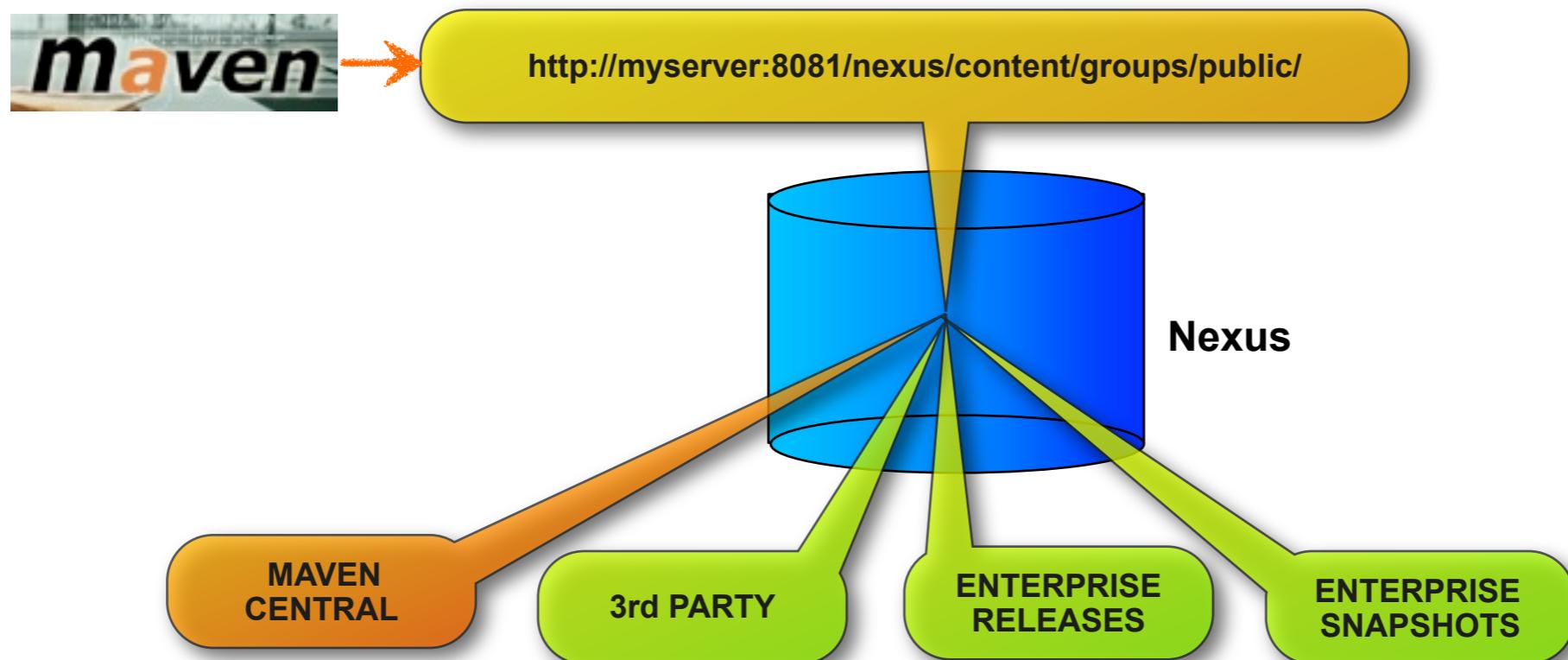
The screenshot shows the Sonatype Nexus administration interface. The top navigation bar includes links for Maintenance, Repositories, and Groups, with 'Groups' currently selected. A yellow callout bubble points to the 'Groups' tab with the text 'Enterprise releases and snapshots' and 'Public releases'. Another yellow callout bubble points to the 'Public Repositories' section with the text 'Public snapshots'. The main content area displays a table for 'Repository Group Configuration' with two entries: 'Public Repositories' and 'Public Snapshots'. Each entry has a brief description and its corresponding 'Group Path'. The bottom of the interface features a search bar and navigation buttons for 'Find', 'Next', 'Previous', 'Highlight all', and 'Match case'.

Group	Description	Group Path
Public Repositories	Releases, Snapshots, 3rd party, Maven Central	http://localhost:8081/nexus/content/groups/public
Public Snapshots	Apache Snapshots, Codehaus Snapshots	http://localhost:8081/nexus/content/groups/public-snapshots

Using Nexus



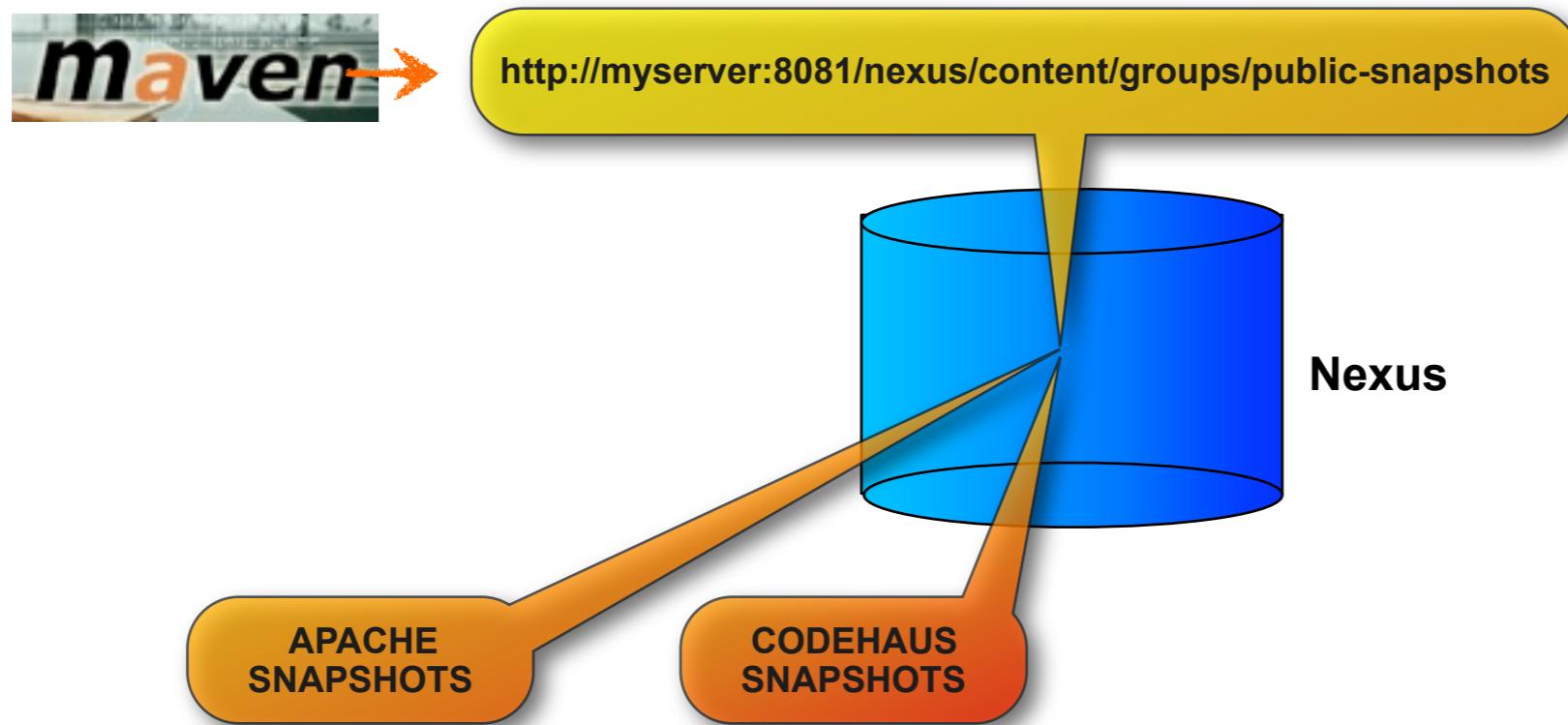
- ▶ Repository Groups
 - ▶ Groups are accessed via their URLs
 - ▶ The public repository:



Using Nexus



- ▶ Repository Groups
 - ▶ Groups are accessed via their URLs
 - ▶ The public snapshot repository:



