Building and Managing Java Projects with Maven

Alan Wang
Connectria

Agenda

- What is Maven?
- A J2EE example
- Customizing and extending Maven
- Tips and discussions

What is Maven?

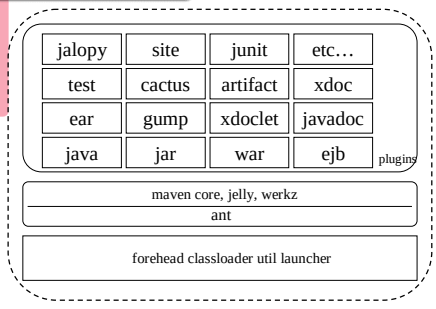
- A Java project management and integration build tool.
- Based on the concept of XML Project Object Model (POM).
- Originally developed for building Turbine.
- A small core with numerous plugins (in Jelly).

Build Tools Retrospective

- One level above ant.
- Make → ant → Maven
- (Assembly → C → C++)

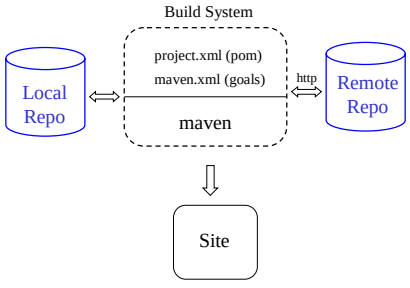
Make	makefile	target
Ant	build.xml	target
Maven	project.xml maven.xml	goals

Under the Hood



Maven

Architecture Overview



Artifact Repository

The Most Important Feature

- Remote Repository

${mave.repo.remote}/<**groupId**>/<**type**>s/<**artifactId**>-<**version**>.<**type**>

```
…
<dependencies>
  <dependency>
    <groupId>xalan</groupId>
    <artifactId>xalan</artifactId>
    <version>2.5.1</version>
    <type>jar</type>
  </dependency>
  …
</dependencies>
…
```

```
- repository
    - […]
    - xalan
        - jars
            - xalan-2.5.0.jar
            - xalan-2.5.1.jar
            - […]
    - […]
```

Artifact Repository

- Local Repository
  - A local mirror/cache of downloaded artifacts from remote repositories.
  - Located at ${user.home}/.maven/repository

${mave.repo.local}/<**groupId**>/<**type**>s/<**artifactId**>-<**version**>.<**type**>

Implementing an Example

- Get Started
  - Download from
    - http://maven.apache.org/start/download.html
  - Current version: 1.0rc1
  - Environment setup
    - export MAVEN_HOME=c:/maven-1.0rc1
    - export PATH="$MAVEN_HOME/bin;$PATH"
      (or set MAVEN_HOME = c:\maven-1.0rc1
      set PATH = %MAVEN_HOME%\bin;%PATH%)
  - run *install_repo.sh* to populate the local repository

Create a New Project

Type:

maven genapp

It will prompt for

- project id
- project name
- project package name

A Sample Service J2EE Project

- EJB (stateless session beans exposed as web services)
- Data components
- Web application

Directory Layout

Project Directory Layout

sampleservice

- project.xml - Master POM of the project
- maven.xml - Reactor definition
- project.properties - Properties related to the project
- application/ - Application component
- service-data/ - Common data component
- service-ejb/ - EJB/WS component
- service-web/ - Web Application component
- target/ - Generated artifact directory
- xdocs/ - Various documents in xml format
- …

Directory Layout

A Component Directory Layout

…

service-data - Data component subproject

- project.xml - POM of the data project
- maven.xml - Goals definition
- project.properties - Properties related to the project
- src/ - Source directory
  - conf/ - Configuration and resource files
  - java/ - Java source files
  - test/ - Test source files
- target/ - Generated artifact directory
- xdocs/ - Various documents in xml format

…

Project Object Model (POM)

Projects are described as Project Object Model.

