**Maven – Built Tool**

1. Manage Dependencies
2. Handles versioning
3. Can easily produce Javadocs

**Ant vs Maven :**

* Ant is a replacement for a build tool called “make”, Make is built on top of unix and it is not cross platform supported
* Ant isn’t a build tool. Its really a scripting tool. You have to explicitly do everything.
* Maven is a full fledged build tool. Lot of implicit functionality
* Consistency across projects. Able to achieve inheritance across projects. Built across versioning. Mainly supports transitive dependency.(You download a jar. It downloads all the necessary jars it needed automatically to work within the application)

1. One major difference between Maven and ANT is that Maven requires *less configuration* than ANT because it works on principle of Convention over configuration and assumes reasonable default e.g. java source file in ${basedir}/src/main/java, resources on ${basedir}/src/main/resources, JUnit test cases on ${basedir}/src/test/java etc. It also creates Java class files on ${basedir}/target/classes and JAR file on ${basedir}/target directory. On the other hand ANT requires all these directories supplied as configuration, usually in [ANT build file e.g. build.xml](http://javarevisited.blogspot.sg/2010/10/ant-tutorial-part-2.html).  
     
     
   2) Another significant difference between Maven and ANT is *dependency Management*. Maven introduced concept of repository, which is a central place to store all libraries, JARs etc. Maven allows you to use central maven repository as well as local repository and automatically download dependency during build process. While ANT based project generally use ${lib} as directory to store dependencies. Changing and updating dependency is much easier in maven than ANT because you don't need to manually download dependency. Having an organization wide central repository also helps to remove redundancy across different projects.  
     
     
   3) Third and most important difference between Maven and ANT is that, Maven offers a consistent and common interface to build Java projects. All you need to do is download the project and run mvn install to build it. Also by knowing maven conventions and looking at pom.xml, one can easily understand where source files are and what are project dependencies.  
     
     
   4) Another technical difference between ANT and Maven is that ANT task doesn't have any lifecycle, you need to define targets and there dependencies. While Maven has lifecycle, which is invoked when you run commands like mvn install. Maven executes a series of steps as a result of this command to produce artifacts e.g. [JAR file](http://javarevisited.blogspot.sg/2012/10/5-ways-to-add-multiple-jar-to-classpath-java.html), which is end of life cycle.  
     
     
   5) Maven also enforce a standard naming convention for artifacts defined using groupId, artifactId and version. Also one more difference between ANT and Maven is that,
2. Maven is more than just a build tool, it act as project management tool and can generate reports etc, Though I have yet to use this feature.

**Java Installation :**

Download the Java SE Development Kit from Oracle site based on your OS and install.

Then set the system variable "path" with JDK and JRE bin directories; user variable JAVA\_HOME to JDK folder

path : C:\Program Files\Java\jdk1.8.0\_161\bin;C:\Program Files\Java\jre1.8.0\_161\bin

JAVA\_HOME : C:\Program Files\Java\jdk1.8.0\_161

Check if the installation is correct by going to the command prompt and typing java, javac,java-version

**Maven Installation :**

Go to maven.apache.org.

Download the binary zip archive file and install maven.

Then set the MAVEN\_HOME system variable, JAVA\_HOME system variable, add following to the path system variable

MAVEN\_HOME : C:\Program Files\maven\apache-maven-3.5.2

JAVA\_HOME : C:\Program Files\Java\jdk1.8.0\_161

path : %JAVA\_HOME%\bin;%MAVEN\_HOME%\bin

Check if the installation is correct by going to the command prompt and typing mvn-version

**Simple project:**

* Open STS. Create new 🡪other 🡪General 🡪 Project. Give the project name
* Right click on the project, create new 🡪 file 🡪 pom.xml

<project>

<groupId>com.pluralsight</groupId><!-- Company Name -->

<artifactId>HelloWorld</artifactId><!-- App Name -->

<version>1.0-SNAPSHOT</version>

<modelVersion>4.0.0</modelVersion>

<packaging>jar</packaging>

</project>

* Right click on the project and create the folders src, under that main and under that java. This is the structure used by maven. Then create a java file under java folder and write a sample code.
* Open up command prompt, go to the project folder and give the following commands,