Using Nexus



- ▶ Using Nexus with Maven
 - ▶ Define a mirror in your `~/.m2/settings.xml`
 - ▶ Maven will now look in Nexus for artifacts coming from the Maven Central repository.

```
<settings>
  ...
  <mirrors>
    <mirror>
      <id>Nexus</id>
      <name>Nexus Public Mirror</name>
      <url>http://localhost:8081/nexus/content/groups/public</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
</settings>
```

Using Nexus



- ▶ Using Nexus with Maven
 - ▶ Or use a wildcard mirror setting
 - ▶ Maven will now get ALL dependencies from here

```
<settings>
  ...
  <mirrors>
    <mirror>
      <id>Nexus</id>
      <name>Nexus Public Mirror</name>
      <url>http://localhost:8081/nexus/content/groups/public</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
</settings>
```

Managing nexus

Managing Nexus



- ▶ The Nexus Repositories
- ▶ Nexus handles several types of repository

The screenshot shows the Nexus Repository Manager interface. The top section is a table titled "Repositories" with columns: Repository, Type, Status, and Repository Path. The table lists several repositories:

Repository	Type	Status	Repository Path
3rd party	hosted	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	In Service	http://localhost:8081/nexus/content/repositories/central-m1
Codehaus Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snapshots
Maven Central	proxy	In Service	http://localhost:8081/nexus/content/repositories/central
Releases	hosted	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	In Service	http://localhost:8081/nexus/content/repositories/snapshots

The bottom section is titled "Repository Information" and shows a tree view of the "Maven Central Repository Content". The tree starts with "Maven Central" and branches into "index", "ant", "classworlds", "com", "commons-beanutils", "commons-cli", and "commons-digester".

Managing Nexus



- ▶ The Nexus Repositories
 - ▶ Nexus handles several types of repository
 - ▶ Proxy repositories
 - ▶ Hosted repositories
 - ▶ Virtual repositories

Managing Nexus



- ▶ The Nexus Repositories – Proxy repositories
 - ▶ Proxies for public repositories on the internet
 - ▶ Central Maven repository
 - ▶ Apache snapshots
 - ▶ Codehaus snapshots

Managing Nexus



- ▶ The Nexus Repositories – Hosted repositories
 - ▶ JARs hosted on the Nexus server
 - ▶ Used for internal or 3rd party JARs, eg Oracle JDBC libraries
 - ▶ 3rd party
 - ▶ Snapshots
 - ▶ Releases

Managing Nexus



- ▶ The Nexus Repositories – Virtual repositories
- ▶ Used to connect to old Maven I repositories

Managing Nexus



- ▶ The Nexus Repositories – uploading an artifact
 - ▶ Manually upload a third-party library
 - ▶ Some common libraries cannot be listed on the public repositories due to licensing issues:
 - ▶ JTA
 - ▶ JMS
 - ▶ Proprietary JDBC drivers (Oracle...)
 - ▶ Allow vendors to deliver their libraries into your repository

Managing Nexus



► The Nexus Repositories – uploading an artifact

The screenshot shows the Sonatype Nexus Maven Repository Manager interface. The title bar reads "Sonatype Nexus Maven Repository Manager" and the address bar shows "http://192.168.1.199:8081/nexus/index.html". The top right corner displays "admin | Log Out" and "Version 1.2.1 Open Source".

The left sidebar contains several sections: "Sonatype™ Servers" (Nexus selected), "Artifact Search" (with "Advanced Search" link), "Views" (including "Browse Repositories", "System Feeds", and "Logs and Config Files"), "Administration" (with "Server", "Repositories", "Groups", "Routing", "Scheduled Tasks", and "Repository Targets" links), and "Security" (with "Change Password", "Users", "Roles", and "Privileges" links). The main content area is titled "Welcome" and "Repositories". It lists "User Managed Repositories" under "Repository Path":

Repository	Type	Status	Repository Path
Public Snapshot Repositories	group		http://192.168.1.199:8081/nexus/content/groups/public-snapshots
Public Repositories	group		http://192.168.1.199:8081/nexus/content/groups/public
3rd party			http://192.168.1.199:8081/nexus/content/repositories/thirdparty
Apache Snapshots			http://192.168.1.199:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow			http://192.168.1.199:8081/nexus/content/repositories/central-m1
Codehaus Snapshots			http://192.168.1.199:8081/nexus/content/repositories/codehaus-snap...
Effacy Development Maven			http://192.168.1.199:8081/nexus/content/repositories/effacy-repo

A context menu is open over the "3rd party" repository entry, listing options: "View", "Put Out of Service", "Expire Cache", "Re-Index", "Rebuild Attributes", and "Upload Artifact...". A yellow callout bubble points to the "Upload Artifact..." option with the text "Upload an artifact".

Managing Nexus



► The Nexus Repositories – uploading an artifact

The screenshot shows the Nexus Repository Manager interface. The top navigation bar has tabs for 'Welcome' and 'Repositories'. The 'Repositories' tab is selected, showing a list of repositories:

Repository	Type	Status	Repository Path
Public Snapshot Repositories	group	In Service	http://192.168.1.199:8081/nexus/content/groups/public-snapshots
Public Repositories	group	In Service	http://192.168.1.199:8081/nexus/content/groups/public
3rd party	hosted	In Service	http://192.168.1.199:8081/nexus/content/repositories/3rd-party
Apache Snapshots	proxy	In Service	http://192.168.1.199:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	In Service	http://192.168.1.199:8081/nexus/content/repositories/central-m1

Repository Content

Artifact Upload to 3rd party

Select a File For Upload: /Users/johnsmart/Desktop/jta-1_0_1B-classes.zip

Specify Artifact Information

POM File Attributes

Auto Group: javax.transaction

Group	javax.transaction
Artifact	jta
Version	1.0.1B
Packaging	jar
Classifier	
Extension	

Upload Cancel

Three yellow callout boxes provide instructions:

- Select the file to be uploaded
- If a pom.xml file is available, you're in luck!
- You usually have to provide the details yourself

Managing Nexus



► The Nexus Repositories – uploading an artifact

The screenshot shows the Nexus Repository Manager interface. The top navigation bar includes 'Welcome' and 'Repositories'. The 'Repositories' tab is active, displaying a table of managed repositories:

Repository	Type	Status	Repository Path
Public Snapshot Repositories	group	In Service	http://192.168.1.199:8081/nexus/content/groups/public-snapshots
Public Repositories	group	In Service	http://192.168.1.199:8081/nexus/content/groups/public
3rd party	hosted	In Service	http://192.168.1.199:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	In Service	http://192.168.1.199:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	In Service	http://192.168.1.199:8081/nexus/content/repositories/central-m1

The 'Repository Content' section shows the structure of the '3rd party' repository. A yellow callout points from the text 'Now the artifact is available from the repository' to the 'jta-1.0.1B' folder, which contains files like 'jta-1.0.1B.jar', 'jta-1.0.1B.jar.md5', etc.

Now the artifact is available from the repository

Managing Nexus



► The Nexus Repositories – adding a new repository

Sonatype Nexus™

Sonatype™ Servers

Nexus

Artifact Search

Advanced Search

Views

Browse Repositories

System Feeds

Logs and Config Files

Administration

- Server
- Repositories
- Groups
- Routing
- Scheduled Tasks
- Repository Targets

Security

Welcome Repository Configuration

Repositories

Add a new proxy repository

Repository	Hosted	Type	Policy	Repository Path
3rd party	Proxy	hosted	release	http://192.168.1.199:8081/nexus/content/repositories/thirdparty
Apache Sna	Virtual	proxy	snapshot	http://192.168.1.199:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow		virtual		http://192.168.1.199:8081/nexus/content/repositories/central-m1
Codehaus Snapshots		proxy	snapshot	http://192.168.1.199:8081/nexus/content/repositories/codehaus-snapshots
Effacy Development Maven 2 ...		proxy	release	http://192.168.1.199:8081/nexus/content/repositories/effacy-repo
Maven Central		proxy	release	http://192.168.1.199:8081/nexus/content/repositories/central
Openqa Release Repository		proxy	release	http://192.168.1.199:8081/nexus/content/repositories/openqa-releases
Releases		hosted	release	http://192.168.1.199:8081/nexus/content/repositories/releases
Snapshots		hosted	snapshot	http://192.168.1.199:8081/nexus/content/repositories/snapshots

Managing Nexus



► The Nexus Repositories – adding a new repository

Screenshot of the Nexus Repository Manager interface showing the 'Add...' dialog for creating a new repository.