- Project Management
  - Detailed description of the project.
  - Company information.
  - Developer roles and information.
  - Mailing list and source control modules configuration.
- Project Build
  - Source code and test code location.
  - Resources location

Project Object Model (POM)

- Project Dependency
  - Libraries needed for build and runtime.
- Project Reports
  - Junit reports
  - Javadoc reports
  - Checkstyle reports, ….etc

Project Management Section

```
<project>
  <pomVersion>3</pomVersion>
  <groupId>sampleservice</groupId>
  <name>Sample Service</name>  <!-- Used in Javadoc -->
  <id>sampleservice</id>
  <currentVersion>1.0</currentVersion>

   <!-- Used for document creation -->
  <organization>
   <name>My, Inc.</name>
   <url>http://www.myinc.com</url>
   <logo>/images/logo.gif</logo>
  </organization>
  …
```

(elements in bold are required)

Project Management Section

```
<project>
    […]
    <inceptionYear>2003</inceptionYear> <!-- Used in JavaDoc –
    <package>com.myinc.sampleservice</package> <!-- Used in JavaDoc -->
    <shortDescription>Demo to use maven</shortDescription>  <!-- one liner -->

    <!-- Used in front page-->
    <description>
        A detailed description about this demo
    </description>
    <url>http://www.myinc.com/sampleservice/</url>
    <issueTrackingUrl/>
    <siteAddress>dev.myinc.com</siteAddress> <!-- Used in deployment -->
    <siteDirectory>/www/sampleservice/</siteDirectory>  <!-- Used in deployment -->
    <!-- Used in deployment. If defined, it overrides ${maven.repo.central} -->
    <distributionSite>/www/dist/sampleservice</distributionSite>
    <!-- Used in deployment, final distribution directory -->
    <distributionDirectory>/www/www.myinc.com/somedir</distributionDirectory>
```

Project Management Section

```
<project>
    […]
    <repository>
        <connection>scm:cvs:pserver:anoncvs@cvs.myinc.com:/cvsroot:samples
          ervice</connection>
        <developerConnection>scm:cvs:pserver:${maven.username}@cvs.myinc
          .com:/cvsroot:sampleservice</developerConnection>
        <url>http://cvs.myinc.org/viewcvs/sampleservice/</url>
    </repository>
    <!-- Used in maven:dist -->
    <versions>
        <version>
            <id>1.0-beta-1</id>
            <name>1.0-beta-1</name>
            <tag>1.0-beta-1</tag>
        </version>
    </versions>
    <branches/>
    […]
```

…

```
<mailingLists/>
<contributors/>
<developers/>
…
```

Project Dependency Section

```
<project>
    […]
    <dependencies>
      <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.8</version>
        <properties>
            <ear.bundle>true</ear.bundle>
            <ejb.manifest.classpath>true</ejb.manifest.classpath
              >
        </properties>
      </dependency>
    </dependencies>
    […]
```

Special Dependency: **SNAPSHOT**

Project Dependency Section

Dependency Classloader

```
[...]
<dependency>
    <groupId>bcel</groupId>
    <artifactId>bcel</artifactId>
    <version>5.1</version>
    <properties>
        <classloader>root</classloader>
    </properties>
</dependency>
[...]
```

Maven has three classloaders:
- root -- ant classloader
- root.maven – maven core classloader
- default – plugin classloader

Project Dependency Section

Dependency Override

project.xml

```
...
<dependency>
    <groupId>weblogic</groupId>
    <artifactId>weblogic</artifactId>
    <version>8.1.1</version>
    <properties>
        <classloader>root</classloader>
    </properties>
</dependency>
```

project.properties

```
...
## Dependency override
maven.jar.override = on
maven.jar.weblogic = ${weblogic.home}/lib/weblogic.jar
maven.jar.webservices = ${weblogic.home}/lib/webservices.jar
```

