Maven

Overview

- Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
- Simply put, a build tool for Java applications.
- History
  - Maven 1 (2003)
  - Maven 2 (2005)
    - Complete rewrite
    - Not backwards compatible
  - Maven 3 (2010)
    - Same as Maven 2 but more stable
    - Current stable version is 3.5.2 available at https://maven.apache.org

Features

- Dependency system
- Multi-module builds
- Consistent project structure
- Consistent build model
- Plugin-oriented

Maven POM

- Stands for Project Object Model.
- Describes a project:
  - Name and version
  - Artifact type
  - Dependencies
  - Plugins
  - Profiles (alternate build configurations for e.g. test, stage, production)

Project identifier and GAV syntax

- Maven uniquely identifies a project using:
  - **groupId**: Project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package.
  - **artfiactId**: Project name (no spaces or colons)
  - **version**: Project version
    - Format: {major}.{minor}.{maintenance}
    - Add “-SNAPSHOT” to specify that the module is in development
- GAV syntax
  - Stands for **groupId:artifactId:version**

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.seavus.someproject</groupId>
    <artifactId>some-module</artifactId>
    <version>1.0</version>
</project>
```

Packaging

- Build type identified using the “packaging” element.
- Supported packaging types:
  - “pom”, “jar”, “war”, “ear”, custom
  - Default is “jar”
- Example:

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>com.seavus.someproject</artifactId>
    <groupId>some-module</groupId>
    <version>1.0</version>
    <packaging>jar</packaging>
</project>
```

Multi-module projects (project aggregation)

- Maven has 1st class support for multi-module projects.
- Each Maven project creates one primary artifact.
- A parent POM is used to group modules.
- Example:

```
<project>
    <groupId>com.seavus.someproject</groupId>
    <artifactId>master-module</artifactId>
    <packaging>pom</packaging>
    <modules>
        <module>some-module</module>
        <module>another-module</module>
    </modules>
</project>
```

Inheritance

- POM files can inherit configuration:
  - groupId, version, project configuration, plugin configuration, dependencies, etc.
- Example:

```
<project>
  <parent>
    <artifactId>com.seavus.someproject</artifactId>
    <groupId>some-parent</groupId>
    <version>1.0</version>
  </parent>
</project>
```

Conventions

Maven is opinionated about project structure (convention over configuration)

- **target**: Default work directory
- **src**: All project source files go in this directory
- **src/main**: All sources that go into primary artifact
- **src/test**: All sources contributing to testing project
- **src/main/java**: All java source files
- **src/main/webapp**: All web source files
- **src/main/resources**: All non compiled source files
- **src/test/java**: All java test source files
- **src/test/resources**: All non compiled test source files

Build lifecycle

A Maven build follows a lifecycle.

- The **“default”** lifecycle:
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - integration-test
  - install
  - deploy
- There are also the **“clean”** and **“site”** lifecycles.

Maven on the command line

Examples:

- **mvn clean** - Invokes the clean lifecycle that deletes all compiled classes and resources.
- **mvn clean compile** - Executes the clean lifecycle followed by the default lifecycle up until its compile phase.
- **mvn test** - Invokes the default lifecycle up until its test phase.
- **mvn install** - Invokes the default lifecycle up until its install phase.

Maven dependency management

Maven and dependencies

- Maven revolutionized Java dependency management
  - No more checking libraries into version control
- Introduced the Maven Repository concept
  - Established Maven Central (https://repo.maven.apache.org/maven2)
- Introduced concept of transitive dependency
- Often include source and Javadoc artifacts

Declaring a dependency

- Dependencies consist of:
  - GAV
  - Scope: compile, test, provided (default = compile)
  - Type: “jar”, “pom”, “war”, “ear”, “zip” (default = “jar”)
- Example:

```
<project>
  <dependencies>
    <dependency>
      <groupId>commons-io</groupId>
      <artifactId>commons-io</artifactId>
      <version>2.5</version>
    </dependency>
  </dependencies>
</project>
```

Repositories

- Dependencies are downloaded from repositories via HTTP(S)
- Downloaded dependencies are cached in a local repository
  - Usually found in ${user.home}/.m2/repository
- Repository follows a simple directory structure
  - {groupId}/{artifactId}/{version}/{artifactId}-{version}.jar
- Maven Central is primary community repository
  - https://repo.maven.apache.org/maven2
- Search Maven Central with:
  - http://search.maven.org
  - https://mvnrepository.com

Adding a third-party repository

- Additional repositories can be defined in the POM
- Example:

```
<project>
  <repositories>
    <repository>
      <id>public-jboss</id>
      <name>Public JBoss</name>
      <url>https://repository.jboss.org/…</url>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

Dependency transitivity

- If project A depends on project B (A → B) and project B depends on project C (B → C), then project A depends on project C as well (A → B → C).
- Only compile and runtime scopes are transitive.
- Transitive dependencies are controlled using:
  - Exclusions
  - Optional declarations

Dependency exclusion

Exclusions exclude transitive dependencies.

```
<project>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.0.5.RELEASE</version>
      <exclusions>
        <exclusion>
          <groupId>commons-logging</groupId>
          <artifactId>commons-logging</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>
</project>
```

Exercises

Exercise: ImprovedWordCounter

- Modify the WordCounter application to ignore the word capitalization using Apache's Commons Lang 3 library.
- Hint: Use https://mvnrepository.com to find the correct GAV for the library in question.

Exercise: MultiModuleWaterHeater

- Modify the project structure of the WaterHeater to use a module for each component and configure the dependencies among the modules.
- Create a separate “application” module to for the “main” application class.

* Alternative: MultiModuleElectricalSystem.Best