The 'Repository ID' field contains `maven2-repository.dev.java.net`. A yellow callout points to this field with the text: "A machine-friendly name for this repository".

The 'Repository Name' field contains `Java.net Repository for Maven`. A yellow callout points to this field with the text: "Human-friendly repository name".

Other configuration fields shown include:

- Repository Type: proxy
- Format: maven2
- Repository Policy: Release
- Default Local Storage Location (empty)
- Override Local Storage Location (empty)
- Remote Repository Access settings:
 - Remote Storage Location: `http://download.java.net/maven/2`
 - Download Remote Indexes: True
 - Checksum Policy: Warn
- Authentication (optional) checkbox (unchecked)

At the bottom right of the dialog are 'Save' and 'Cancel' buttons.

Managing Nexus



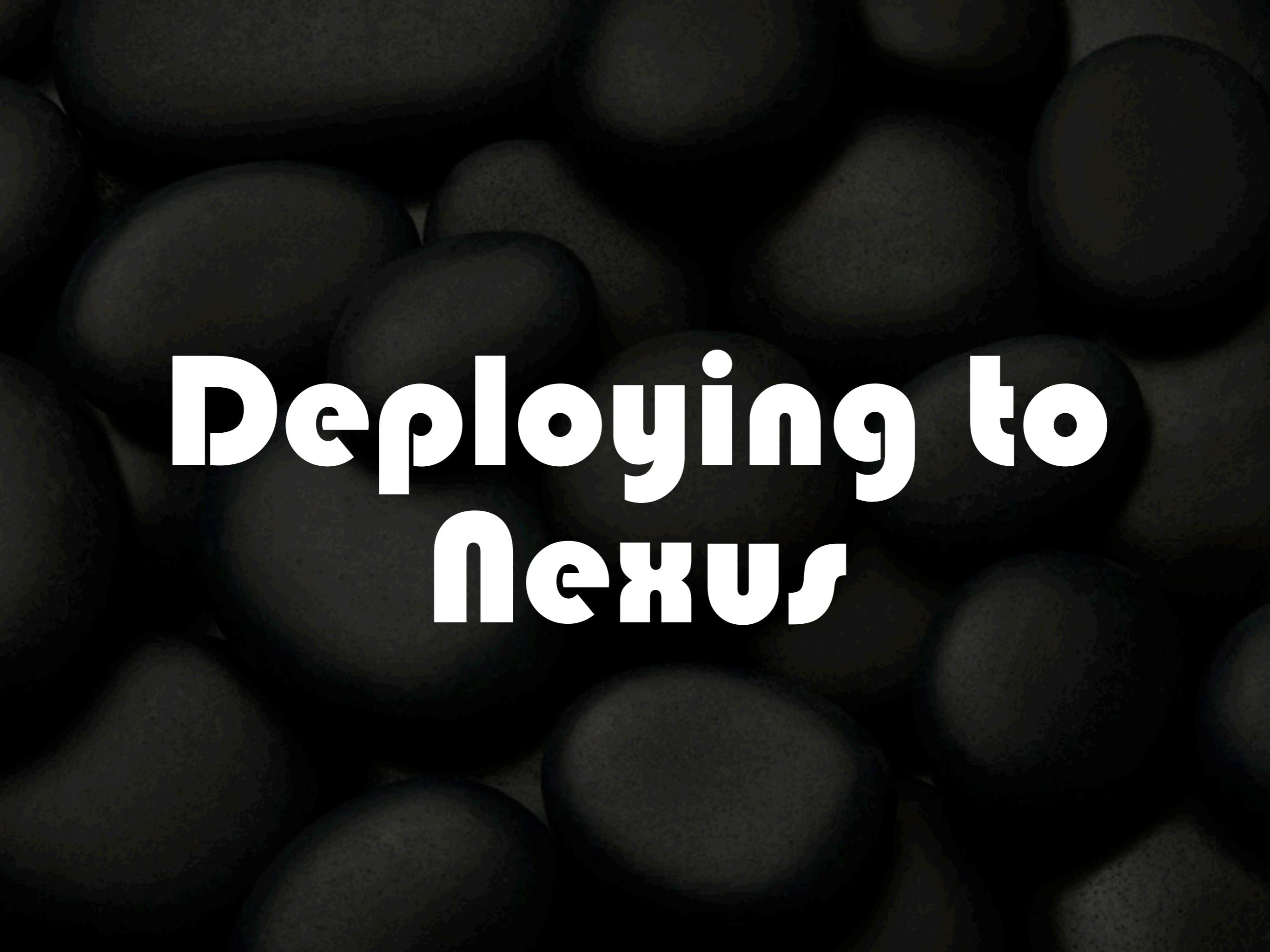
► The Nexus Repositories – adding a new repository

The screenshot shows the Sonatype Nexus interface with the 'Groups' tab selected. A yellow callout bubble points to the 'Add' button in the top left of the main content area, with the text: "Now add the repository to a public group to make it visible".

The 'Repository Group Configuration' section shows a 'Group ID' of 'public' and a 'Group Name' of 'Public Repositories'. A yellow callout bubble points to the 'Available Repositories' list, with the text: "Drag the repository into the Public Repositories group".

The 'Available Repositories' list includes: Apache Snapshots, Java.net Repository for Maven, Central M1 shadow, and Java.net Repository for Maven. A yellow callout bubble points to this list with the text: "These libraries will now be available".

At the bottom right of the configuration screen are 'Save' and 'Cancel' buttons.



Deploying to
nexus

Deploying your application



▶ The Maven deployment process

▶ Install in your local repository

```
$ mvn install
```

Available for other projects or modules on
your machine

▶ Then deploy to a remote server

```
$ mvn deploy
```

Available for other developers

Deploying your application



▶ Deploying to a Nexus repository

```
<project>
  ...
  <distributionManagement>
    <repository>
      <id>releases</id>
      <name>Internal Releases</name>
      <url>http://buildserver:8081/nexus/content/repositories/releases</url>
    </repository>
  </distributionManagement>
</project>
```

Nexus Repository Server URL

Target repository

settings.xml

```
<settings>
  ...
  <servers>
    <server>
      <id>releases</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
  ...
</settings>
```

You also need to configure an
(appropriate) username and password for
this server

Deploying your application



- ▶ Deploying to a Nexus repository
- ▶ Artifacts are deployed to the Nexus server

Sonatype Nexus

admin | Log Out | Beta 2

Sonatype Servers

Nexus

Views

- Artifact Search
- Recently Cached Artifacts
- Recently Deployed Artifacts
- System Changes

Maintenance

- Repositories
- View Server Config
- View Server Logs

Configuration

- Server
- Repositories
- Groups
- Routing

Welcome Maintenance Repositories Groups

Refresh

Repository	Type	Status	Repository Path
3rd party	hosted	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	In Service	http://localhost:8081/nexus/content/repositories/central-m1
Codehaus Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snapshot
Maven Central	proxy	In Service	http://localhost:8081/nexus/content/repositories/central
Releases	hosted	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	In Service	http://localhost:8081/nexus/content/repositories/snapshots

Repository Information

Releases Repository Content

- Releases
- .index
- com
- wakaleo
- tax
- tax-calculator-core
- 1.0
- tax-calculator-core-1.0.jar
- tax-calculator-core-1.0.pom
- maven-metadata.xml

Target repository

Artifact coordinates and deployed artifacts

Repository	Type	Status	Repository Path
3rd party	hosted	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central M1 shadow	virtual	In Service	http://localhost:8081/nexus/content/repositories/central-m1
Codehaus Snapshots	proxy	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snapshot
Maven Central	proxy	In Service	http://localhost:8081/nexus/content/repositories/central
Releases	hosted	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	In Service	http://localhost:8081/nexus/content/repositories/snapshots

labs

- ▶ Lab 14 - Using Nexus



Maven Intermediate Concepts

Part 12

Automating the Release Process



Snapshots and Releases

Release Management



- ▶ Snapshots and releases
- ▶ A SNAPSHOT represents work in progress
 - ▶ Each release produces a new time-stamped artifact
 - ▶ Snapshot versions are identified by the SNAPSHOT keyword

```
<project>
  ...
  <groupId>com.sonatype.training</groupId>
  <artifactId>babble</artifactId>
  <packaging>pom</packaging>
  <version>1.0-SNAPSHOT</version>
  ...