Project Build Section

Defines the location of source, test and resource files.

```
[...]
<build>
    <nagEmailAddress>buildmaster@myinc.com</nagEmailAddress>
    <sourceDirectory>${src.java.dir}</sourceDirectory>
    <unitTestSourceDirectory>${src.test.dir}</unitTestSourceDirectory>
    <aspectSourceDirectory/>

[...]
```

src/java

src/aspect

src/test

Project Build Section

```
<unitTest>
    <includes>
        <include>**/*Test.java</include>
    </includes>
    <resources/>
</unitTest>

<resources>
    <resource>
        <directory>${src.conf.dir}</directory>
        <targetPath/>
        <includes>
            <include>**/*.properties</include>
        </includes>
    </resource>
</resources>
```

src/conf

prefix package name, e.g. com.myinc. sampleservice

Project Report Section

Defines various reports to be generated

```
<reports>
    <report>maven-jdepend-plugin</report>
    <report>maven-checkstyle-plugin</report>
    <report>maven-changelog-plugin</report>
    <report>maven-developer-activity-plugin</report>
    <report>maven-file-activity-plugin</report>
    <report>maven-javadoc-plugin</report>
    <report>maven-jxr-plugin</report>
    <report>maven-junit-report-plugin</report>
    <report>maven-linkcheck-plugin</report>
    <report>maven-tasklist-plugin</report>
</reports>
```

Project Report - Example

Jakarta Turbine

Project Report - XDoc

xdocs/**navigation.xml**

```
…
<menu name="General Information">
   <item name="Overview"          href="/index.html"/>
   <item name="Features"          href="/features.html"/>
   <item name="Specification"     href="/fsd.html"/>
   <item name="Getting Started"   href="/getting-started.html"/>
</menu>
```

General Information
- Overview
- **Features**
- Specification
- Getting Started

Documentation
- Changes
- Core DB Schema
- Services
- How-To's
- JavaDocs

Development
- Proposals
- How To Help
- Todo

xdocs/**features.xml**

```
<document>
    <properties>
        <title>Turbine Features</title>
        <author email="">Jon S. Stevens</author>
    </properties>
    <body>
     <section name="Features">
     <p>This document is for bragging about all of Turbine's ….
     …
</document>
```

Property Processing

${project.home}/project.properties	Project scope properties
${project.home}/build.properties	Properties specific to each build
${user.home}/build.properties	Properties specific to each user
CLI –Dfoo=bar	

Sample build.properties:
…
bea.home=c:/bea81sp1
oracle.home=c:/oracle/ora9i

The ***last*** definition wins

Build Process Design

Sample Service project components
- Common data module
- EJBs
- Web Application

Data (jar) → EJB (jar) → Webapp (war) → EAR

Subproject: service-data

project.xml (POM)

```
<project>
    <extend>../project.xml</extend>
    <name>Sample Service Data
      Module</name>
    <id>sampleservice-data</id>
</project>
```

Note: POM can be inherited.

sampleservice
- –project.xml
- –maven.xml
- –application/
- –**service-data**/
  - –**project.xml**
  - –maven.xml
  - –src/
- –service-ejb/
- –service-web/
- –xdocs/

Subproject: service-data

maven.xml (Goals)

```
<project default="build"
  xmlns:j="jelly:core"
  xmlns:maven="jelly:maven"
  xmlns:ant="jelly:ant">

  <goal name="build"
        prereqs="jar:install"/>

</project>
```

Note: Goals can also be inherited

sampleservice
- –project.xml
- –maven.xml
- –application/
- –**service-data**/
  - –project.xml
  - –**maven.xml**
  - –src/
- –service-ejb/
- –service-web/
- –xdocs/