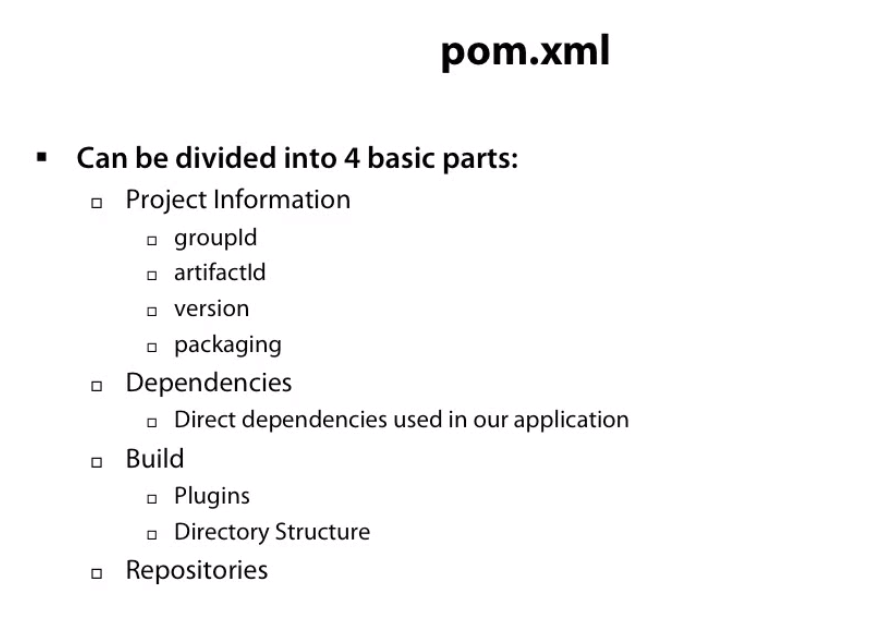
Mvn clean

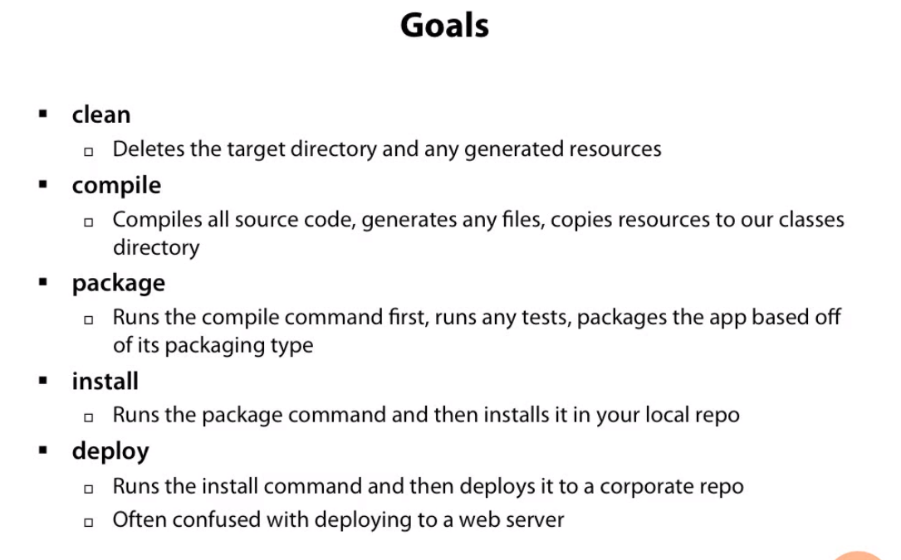
Mvn compile

Mvn package

* Now target & classes folders will be created and the class file will be available there. Mvn package will create a jar file as we have specified jar in pom.xml