```

A SNAPSHOT version

Release Management



- ▶ Snapshots and releases
- ▶ A RELEASE represents a finished product
 - ▶ A stable, tested and officially released artifact
 - ▶ A released artifact is unique

```
<project>
  ...
  <groupId>com.sonatype.training</groupId>
  <artifactId>babble</artifactId>
  <packaging>pom</packaging>
  <version>1.0</version>
  ...
  ...
```

A RELEASE version

Release Management



- ▶ Deploying a snapshot
- ▶ Snapshots can only be deployed to a Snapshot repository
- ▶ Needs to be configured in the `<distributionManagement>` section

```
<project>
  ...
  <distributionManagement>
    <snapshotRepository>
      <id>local-snapshots</id>
      <name>Internal Snapshots</name>
      <url>http://localhost:8081/nexus/content/repositories/snapshots</url>
    </snapshotRepository>
  ...
</distributionManagement>
...
```

Define where snapshots should go

settings.xml

```
<settings>
  ...
  <servers>
    <server>
      <id>local-snapshots</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
  ...
</settings>
```

And configure an (appropriate) username and password for this server

Release Management



- ▶ Deploying a snapshot
- ▶ Now use **mvn deploy** to deploy your SNAPSHOT

```
$ mvn deploy
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building babble-core
[INFO]   task-segment: [deploy]
[INFO] -----
...
Uploading: http://localhost:8081/nexus/content/repositories/snapshots/com/sonatype/training/
babble-core/1.0-SNAPSHOT/babble-core-1.0-20090330.052432-1.jar
3K uploaded  (babble-core-1.0-20090330.052432-1.jar)
...
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
```

Uploading snapshot to Nexus

Release Management



▶ Deploying a snapshot

The screenshot shows the Sonatype Nexus Maven Repository Manager interface. The left sidebar includes 'Sonatype™ Servers' (Nexus selected), 'Artifact Search' (Advanced Search button), 'Views/Repositories' (Repositories tab selected), 'Repositories' (System Feeds), and 'Help'. The main area has tabs for 'Welcome' and 'Repositories'. The 'Repositories' tab displays a table of repositories:

Repository	Type	Format	Policy	Repository Status	Repository Path
Public Repositories	group	maven2		Releases, Snapshots, 3rd party, Ma...	http://localhost:8081/nexus/content/groups/public
Public Snapshot Repositories	group	maven2		Apache Snapshots, Codehaus Snap...	http://localhost:8081/nexus/content/groups/public-snapshots
3rd party	hosted	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapsh...
Central M1 shadow	virtual	maven1		In Service	http://localhost:8081/nexus/content/repositories/central-m1
Codehaus Snapshots	proxy	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snap...
Maven Central	proxy	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/central
Releases	hosted	maven2	release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	maven2	snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots

The 'Snapshots' section shows a tree view of artifacts under 'Schemas'. A yellow callout bubble points to the 'babble-core' folder in the '1.0-SNAPSHOT' directory, containing files like 'babble-core-1.0-20090330.052432-1.jar', 'babble-core-1.0-20090330.052432-1.jar.md5', etc. The callout bubble contains the text: "A new timestamped artifact is now on Nexus".

Automating Releases

Release Management



- ▶ Automating releases
- ▶ Deploying a release involves a number of standard steps:
 - ▶ Commit all current changes to Subversion
 - ▶ Update the version number (release version) and commit the changes
 - ▶ Tag the release version
 - ▶ Build and deploy this version
 - ▶ Update the version number (development version) and commit the changes

Release Management



- ▶ Use the maven-release-plugin to automate these steps
- ▶ Uses the SCM block in your pom.xml file

```
<scm>
  <connection>scm:svn:https://myserver/.../trunk/babble</connection>
  <developerConnection>scm:svn:https://myserver/...trunk/babble</developerConnection>
</scm>
```

- ▶ Also works fine with other version control systems

```
<scm>
  <connection>scm:git:git://github.com/someproj/babble.git</connection>
  <developerConnection>scm:git:git://github.com/someproj/babble.git</developerConnection>
</scm>
```

Using git

- ▶ Easy to configure

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-release-plugin</artifactId>
  <configuration>
    <tagBase>https://myserver/.../releases/babble</tagBase>
    <username>${scm.username}</username>
    <password>${scm.password}</password>
  </configuration>
</plugin>
```

(Optional) Releases go here

Authentication (property values go in settings.xml)

Release Management



- ▶ Use the maven-release-plugin with multi-module projects
- ▶ Sometimes needs a little extra configuration

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-release-plugin</artifactId>
  <configuration>
    <preparationGoals>clean verify install</preparationGoals>
  </configuration>
</plugin>
```

Multi-module projects may need this option

Release Management



▶ Preparing a release

```
$ mvn release:prepare
```

▶ Maven will:

- ▶ Check that there is no uncommitted code
- ▶ Check that there are no snapshot dependencies
- ▶ Ask you to confirm the release version, release tag, and new development version

Release Management



▶ Preparing a release - release:prepare in action

```
$ mvn release:prepare
[INFO] Scanning for projects...
[INFO] Reactor build order:
[INFO]   babble
[INFO]   babble-core
[INFO]   babble-services
[INFO]   babble web application
[INFO] -----
[INFO] Building babble
[INFO]   task-segment: [release:prepare] (aggregator-style)
[INFO] -----
[INFO] [release:prepare]
[INFO] Resuming release from phase 'map-release-versions'
What is the release version for "babble"? (com.sonatype.training:babble) 1.0.5: :
What is the release version for "babble-core"? (com.sonatype.training:babble-core)
1.0.5: :
What is the release version for "babble-services"?
(com.sonatype.training.babble:babble-services) 1.0.5: :
What is the release version for "babble web application"?
(com.sonatype.training:babble-web) 1.0.5: :
What is SCM release tag or label for "babble"? (com.sonatype.training:babble)
babble-1.0.5: :
What is the new development version for "babble"? (com.sonatype.training:babble)
1.0.6-SNAPSHOT: :
What is the new development version for "babble-core"?
(com.sonatype.training:babble-core) 1.0.6-SNAPSHOT: :
What is the new development version for "babble-services"?
(com.sonatype.training.babble:babble-services) 1.0.6-SNAPSHOT: :
What is the new development version for "babble web application"?
(com.sonatype.training:babble-web) 1.0.6-SNAPSHOT: :
...
[INFO] Working directory: /Users/someuser/labs/labs-git
[INFO] Executing: /bin/sh -c cd /Users/someuser/labs/labs-git && git push
[INFO] Working directory: /Users/someuser/labs/labs-git
[INFO] Release preparation complete.
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
```

Confirm the release version numbers

Confirm the release tag

Confirm the new development version numbers

Update the SCM

Release Management



▶ Preparing a release

```
$ mvn release:prepare
```

▶ Maven will also:

- ▶ Update the SNAPSHOT version number to a release version number
- ▶ Perform a build to verify everything is ok
- ▶ Commit the modified POMs to the SCM
- ▶ Create a tag in the SCM for this release
- ▶ Update the POM version number to the next SNAPSHOT version
- ▶ Commit the modified POMs to the SCM

Release Management



▶ Preparing a release in batch mode

```
$ mvn release:prepare -B
```

- ▶ Maven will use the default version numbers
- ▶ Useful for a Continuous Integration environment

