CS520 Web Programming
Introduction to Maven

Chengyu Sun
California State University, Los Angeles

Build

- Preprocessing
- Compilation
- Postprocessing
- Distribution
- Deployment

What is Maven?

- Mostly used as a build tool for Java projects
- It is more than a build tool
  - Project Object Model (POM)
  - Project lifecycles
  - Dependency management
  - Plugin framework
- It is a project management tool

A Simple Maven Example

pom.xml

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>edu.calstatela.cs520</groupId>
    <artifactId>maven-exmaple</artifactId>
    <version>1.0</version>
</project>
```

Run:

```
mvn compile
mvn package
```

pom.xml and modelVersion

- `pom.xml` is a description of the project
- `modelVersion` is the version of the "grammar" of the description

Maven Coordinates

- `groupId`
  - Name of the company, organization, team etc., usually using the reverse URL naming convention
- `artifactId`
  - A unique name for the project under groupId
- `version`
- `packaging`, default: jar
- `classifier`

Maven coordinates uniquely identifies a project.

Convention Over Configuration

- Systems, libraries, and frameworks should assume *reasonable defaults*.

Default Directory Structure

- `src/main/java`
- `src/main/resources` for files that should be placed under classpath
- `src/main/webapp` for web applications
- `src/test/java`
- `target`

Build Lifecycle

- The process for building and distributing a project
- A build lifecycle consists of a number of steps called phases.

Some Default Lifecycle Phases

- `validate`
- `compile`
- `test`
- `package`
- `deploy`

http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference

Goals and Plugins

- Goals, a.k.a. Mojos, are operations provided by Maven plugins

Some Maven Plugins

- `resources`
- `compiler`
- `surefire`
- `jar, war`

http://maven.apache.org/plugins/index.html

Example of Using a Plugin

```
<build><plugins><plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>2.3.2</version>
  <executions><execution>
      <id>default-compile</id>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
      <configuration>
        <target>1.6</target>
      </configuration>
  </execution></executions>
</plugin></plugins></build>
```

About The Plugin Example

- A plugin is uniquely identified by its coordinates just like any other project
- Goals are usually associated (i.e. *bound*) to a build lifecycle phase
- The behavior of a goal can be customized with additional parameters in the <configuration> section

Run a Maven Build

```
mvn <phase>
```

- Maven will go through each build lifecycle phase up to the specified phase
- In each phase, execute the goals bound to that phase

Run a Maven Build in Eclipse

- Need the `m2e` Eclipse plugin
- Right click on the project then select `Run As` → `Maven Build …`
- Give the build a name
- Enter the phase name for `Goals`
- Click `Run`

Why Not Just Use an IDE

- Can your IDE do *everything* you want?
  - Deploy a web application to a remote server
  - Generate source code from some metadata files
  - Create a zip package of selected files for homework submission
  - …

Why Use Maven

- Everybody uses it!
- Common framework for project build and management
  - Project Object Model
  - Build lifecycles
- Archetype
- Dependency management
- Resource filtering

Archetype

- An archetype is a *template* for a Maven project which can be used to create new projects quickly
- Example: creating a project from archetype
  - `maven-archetype-quickstart`
  - `maven-archetype-webapp`
- Users can create new archetypes and publish them through catalogs
  - Main Maven archetype catalog: http://repo.maven.apache.org/maven2/archetype-catalog.xml

Dependency Management

- A dependency of a project is a library that the project depends on
- Adding a dependency to a project is as simple as adding the coordinates of the library to `pom.xml`
- Maven *automatically downloads the library from an online repository* and store it locally for future use

Dependency Example

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
  </dependency>
</dependencies>
```

- Add a dependency to `pom.xml`
- Add a dependency in Eclipse

Dependencies and Repositories

- Search for dependency coordinates at http://mvnrepository.com/
- Maven Central Repository - http://repo.maven.apache.org/maven2/
- Additional libraries and repositories - https://maven.nuxeo.org/

More About Dependency Management

- Dependencies of a dependency are automatically included
- Dependency conflicts are automatically resolved
- See `CSNS2` for example

Resource Filtering

- Use placeholders in resource files and replace them with actual value during the build process

```
<param name="File" value="${app.dir.log}/csns2.log" />
⬇
<param name="File" value="F:/TEMP/csns2/csns2.log" />
```

Resource Filtering Example

```
<build>
   <filters>
      <filter>build.properties</filter>
   </filters>
   <resources>
      <resource>
         <directory>src/main/resources</directory>
         <filtering>true</filtering>
      </resource>
   </resources>
</build>
```

Summary

- Project Object Model (POM)
- Coordinates
- Lifecycles and phases
- Plugins and goals
- Archetype
- Dependency management
- Resource filtering

Further Readings

- *Maven: The Definitive Guide* by Sonatype