Make jar → Local Repo

Subproject: service-ejb

project.xml (POM)

```
<project>
    <extend>../project.xml</extend>
    <name>Sample Service EJB</name>
    <id>sampleservice-ejb</id>

    <dependency>
        <groupId>${pom.groupId}</groupId>
        <artifactId>sampleservice-data</artifactId>
        <version>${pom.currentVersion}</version>
        <properties>
            <ejb.manifest.classpath>true</ejb.manifest
              .classpath>
        </properties>
    </dependency>
</project>
```

sampleservice
- –project.xml
- –maven.xml
- –application/
- –service-data/
- –**service-ejb**/
  - –**project.xml**
  - –maven.xml
  - –src/
- –service-web/
- –xdocs/

xdoclet → compile → ws-gen → ejb-jar → Local Repo

Subproject: service-ejb

Maven XDoclet Plugin

- Support all standard tags
- Automatically includes generated src dir in compilation src set
- Support util class generation
- Dependencies
  - xdoclet-ejb-module-1.2
  - xdoclet-web-module-1.2
  - xdoclet-bea-module-1.2
    (for WebLogic Server Deployment)

Note: Current version 1.2 does not work out-of-box (needed to fix the project.xml)

Subproject: service-ejb

Maven XDoclet Plugin - Properties

```
## EJBDoclet
maven.xdoclet.ejbdoclet.fileset.0.include=**/ejb/**/*Bean.java
maven.xdoclet.ejbdoclet.ejbSpec=2.0
maven.xdoclet.ejbdoclet.verbose=true
maven.xdoclet.ejbdoclet.session.0=false
maven.xdoclet.ejbdoclet.localhomeinterface.0=true
maven.xdoclet.ejbdoclet.localinterface.0=true
maven.xdoclet.ejbdoclet.utilobject.0=true

## EJBDoclet WebLogic Nested Element
maven.xdoclet.ejbdoclet.weblogic.0=true
maven.xdoclet.ejbdoclet.weblogic.0.mergeDir=${src.dir}/ejbdoclet
maven.xdoclet.ejbdoclet.weblogic.0.destDir=${maven.xdoclet.ejbdoclet.deploym
    entdescriptor.0.destDir}
```

Subproject: service-ejb

Web Services

- Currently Maven has no container specific plugins

```
<goal name="wsgen" prereqs="wsgen.autotype, wsgen.source2wsdd"/>

 <goal name="wsgen.autotype">
   <taskdef name="autotype"
            classname="weblogic.ant.taskdefs.webservices.javaschema.JavaSchema"/>
   ....
 </goal>

<goal name="wsgen.source2wsdd">
   ...
</goal>
```

project.properties

```
## Web Services
maven.webservice.javaComponents=\
   com.myinc.sampleservice.ejb.BasicAccountInquiry,\
   com.myinc.sampleservice.ejb.ExpandedAccountInquiry,\
   com.myinc.sampleservice.ejb.DetailedAccountInquiry
maven.webservice.autotype.package=com.myinc.sampleservice.autotype
maven.webservice.autotype.keepgenerated=false
```

Use tag: @wlws

Subproject: service-ejb

Maven EJB Plugin

- Currently no container specific ejb-jar

```
<goal name="ejbjar" prereqs="java:compile">
 <path id="ejb.classpath">
   <pathelement location="${maven.build.dest}"/>
   <path refid="maven.dependency.classpath"/>
 </path>
 <j:set var="maven.ejb.descriptordir"
    value="${pom.getPluginContext('maven-xdoclet-plugin')…./>
 <ejbjar srcdir="${maven.build.dest}"
    descriptordir="${maven.ejb.descriptordir}"
    flatdestdir="true"
    basejarname="${pom.artifactId}-${pom.currentVersion}">
    <classpath refid="ejb.classpath"/>
    <weblogic destdir="${maven.build.dir}"newCMP="true"
          outputdir="${maven.build.dir}/ejb" rebuild="false"
          ejbcclass="weblogic.ejbc"
    </weblogic>
    …
```

```
sampleservice
    –project.xml
    –maven.xml
    –application/
    –service-data/
    –service-ejb/
        –project.xml
        –maven.xml
        –src/
    –service-web/
    –xdocs/
```