Release Management



▶ Deploying the release

```
$ mvn release:perform
```

- ▶ Maven will
 - ▶ Checkout the release we just tagged
 - ▶ Build the application
 - ▶ Deploy the application

labs

- ▶ Lab 15 - Automating Releases



Maven Intermediate Concepts

Part 13

*Continuous Integration with
Hudson*

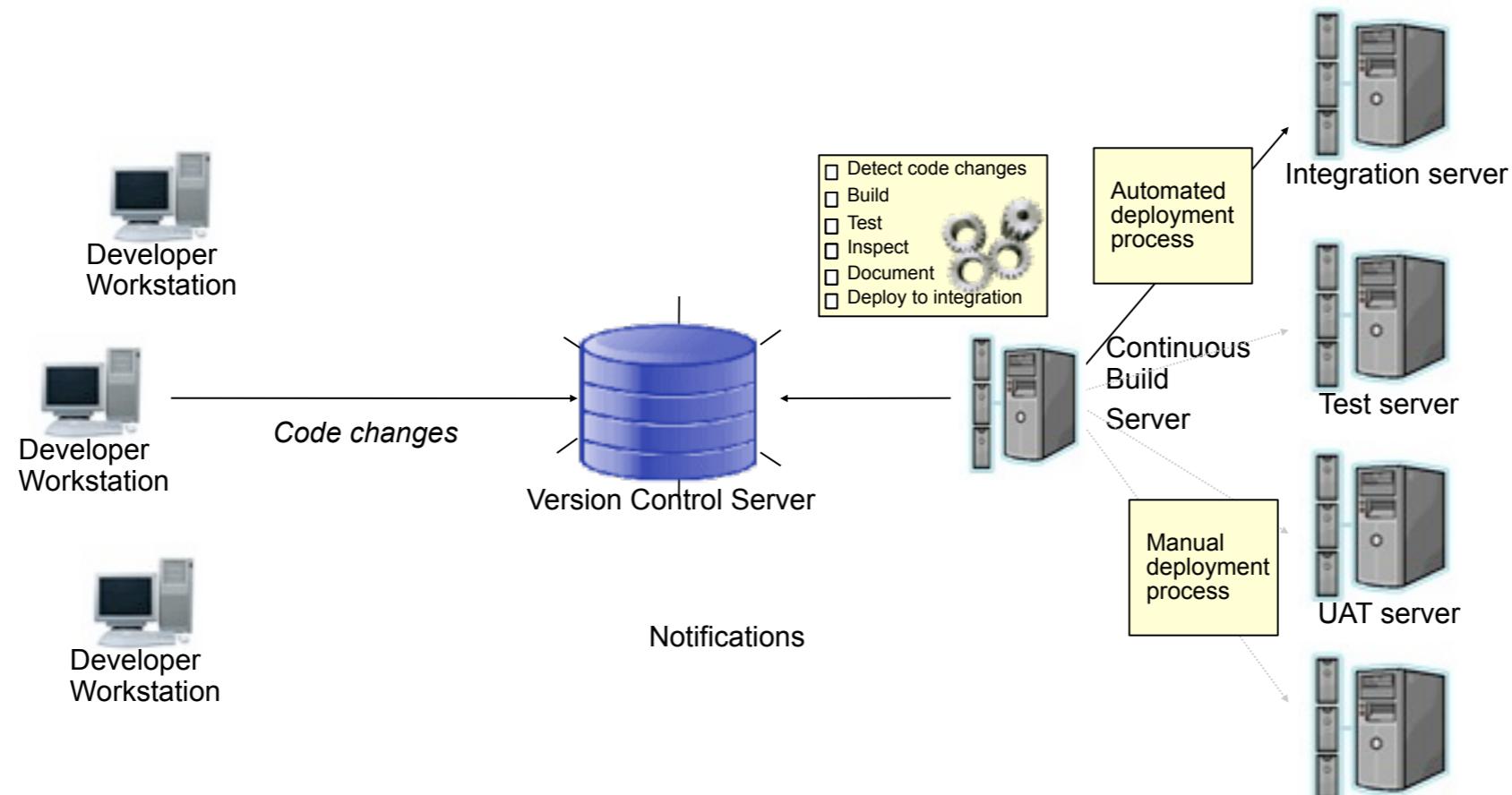


Continuous Integration

Continuous Integration



▶ What is Continuous Integration?



Continuous Integration



- ▶ Benefits of Continuous Integration?
- ▶ Continuous Integration – an industry best practice
 - ▶ Smoother integration
 - ▶ Automatic regression testing
 - ▶ Regular working releases
 - ▶ Earlier functional testing
 - ▶ Faster and easier bug fixes
 - ▶ Better visibility

A dark, moody background featuring a person's face in profile and a hand holding a cigarette.

Hudson

Continuous Integration



- ▶ Introducing Hudson
- ▶ An open source Continuous Integration server
 - ▶ Easy to install
 - ▶ Easy to configure
 - ▶ Stores artifacts
 - ▶ Nice reporting features
 - ▶ Distributed builds
 - ▶ Lots of plugins

Continuous Integration



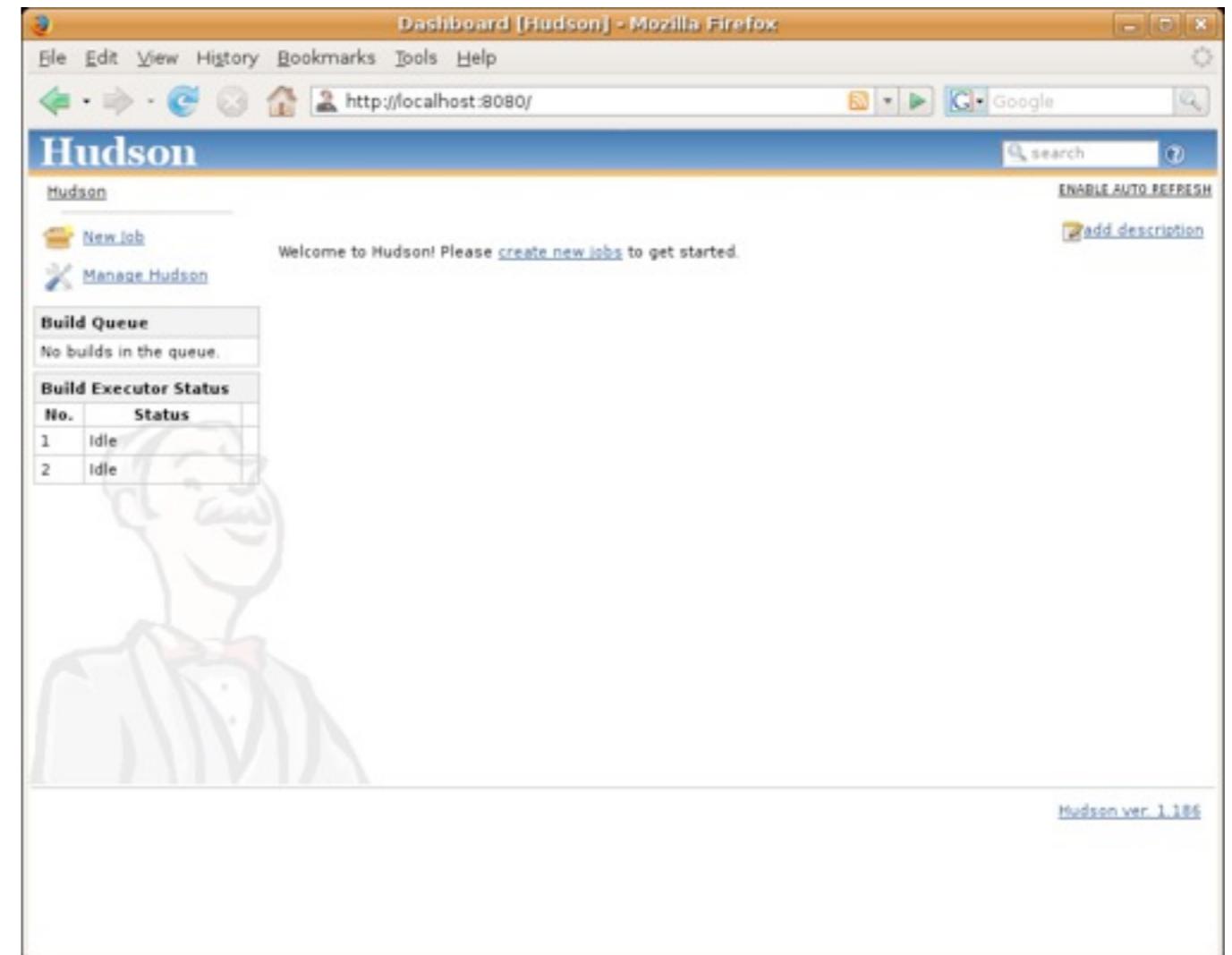
- ▶ Installing Hudson
 - ▶ Download the hudson.war file from the website
 - ▶ Two installation options
 - ▶ As a stand-alone application
 - ▶ Running in a servlet container