POM – Project Object Model



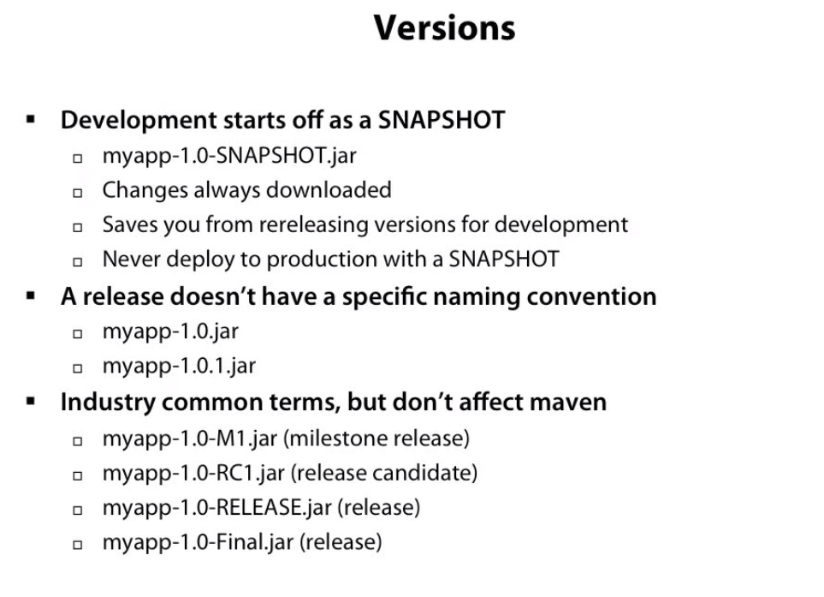


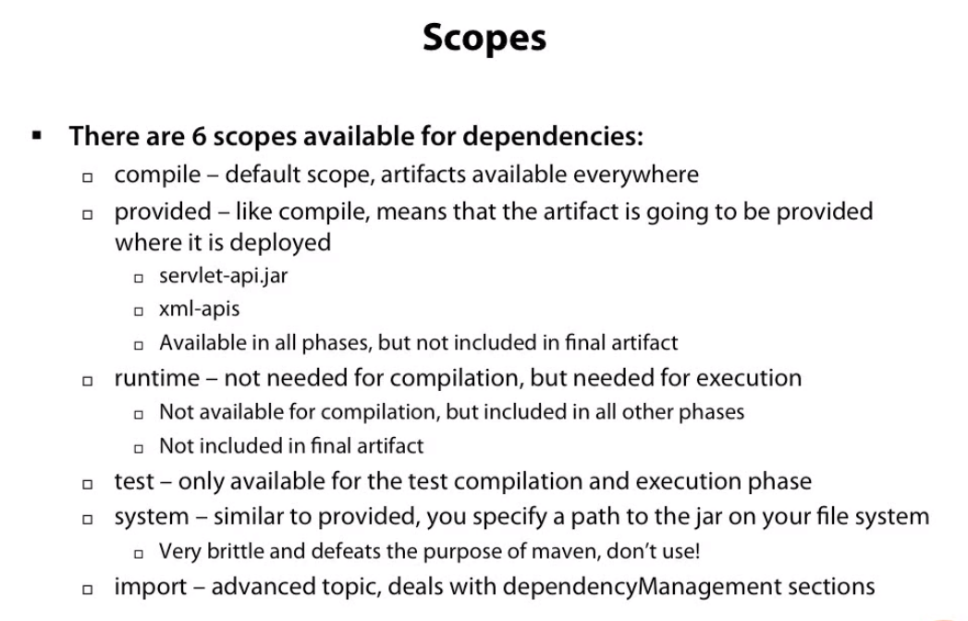
All the dependency jars are downloaded in the following location by default,

C:\Users\<yourusername>\.m2\repository

**Maven Dependencies :**

**Transitive Dependencies :** Dependencies of a dependency





**Maven Repositories:**

By default, it looks at the local repository(.m2 folder in user directory) for the files. If the files are not found, it looks at the central repository(<http://repo.maven.apache.org/maven2/> - Default location), then on Remote repository. You can add multiple repositories

Two types : Dependency repository, Plugin Repository

<repositories>

<repository>

<id></id>

<name></name>

<url></url>

<snapshots>

<enabled></enabled>

</snapshots>

<releases>

<enabled></enabled>

</releases>

</repository>

</repositories>

<pluginRepositories>

<repository>

<id></id>

<name></name>

<url></url>

<snapshots>

<enabled></enabled>

</snapshots>

<releases>

<enabled></enabled>

</releases>

</repository>

</pluginRepositories>

**Maven Plugins**

Goals are plugins configured in Maven install itself. They can be executed in the appropriate phases.

Phases/Build Life cycle are : validate, compile, test, package, integration test, verify, install, deploy

*Compiler Plugin:* By default,compiler version is 1.5. You can change it using compiler plugin configuration.

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

*Jar Plugin:* You have options to create the manifest file with this plugin

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>2.4</version>

<configuration>

<useDefaultManifestFile>true</useDefaultManifestFile>

</configuration>

</plugin>

*Source Plugin:*  With this plugin, you can tie the source code to the JAR file

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>2.2.1</version>

<executions>

<execution>

<id>attach-sources</id>

<phase>verify</phase>

<goals>

<goal>jar</goal>

</goals>

</execution>

</executions>

</plugin>

*Javadoc plugin:* Adding Javadoc to the jar

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-javadoc-plugin</artifactId>

<version>2.9</version>

<executions>

<execution>

<id>attach-javadocs</id>

<phase>verify</phase>

<goals>

<goal>jar</goal>

</goals>

</execution>

</executions>

</plugin>

Maven Archetypes : archetype : generate -To create a sample projects. For example, to generate sample MVC project