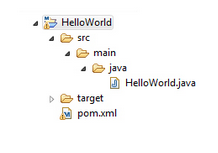
**At its simplest, Maven is a build tool**

* It is used to produce Project
* It helps us manage dependencies (Jars in Java) easily
* It handles versioning and releases
* Describes what your project is doing or what it produces
* Can easily produce Javadocs as well as other site information
* Maven is managed by the Apache Software Foundation and it is Open Source
* It is supported by all the popular IDEs

**Maven pom.xml**



**Maven Folder Structure**



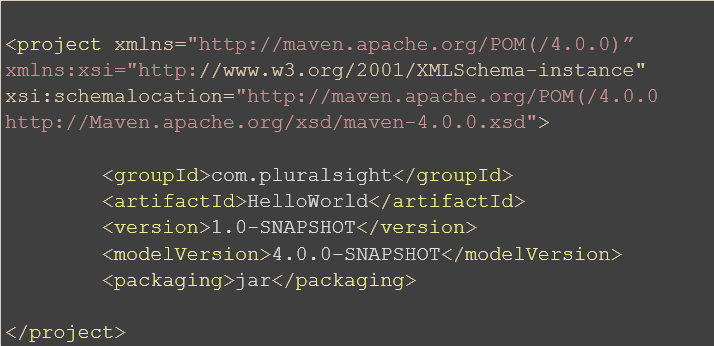
**src/main/java [Folder]**

* **Where we store our Java code**
  + The beginning of our package declaration
    - com.yourcompanyname.division
* **What about other languages**
  + src/main/groovy
* **What about testing**
  + src/test/java [We Can Writer our Test Cases]

**target [Folder]**

* Where everything gets compiled to
* Also where tests get ran from
* Contents in this directory get packaged into a jar, war, etc…

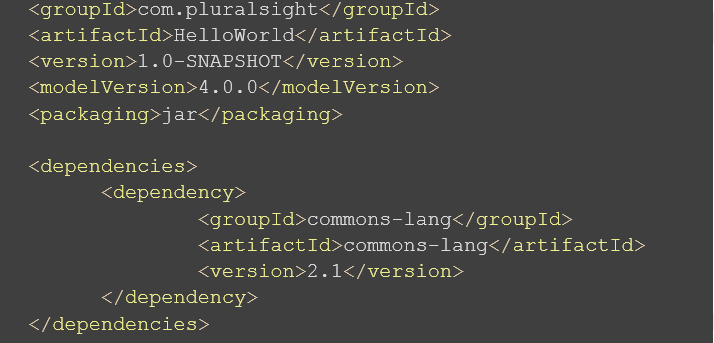
## pom.xml



**Dependencies**

* What we want to use in our application
* We have to know the groupId, artifactId, and the version of what we are looking for
* Added to a dependencies section to our pom file

## pom.xml with our new dependency



**Goals**

**Goal is responsible for a specific task.**

* **validate**: validate the project is correct and all necessary information is available.
* **compile**: compile the source code of the project.
* **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed.
* **package**: take the compiled code and package it in its distributable format, such as a JAR.
* **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run.
* **verify**: run any checks to verify the package is valid and meets quality criteria
* **install**: install the package into the local repository, for use as a dependency in other projects locally.
* **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

**Local Repo**

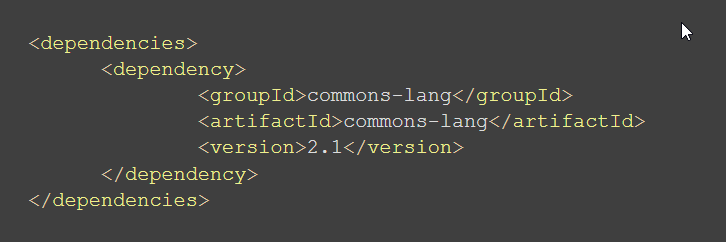
* **Where Maven stores everything it downloads**
  + Installs in your home directory\.m2
    - C:\Users\<yourusername>\.m2\repository

## 

## Dependency

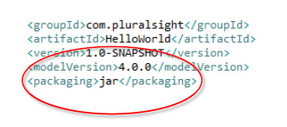
Need at a minimum 3 things:

* groupId
* artifactId
* version



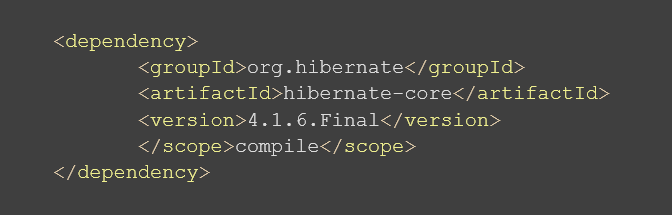
**Types**

* Current core packaging types are:
  + pom, jar, maven-plugin, ejb, war, ear, rar, par
  + The default packaging type is jar
* The type of pom is referred to as a dependency pom
  + Downloads dependencies from that pom

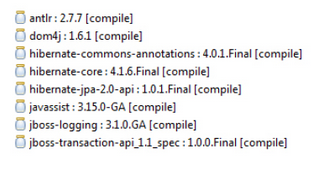


**Transitive Dependencies**

* The main reason people begin using maven
* If we add a dependency:



 If we add a dependency it downloads it’s transitive dependencies:



### Scopes

* There are 6 scopes available for dependencies:
  + compile – default scope, artifacts available everywhere
  + provided – like compile, means that the artifact is going to be provided where it is deployed
    - servlet-api.jar
    - xml-apis
    - Available in all phases, but not included in final artifact
  + runtime – not needed for compilation, but needed for execution
    - Not available for compilation, but included in all other phases
    - Not included in final artifact
  + test – only available for the test compilation and execution phase
  + system – similar to provided, you specify a path to the jar on your file system
    - Very brittle and defeats the purpose of maven, don’t use!
  + import – advanced topic, deals with dependencyManagement sections

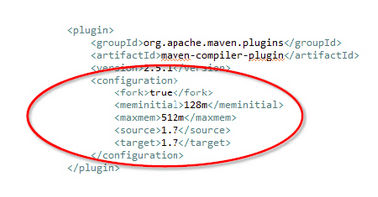
**Repositories**

* **Simply web location from where you can download the Dependencies.**

 Default **location** : [http://repo.maven.apache.org/maven2](https://www.devopsschool.com/slides/maven/maven-fundamentals/index.html#/43)

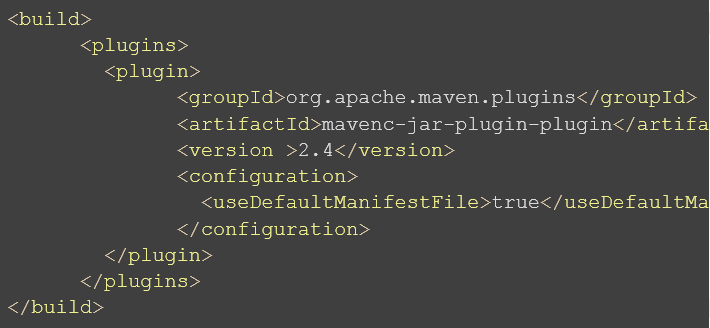
**Compiler Plugin**

* **Used to compile code and test code**
* **http://maven.apache.org/plugins/maven-compiler-plugin/index.html**
* **Invokes Javac, but with the classpath set from the dependencies**
* **Defaults to Java 1.5 regardless of what JDK is installed**



**Jar Plugin**

* **Used to package code into a jar**
* **http://maven.apache.org/plugins/maven-jar-plugin/index.html**
* **Tied to the package phase**
* **Configuration section allows customization**
  + Includes/Excludes
  + Manifest



**Source Plugin**

* **Used to attach source code to a jar**
* **http://maven.apache.org/plugins/maven-source-plugin/index.html**
* **Tied to the package phase**
  + Often overridden to a later phase

<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>

**Notes**

* Goals are really just configured plugins in your application
* Plugins are tied to one of the defined phases, but can usually be overridden.
* The compile plugin is already defined for you, but is often changed to use a specific version of Java.
* The jar plugin is one of the default plugins and can be configured to produce artifacts to specific needs.
* Source and Javadocs can easily be generated to be installed in your corporate repository for use by other developers.