Continuous Integration



- ▶ Running Hudson as a stand-alone application
 - ▶ Run from the command line
 - ▶ Uses an embedded web server

```
$ java -jar hudson.war
```





Hudson &
maven

Continuous Integration



- ▶ Configuring Hudson for Maven
 - ▶ Hudson configuration is done in the *Manage Hudson* screen
 - ▶ To set up Java and Maven, go to the *Configure System* screen

The image displays two screenshots of the Hudson web interface. The left screenshot shows the 'Manage Hudson' screen, which includes a sidebar with links like 'New Job', 'Manage Hudson' (which is highlighted), 'Build History', and 'Claim Report'. The main area lists various configuration options such as 'Reload Configuration from Disk', 'Manage Plugins', 'System Information', 'System Log', 'Load Statistics', 'Script Console', 'Manage Nodes', 'Disk usage', 'Manage Users', and 'Prepare for Shutdown'. The right screenshot shows the 'Configure System' screen, which allows users to set up global settings and paths. It includes fields for 'Home directory' (set to '/home/hudson/.hudson'), 'System Message' (containing '<h3>Welcome to the Wakaleo Consulting Hudson Build Server</h3>'), '# of executors' (set to 2), 'Quiet period' (set to 5), 'Enable security' (checked), 'TCP port for JNLP slave agents' (radio buttons for 'Fixed', 'Random', and 'Disable' with 'Random' selected), 'Access Control', 'Security Realm' (radio buttons for 'Hudson's own user database' (selected), 'Allow users to sign up', 'LDAP', 'Delegate to servlet container', and 'Unix user/group database'), and 'Authorization' (radio buttons for 'Matrix-based security', 'Logged-in users can do anything' (selected), 'Anyone can do anything', and 'Project-based Matrix Authorization Strategy').

Continuous Integration



▶ Configuring Hudson for Maven

▶ The main things to configure are:

- ▶ JDK installation(s)
- ▶ Maven installation(s)
- ▶ Mail server configuration

JDKs

JDK installations	name	Java 1.5.0
	JAVA_HOME	/usr/lib/jvm/java-1.5.0-sun

Maven

Maven installation	name	Maven 2.1.0
	MAVEN_HOME	/usr/local/maven

E-mail Notification

Test configuration by sending e-mail to System Admin Address [Test configuration](#)

SMTP server	localhost
Default user e-mail suffix	
System Admin E-mail Address	hudson@taronga
Hudson URL	http://192.168.1.199:8082/

[Advanced...](#)

Continuous Integration



- ▶ Hudson build jobs
 - ▶ A job is a particular execution of an automated script
 - ▶ A project may require several “jobs”

Continuous Integration



- ▶ Setting up a Maven build job in Hudson
- ▶ Hudson has several types of build jobs

The screenshot shows the Hudson dashboard in a Mozilla Firefox browser. On the left, there's a sidebar with links for 'New Job' (highlighted with a yellow oval), 'Manage Hudson', 'Build Queue' (empty), and 'Build Executor Status' (two idle entries). The main content area is titled 'Hudson' and shows a 'New Job' creation form. It includes fields for 'Job name' (with a placeholder 'Job name...'), a radio button for 'Build a maven2 project' (selected), and a detailed description of what it does. Below that are options for 'Monitor an external job', 'Build a free-style software project', 'Build multi-configuration project (alpha)', and 'Copy existing job'. A yellow speech bubble points to the 'Build a maven2 project' option with the text 'Maven 2 projects'. Another yellow speech bubble points to the 'Monitor an external job' section with the text 'Monitoring external jobs'. A third yellow speech bubble points to the bottom section with the text 'General purpose jobs'.

Maven 2 projects

Monitoring external jobs

General purpose jobs

Continuous Integration



- ▶ Setting up a Maven build job in Hudson
 - ▶ Freestyle jobs
 - ▶ Build any type of projects (Ant, Maven, Makefile, Shell script...)
 - ▶ Work with all the reporting plugins
 - ▶ Require the most work to configure
 - ▶ Maven 2 jobs
 - ▶ Hudson reads the pom.xml file to make setting up the project easier.
 - ▶ Build projects in the correct order based on the Maven dependencies
 - ▶ Less reporting plugins are available than with free-style projects

Continuous Integration



- ▶ Setting up a Maven build job in Hudson
- ▶ Creating a new Maven build job

Hudson

john | log out

New Job

Manage Hudson

People

Build History

Claim Report

Build Queue
No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Job name: tax-calculator-core

Build a maven2 project

Build a maven2 project. Hudson takes advantage of your POM files and drastically reduces the configuration.

Monitor an external job

This type of job allows you to record the execution of a process run outside Hudson, even on a remote machine. This is designed so that you can use Hudson as a dashboard of your existing automation system. See [the documentation for more details](#).

Build a free-style software project

This is the central feature of Hudson. Hudson will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build multi-configuration project (alpha)

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Copy existing job

Copy from:

OK

A yellow callout bubble points to the "Build a maven2 project" radio button, highlighting it.

Continuous Integration



- ▶ Setting up a Maven build job in Hudson
- ▶ Enter SCM details - where does the code come from?

Hudson

Hudson > tax-calculator-core

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure Modules

Project name: tax-calculator-core

Description:

Discard Old Builds
 This build is parameterized
 Disable Build (No new builds will be executed until the project is re-enabled.)
 Tie this project to a node

Node: master (the master Hudson node)

Advanced Project Options

Source Code Management

None
 CVS
 Subversion

Modules

Repository URL: `guard.com/svn/jpt-bootcamp/tax-calculator/trunk/tax-calculator-core/`

Local module directory (optional):

Add more locations...

Use update:
If checked, Hudson will use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser: (Auto)