Subproject: service-ejb

Making the Final EJB Jar

```
<project default="build">
 <goal name="build" prereqs="ejb:install"/>

 <preGoal name="java:compile">
    <attainGoal name="xdoclet:ejbdoclet"/>
 </preGoal>

<postGoal name="java:compile">
    <attainGoal name="wsgen"/>
    <attainGoal name="java:jar-resources"/>
</postGoal>

<preGoal name="ejb:ejb">
   <attainGoal name="ejbjar"/>
</preGoal>

<postGoal name="ejb:install">
  <artifact:install
     artifact="${maven.build.dir}/${pom.artifactId}-${pom.currentVersion}.xml"
     type="xml"  project="${pom}"/>
</postGoal>
```

```
sampleservice
    –project.xml
    –maven.xml
    –application/
    –service-data/
    –service-ejb/
    –service-web/
        –project.xml
        –maven.xml
        –src/
    –xdocs/
```

Subproject: service-web

project.xml (POM)

```
<project>
      <extend>../project.xml</extend>
      <name>Sample Service Web Application</name>
      <id>sampleservice-web</id>
      …
      <dependency>
            <groupId>${pom.groupId}</groupId>
            <artifactId>sampleservice-data</artifactId>
            <version>${pom.currentVersion}</version>
      </dependency>
      <dependency>
            <groupId>${pom.groupId}</groupId>
            <artifactId>sampleservice-ejb</artifactId>
            <version>${pom.currentVersion}</version>
            <type>ejb</type>
            <properties>
                <web-service>true</web-service>
            </properties>
      </dependency>
```

custom property

```
sampleservice
    –project.xml
    –maven.xml
    –application/
    –service-data/
    –service-ejb/
    –service-web/
        –project.xml
        –maven.xml
        –src/
    –xdocs/
```

Subproject: service-web

maven.xml (goals)

```
<project default="build">
    <goal name="build" prereqs="war:install"/>
    <preGoal name="war:war">
      <j:forEach var="dep" items="${pom.dependencies}">
        <j:if test="${dep.getProperty('web-service')=='true'}">
          <util:file var="xmlFile"
            name="${maven.repo.local}/${dep.groupId}/xmls/${dep.artifactId}-
                  ${dep.version}.xml"/>
          <j:if test="${xmlFile.exists()}">
            <x:parse var="xml" xml="${xmlFile}"/>
            <x:forEach var="node" select="$xml/*/*">
              <j:set var="temp" value="${root.add(node.detach())}"/>
            </x:forEach>
          </j:if>
        </j:if>
      </j:forEach>
    <j:if test="${!root.elements().isEmpty()}">
      <!-- output the "web-services.xml" -->
    </j:if>
  </preGoal>
```

preGoal is used to generate web services DD file if defined

Subproject: application

EAR Packaging

```
<project>
  <extend>../project.xml</extend>
  <name>Sample Service Application</name>
  <id>sampleservice</id>
```

```
sampleservice
    –project.xml
    –maven.xml
    –application/
        –project.xml
        –maven.xml
        –src/
    –service-data/
    –service-ej/
    –service-web/
    –xdocs/
```

```
<!-- Data Component -->
<dependency>
  <groupId>${pom.groupId}</groupId>
  <artifactId>sampleservice-data</artifactId>
  <version>${pom.currentVersion}</version>
  <type>jar</type>
  <properties>
    <ear.bundle>true</ear.bundle>
  </properties>
</dependency>
```

```
<!-- EJB Component -->
<dependency>
  <groupId>${pom.groupId}</groupId>
  <artifactId>sampleservice-ejb</artifactId>
  <version>${pom.currentVersion}</version>
  <type>ejb</type>
  <properties>
    <ear.bundle>true</ear.bundle>
  </properties>
</dependency>
```

```
<!-- WAR Component -->
<dependency>
  <groupId>${pom.groupId}</groupId>
  <artifactId>sampleservice-web</artifactId>
  <version>${pom.currentVersion}</version>
  <type>war</type>
  <properties>
    <ear.bundle>true</ear.bundle>
    <ear.appxml.war.context-root>
      /${pom.artifactId}
    </ear.appxml.war.context-root>
  </properties>
</dependency>
```

