# BUILDING WITH MAVEN

# Building with Maven

- Introduction to Maven & its objectives
- Install and setting up Maven
- Maven project structure
- POM.xml description

# Project Management Tool

- A Java build tool
  - “project management and comprehension tool”
    - **It means - *accumulator of knowledge,***
- An Apache Project
  - Mostly sponsored by Sonatype
  - Standard  and Easy way to build and Publish the projects
  - To share JARs across several projects.
  - Communication among members of a working team.

# Installing Maven

- Download Maven :http://maven.apache.org/download.html
- Unzip the installation archive
- Set the M2_HOME and Path environment variables in the following way:
  - **M2_HOME=C:\apache-maven-2.2.1**
  - **Path=%M2_HOME%\bin**
- **mvn –version**
  - Will print out installed version of Maven, indicating successful installation .
  - *Maven is Bunded with all Popular IDEs*

# Pom.xml

- Stands for Project Object Model
- POM is the fundamental unit of work in Maven.
- An XML file that contains information about the project and configuration details used by Maven to build the project.
- While executing Maven looks for the POM in the current directory.
- It reads the POM, gets the needed configuration information, then executes the goal.

# POM

- The following Details about the project can be configured in POM
  - Name and Version
  - Artifact Type
  - Dependencies
  - goals
  - Plugins
  - Profiles (Alternate build configurations)

- **Super POM**
  - Maven's default POM.
  - All POMs extend the Super POM
  - Configuration is inherited by the POMs

# Minimal POM

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1</version>
</project>
```

# Minimal Pom.xml

- Projects are the top-level element in all **pom.xml** files.

– **groupID:**
  – Arbitrary project grouping identifier (no spaces or colons
  – Loosely based on Java package name
- **artfiactId:**
  – Arbitrary name of project (no spaces or colons)
  – Usually base name of the primary artifact
  – Typically files like a JAR or war

– **version:**
  – Version of project
  – Format {Major}.{Minor}.{Maintanence}
  – Add ‘-SNAPSHOT ‘ to identify in development

# Maven Dependency Management

- Maven revolutionized Java dependency management
    - No more checking libraries into version control
- Introduced the Repository concept
    - Established Maven Central
- Created a module metadata file (POM)
- Introduced concept of transitive dependency

# Dependencies

- Used to store Dependencies and then downloaded
  - Via http
- Downloaded dependencies are cached in a local repository
  - ***Usually found in ${user.home}/.m2/repository***

- Maven Central is primary community repo
  - http://repo1.maven.org/maven2
  - After downloading repository,
  - Maven will always look for the artifact in the local repository before looking elsewhere.

# Dependencies

```
<project>
    ...
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
                <scope>provided</scope>
    </dependency>
    </dependencies>
</project>
```

# Resolving Dependency

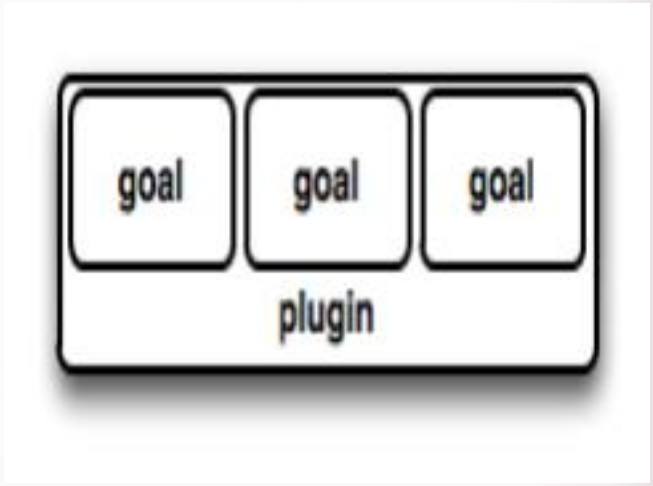
- During the build process dependencies are resolved with dependency management engine.
- Specified in <dependencies> elements within a pom.xml file.
- They are resolved in the following order:
  - Local repository is checked for the dependency.
  - A list of remote repositories is checked for the dependency.
  - In Case 1 and 2 fail and error is reported

# Scopes

- **Compile**
  - default scope
  - dependencies that are available in all classpaths of a project.
- **Provided**
  - Indicates the JDK or a container to provide the dependency at runtime.
  - dependency on the Servlet API
- **Test**
  - The dependency is not required for normal use
  - Available for the test compilation and execution phases.

# Maven Plugin

- Maven is a plugin execution framework
- Tasks are done by plugins.
- Generally used to :
  - create jar file
  - create war file
  - compile code files
  - unit testing of code




- Plugin provides a set of goals

- **mvn [plugin-name]:[goal-name]**

- **mvn compiler:compile**

# Archetype Pluggin

- Maven project templating toolkit.
  - *original pattern or model from which all other things of the same kind are made*.
- Provides a  quick consistent way  with best practices to create a Project
- Developers can have a working Maven project to use as a jumping board
- A Directory with the artifact Id is Created along with Some Sample Code
- mvn  archetype: generate

# Plugin Goals

- A plugin goal represents a specific task to building and managing of a project.
- If a build phase has one or more goals bound to it, it will execute all those goals one by one
- **archetype** plugins with Generate Goal will  create projects.

```
mvn  [plugin-name]:[goal-name]
```

```
mvn  archetype: generate
mvn exec:java
```

# Execute Maven Plugin

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.5.0</version>
        <executions>
          <execution>
            <goals>
              <goal>java</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <mainClass>com.example.demo.App</mainClass>
        </configuration>
    </plugin>
```

-

# Goals

- A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

- **mvn clean**
  - Invokes just clean

- **mvn clean compile**
  - Clean old builds and compile

- **mvn package**
  - **A jar file is created** inside the project/target directory.

# Packaging

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>
      <packaging>jar</packaging>
</project>
```

- Build type identified using the “packaging” element
- Tells Maven how to build the project
- Example packaging types:
  - pom, jar, war, ear, custom
  - Default is jar

# Package

- **mvn package** will execute a series of phases and package to a distributable format, such as a JAR.
  - Validate
  - generate-sources
  - process-sources
  - generate-resources
  - process-resources
  - Compile
- **java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App**

# Maven Jar Plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.2.0</version>
    <configuration>
        <archive>
            <manifest>
                <mainClass>com.example.demo.App</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>
```

# Variables

- *Don't repeat yourself*.
- To assist in ensuring the value is only specified once,
- Maven allows variables in the POM.

```
<properties>
<mavenVersion>2.1</mavenVersion>
</properties>
```

```
<dependency>
    <groupId>org.apache.maven</groupId>
    <artifactId>maven-artifact</artifactId>
    <version>${mavenVersion}</version>
</dependency>
```

-

# Maven Eclipse Java Version

```
<properties>

  <maven.compiler.target>1.8</maven.compiler.target>
  <maven.compiler.source>1.8</maven.compiler.source>

 </properties>
```