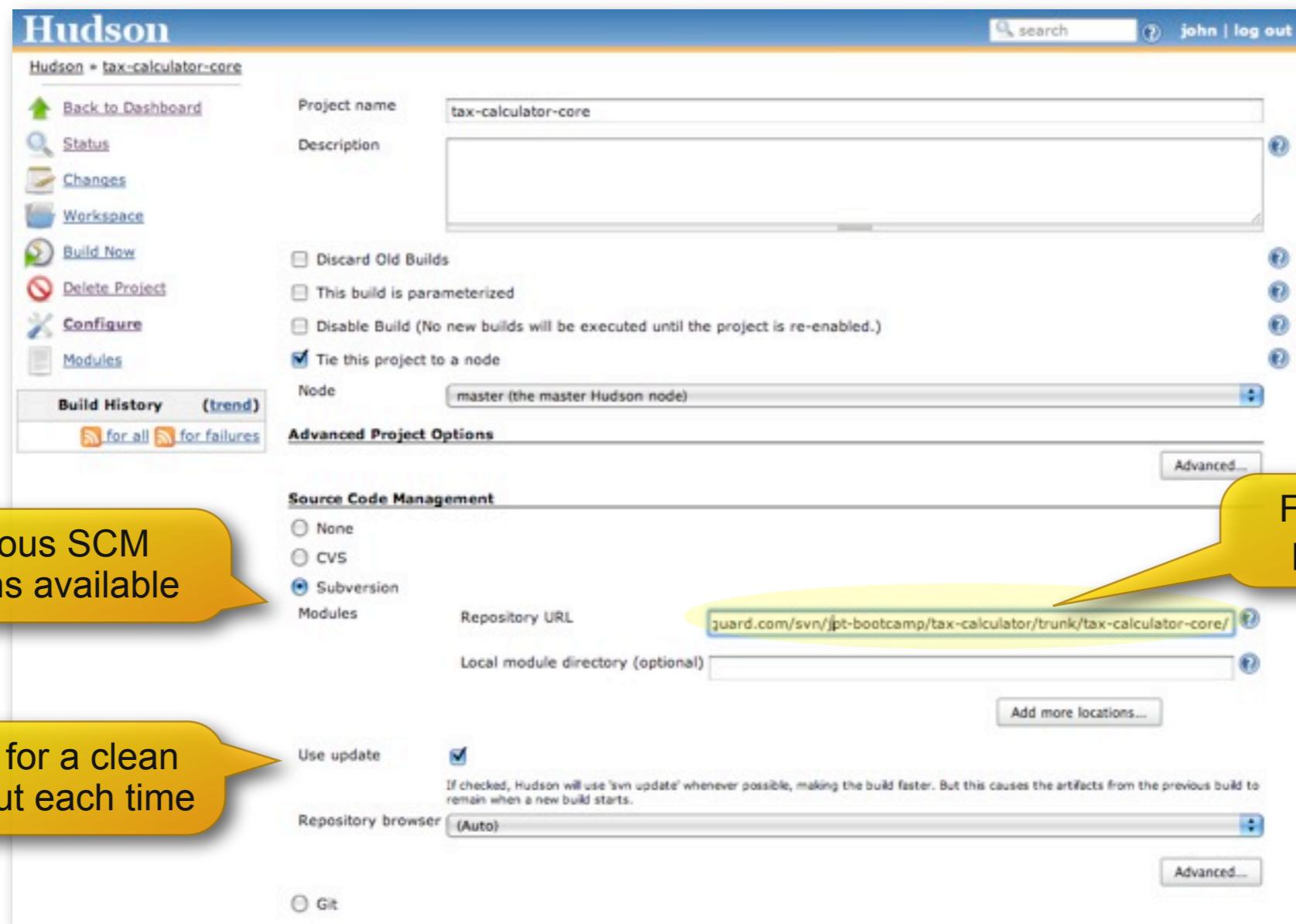
Advanced...

Build History (trend)
for all for failures

Various SCM options available

Untick for a clean checkout each time

For Subversion, provide a URL



Continuous Integration



- ▶ Setting up a Maven build job in Hudson
- ▶ Enter build triggers - when does the build kick off?

The screenshot shows the configuration page for a Hudson job. It includes sections for Build Triggers, Schedule, Build, and Build Settings.

- Build Triggers:** Includes options like "Build whenever a SNAPSHOT dependency is built" (checked) and "Poll SCM". A yellow callout points to "Poll SCM" with the text "Periodic (e.g. nightly) builds". Another yellow callout points to "Build whenever a SNAPSHOT dependency is built" with the text "If there are SNAPSHOT dependencies, Hudson will rebuild if any of these are updated".
- Schedule:** Set to */5 * * * * (every 5 minutes). A yellow callout points to this field with the text "Check Subversion for changes every 5 minutes".
- Build:** Root POM is pom.xml, Goals and options are install. A yellow callout points to the "Goals and options" field with the text "What Maven goal to run".
- Build Settings:** E-mail Notification is checked, with recipients john tim jason. A yellow callout points to the "Recipients" field with the text "Who gets notified if the build fails". Other settings include "Send e-mail for every unstable build" (unchecked) and "Send separate e-mails to individuals who broke the build" (checked).

Continuous Integration



- ▶ Monitoring your builds
- ▶ The Hudson dashboard

Hudson

search john | log out

DISABLE AUTO REFRESH

New Job Manage Hudson People Build History Project Relationship Check File Fingerprint Claim Report

Welcome to the Wakaleo Consulting Hudson Build Server

Build status Build stability edit description

All	S	W	Job	Last Success	Last Failure	Last Duration
			bnbglobal-core	2 hr 10 min (#13)	2 days 14 hr (#12)	55 sec
			bnbglobal-core-metrics	2 hr 9 min (#12)	5 days 6 hr (#10)	6 min 15 sec
			bnbglobal-i10n	2 days 14 hr (#12)	4 days 3 hr (#11)	2 min 2 sec
			bnbglobal-i10n-metrics	2 days 14 hr (#13)	1 mo 10 days (#6)	5 min 55 sec
			bnbglobal-parent	2 hr 15 min (#19)	4 days 4 hr (#13)	23 sec
			tax-calculator	N/A	N/A	N/A

Icon: S M L

Legend: for all for failures for just latest builds

Build in progress Force build

The Hudson dashboard displays the build status and stability of various projects. A yellow callout points to the 'Build status' section, which includes icons for success (green), warning (yellow), and failure (red). Another yellow callout points to a 'Build in progress' entry for the 'tax-calculator' project, which has a blue progress bar. A third yellow callout points to a 'Force build' button at the bottom right of the dashboard.

#	Status
1	Building tax-calculator #2
2	Idle

Continuous Integration



- ▶ Extending Hudson
- ▶ Hudson plugins

The screenshot shows the Hudson Update Center interface. The title bar says "Update Center [Hudson]". Below it, the address bar shows "http://localhost:8080/pluginManager/available". The main area is titled "Hudson > Plugin Manager" and has tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar is at the top right. The "Available" tab displays a table of plugins:

Install	Name	Version
<input type="checkbox"/>	Accurev Plugin	0.6.7
<input type="checkbox"/>	Active Directory plugin	1.11
<input type="checkbox"/>	Audit Trail Plugin	1.3
<input type="checkbox"/>	Batch Task Plugin	1.7
<input type="checkbox"/>	BitKeeper Plugin	1.4
<input type="checkbox"/>	Bugzilla Plugin	1.3
<input type="checkbox"/>	Build Publisher Plugin	1.4
<input type="checkbox"/>	Build-timeout Plugin	1.4
<input type="checkbox"/>	Change Log History Plugin	1.0
<input type="checkbox"/>	Checkstyle Plugin	2.8
<input type="checkbox"/>	The Continuous Integration Game plugin	1.7
<input type="checkbox"/>	Claim plugin	1.4
<input type="checkbox"/>	ClearCase Plugin	0.8.1
<input type="checkbox"/>	Clover Plugin	1.7
<input type="checkbox"/>