Subproject: application

```
sampleservice
    –project.xml
    –maven.xml
    –application/
        –project.xml
        –maven.xml
        –src/
    –service-data/
    –service-ejb/
    –service-web/
    –xdocs/
```

EAR Packaging

```
<project default="build">

  <goal name="build" prereqs="ear"/>

  <preGoal name="ear:generate-ear-descriptor">
    <mkdir dir="${maven.ear.appxml.dir}"/>
  </preGoal>
</project>
```

project.properties

```
…
## Ear related
maven.ear.appxml.dir = ${maven.build.dir}/application/META-INF
maven.ear.appxml = ${maven.ear.appxml.dir}/application.xml
maven.ear.appxml.generate = true
```

Tip: container specific DD files can be stored at src/application/META-INF

Putting It Together: Reactor

```
<project default="build"
  xmlns:j="jelly:core"
  xmlns:maven="jelly:maven"
  xmlns:ant="jelly:ant">
  ….
    <goal name="build">
      <maven:reactor
        basedir="${basedir}"
        postProcessing="true"
        includes="*/project.xml"
        excludes=""
        goals="build"
        banner="Building"
        ignoreFailures="false"/>
    </goal>
  …
```

```
sampleservice
    –project.xml
    –maven.xml
    –application/
    –service-data/
    –service-ejb/
    –service-web/
    –xdocs/
```

Putting It Together: Reactor

```
/c/dev/sampleservice/>maven
  __  __
 |  \/  |__ _Apache__ ___
 | |\/| / _` \ V / -_) ' \  ~ intelligent projects ~
 |_|  |_\__,_|\_/\___|_||_|  v. 1.0-rc1-SNAPSHOT

Starting the reactor...
Our processing order:
Sample Service Data Module
Sample Service EJB
Sample Service Web Application
Sample Service Application
+-----------------------------------------
| Building Sample Service Data Module
| Memory: 3M/4M
+-----------------------------------------
build:

java:prepare-filesystem:
    [mkdir] Created dir: C:\dev\sampleservice\service-data\target\classes

java:compile:
….
```

Customizing Maven

- Override plugin properties in
  - project.properties
  - build.properties
- Use maven.xml
  - Override plugin goals
  - Intercept plugin goals with <preGoal/> and <postGoal/>
- Write you own plugin
  - In Java, Jelly, or other scripting language.

Real Life Maven

- Single artifact per project
  - Can be tweaked to deliver multiple artifacts as long as no type conflicts
  - Fine-grained design
- Project Migration/Mavenizing
  - Can co-exist with ant
  - May require different directory structure
- Too Slow?
  - Use console plugin
- Are you ready for maven?
  - Culture change

Summary

- Pros
  - Work out-of-box for standard projects
  - Build assets highly reusable, thanks to core/plugin architecture
  - Build rules are more dynamic
  - Best suited for project integration
  - IDE friendly
- Cons
  - Incomplete documentation
  - Missing convenience details
  - Not yet mature. Still waiting for the R1.

Get More

- http://maven.apache.org
- http://www.onjava.com/pub/a/onjava/2003/10/22/maven.html
- http://www-106.ibm.com/developerworks/java/library/j-maven
- http://www.javausergroup.at/events/maven.pdf
- http://www.theserverside.com/articles/article.jsp?l=MavenMagic
- http://blogs.codehaus.org/people/vmassol/archives/000080.html
- http://www.javaworld.com/javaworld/jw-10-2002/jw-1011-maven.html

Questions