Lots of plugins available

Version Control

Build tools

Issue tracking integration

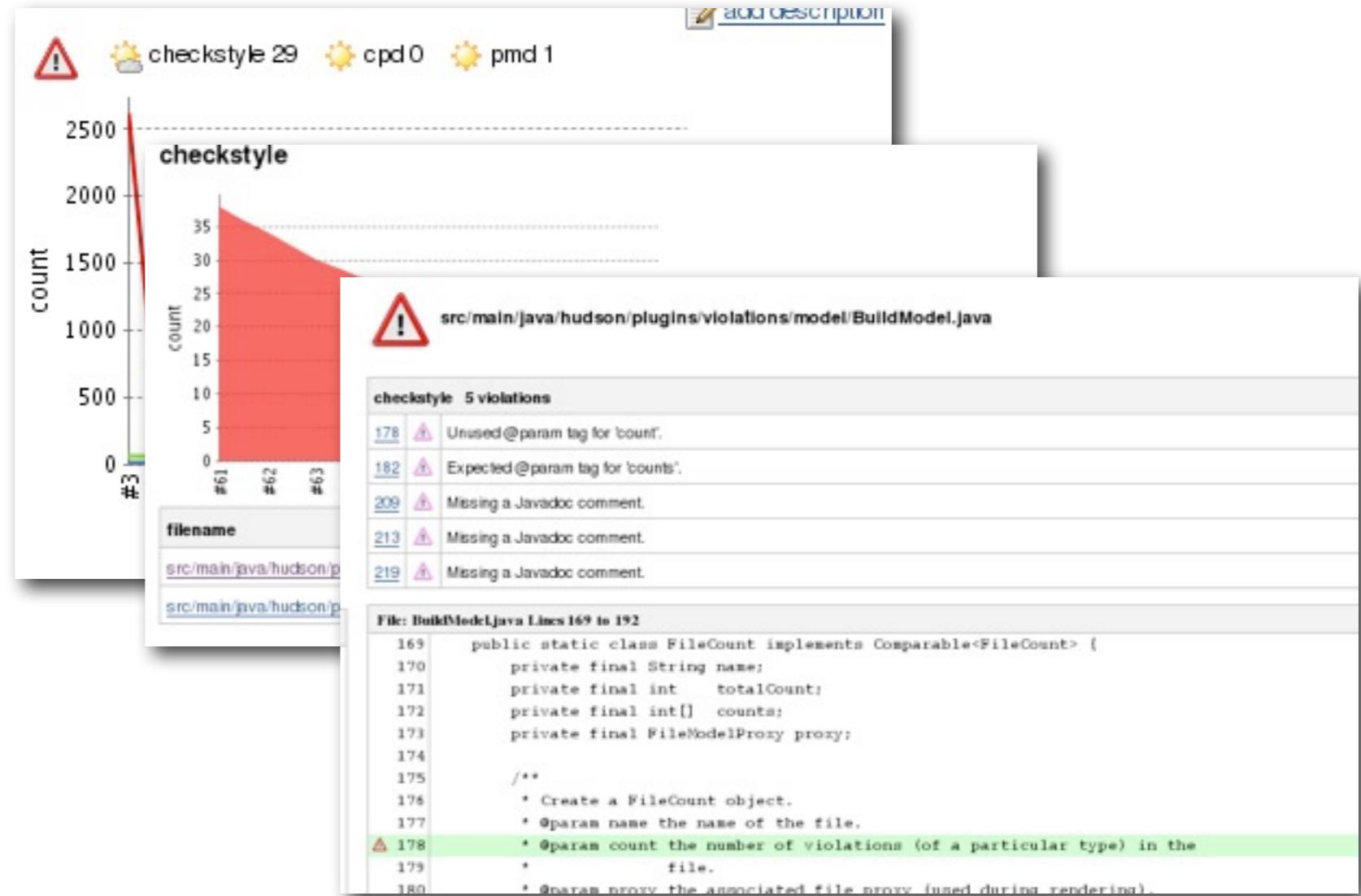
Reporting

...

Continuous Integration



- ▶ Reporting with Hudson
- ▶ Many reporting plugins for Hudson
 - ▶ Violations (checkstyle, PMD, Findbugs)
 - ▶ Cobertura
 - ▶ Clover
 - ▶ Emma
 - ▶ ...
- ▶ Easy to integrate
- ▶ Nice graphs



Demo

- ▶ A Quick Demonstration of Hudson
 - ▶ <http://grid.sonatype.org/ci>
- ▶ Builds the Entire Maven Ecosystem
 - ▶ Maven
 - ▶ Nexus
 - ▶ m2eclipse
 - ▶ Maven Plugins
 - ▶ Development Builds (Maven 3)



labs

- ▶ Lab 16 - Using Hudson





Bamboo &
maven

Bamboo



- ▶ Commercial CI server from Atlassian
- ▶ Strong Maven integration
- ▶ Simple for multi-module builds
- ▶ Strong support for multi-branch builds

**Build Resources**[Agents](#)[Agent Matrix](#)[Builders](#)[JDKs](#)[Server Capabilities](#)[Global Variables](#)**Elastic Bamboo**[Configuration](#)**Plans**[Import With Maven](#)[Build Expiry](#)[Bulk Action](#)[Build Monitoring](#)[Remove Plans](#)[Move Plans](#)[Bulk Edit Plan](#)[Permissions](#)**Security**[Users](#)[Groups](#)[Security Settings](#)[Global Permissions](#)**Communication**[Mail Server](#)[IM Server](#)[JIRA Server](#)**Gadgets**

Import With Maven

You can import a plan from a specified pom.xml using Maven.

Enter pom.xml Details

SVN URL: *The location of your pom.xml in your subversion repository (e.g. http://svn.collab.net/repos/svn/trunk/pom.xml)**Username:**(Optional) The subversion username (if any) required to access the Repository**Authentication Type:****Password:**(Optional) The password required by the subversion username



Commons Collections - Commons Collections 3.3: Live Activity Logs

[Summary](#) [Activity](#) [Completed Builds](#) [Tests](#) [Files](#) [Maven 2](#) [Configuration](#)[Plan Actions](#)

Build Number: COMCOL-COMMONSCOLLECTIONS33-1
(currently building)

Building Started: 21 Mar 2010, 1:17:09 PM

Building on Agent: Default Agent

Changes

This build was manually triggered by [Matthew McCullough](#).

Currently Building

Commons Collections - Commons Collections 3.3 Building for 7 seconds.

21-Mar-2010 13:17:09 Executing build COMCOL-COMMONSCOLLECTIONS33-1
21-Mar-2010 13:17:09 Running pre-build action: VCS Version Collector
21-Mar-2010 13:17:09 Running pre-build action: Build Number Stamper
21-Mar-2010 13:17:09 Building started with Maven2Builder
21-Mar-2010 13:17:09

```
Starting to build 'Commons Collections - Commons Collections 3.3'  
... running command line: /Applications/Dev/apache-maven/bin/mvn -f pom.xml clean test  
... in : /Applications/Dev/bamboo-home/xml-data/build-dir/COMCOL-COMMONSCOLLECTIONS33  
... using java.home: /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
```

21-Mar-2010 13:17:09 Using Java version: 1.6
21-Mar-2010 13:17:12 [INFO] Scanning for projects...
21-Mar-2010 13:17:12 Downloading: http://localhost:8081/nexus/content/groups/public/org/apache/commons/commons-parent/14/commons-parent-14.pom

This page shows 100 activity log entries for the Commons Collections - Commons Collections 3.3 build. Refresh: every second



Build Result COMCOL-COMMONSCOLLECTIONS33-1

Labels: NONE [Add](#)

1

[Summary](#)[Tests](#)[Changes](#)[Artifacts](#)[Logs](#)[Comments](#)[Metadata](#)[Actions](#)

Build COMCOL-COMMONSCOLLECTIONS33-1 was successful.

Build completed on 21 Mar 2010, 1:19:59 PM - 42 minutes ago

Build took 2 minutes

Built on agent [Default Agent](#)

Revision is 925866

Code Changes

This build was manually triggered by [Matthew McCullough](#).

Tests

[Collapse ↑](#)

4338 tests in total

Summary

Summary



- ▶ Properties
- ▶ Filters
- ▶ Profiles
- ▶ Dependency Management
- ▶ Maven Project Sites
- ▶ Metrics
- ▶ Web Projects
- ▶ Repository Managers
- ▶ Continuous Integration

multiple Artifacts per POM



- ▶ Producing multiple artifacts from a Maven build
- ▶ Not a recommended practice
- ▶ Still feasible
- ▶ Sonatype blog post
 - ▶ <http://www.sonatype.com/people/2010/01/how-to-create-two-jars-from-one-project-and-why-you-shouldnt/>



**Encrypted
Passwords**

Encrypting Passwords



- ▶ Symmetric encryption
- ▶ Master Password
 - ▶ mvn -emp <masterpassword>
 - ▶ Stored on local disk
 - ▶ Or on removable USB stick
- ▶ Encrypting connectivity passwords
 - ▶ mvn -ep <connectionpassword>
 - ▶ Copy and paste into settings.xml
 - ▶ Wrapped with curly braces

```
[~]> mvn -ep mysecretpasswordplaintext
Using Java version: 1.6
{6H5YZrmaTP4HE3s1Hq7jdFhsAttIm2HfzTJtwht0gWDjrBfo0gud/9ANV2s1lxY9}
[~]>
```



war Overlays

War Overlays



- ▶ Normally all artifact dependencies are collected in WEB-INF/lib except for war artifacts.
- ▶ Instead, war artifacts are overlayed on the war source.
- ▶ No configuration is necessary.
- ▶ <http://maven.apache.org/plugins/maven-war-plugin/examples/war-overlay.html>