The build tool is used to set up everything which is required to run your java code independently. This can be applied to your entire java project. It generates source code, compiling code, packaging code to a jar etc. Maven provides a common platform to perform these activities which makes programmer’s life easier while handling the huge project.

Maven provides pom.xml which is the core of any project. This is the configuration file where all required information’s are kept. Many of the IDEs (Integrated Development Environments) are available which makes it easy to use. IDEs are available for tools like Eclipse, NetBeans, IntelliJ etc.

Maven stores all project jars. Library jar is in a place called repository which could be a central, local or remote repository. Maven downloads the dependency jar from a central repository. Most of the commonly used libraries are available in [http://repo1.maven.org/maven2/](https://repo1.maven.org/maven2/).

Downloaded libraries are stored in the local repository called m2. Maven uses the libraries available in an m2 folder and if any new dependency added then maven downloads from the central repository to local repository. If libraries are not available in the central repository then maven looks for the remote repository. The user has to configure the remote repository in pom.xml to download from the remote repository.

**Below is the example of configuring a remote repository to *pom.xml* file**. Provide id and URL of the repository where libraries are stored.

|  |
| --- |
| <repositories>       <repository>           <id>libraryId</id>           <url>http://comanyrepositryId</url>       </repository>  </repositories> |

**General Phrases used in Maven:**

* **groupId**: Generally groupId refers to domain id. For best practices company name is used as groupId. It identifies the project uniquely.
* **artifactId**: It is basically the name of the Jar without version.
* **version**: This tag is used to create a version of the project.
* **Local repository**: Maven downloads all the required dependencies and stores in the local repository called m2. More details regarding the same would be shared in the next topic.

**Build Life Cycle:**

Basic maven phases are used as below.

* **clean**: deletes all artifacts and targets which are created already.
* **compile**: used to compile the source code of the project.
* **test**: test the compiled code and these tests do not require to be packaged or deployed.
* **package**: package is used to convert your project into a jar or war etc.
* **install**: install the package into the local repository for use of another project.

**Maven Setup:**

**Step 1**: To setup Maven, download the maven’s latest version form Apache depending upon different OS.

**Step 2**: Unzip the folder and save it on the local disk.

**Step 3**: Create environment variable for MAVEN\_HOME. Follow the below step:

Navigate to System Properties ->Advanced System Settings>Environment Variable ->System Variable ->New ->Add path of Maven folder

**Step 4**: Edit path variable and provide the bin folder path.

**Step 5**: Now verify the maven installation using command prompt and don’t forget to setup JAVA\_HOME

Use mvn –version to verify maven version and output comes like below.

**Install maven IDE in Eclipse:**

Maven provides IDE to integrate with eclipse. I am using eclipse Juno here.

Navigate to Help->Eclipse Marketplace-> Search maven ->Maven Integration for Eclipse ->INSTALL

After installation, you have to restart eclipse.

Then right-click on *pom.xml* and verify all the options are available like below.

**Create Maven project:**

**Step 1**: Navigate to File- new-others-Maven-Maven Project-Click Next

**Step 2**: Check the Create a simple project and click Next

**Step 3**: Provide Group Id and Artifact Id.You can change the version of Jar as per your wish. Here I am using the default name. Click Finish.

**Step 4**: After finish, you will find the project structure is created like below. *pom.xml* is created which is used to download all dependencies.

***pom.xml* file looks like below:**

|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.softwaretestinghelp.test</groupId>  <artifactId>com.softwaretestinghelp.selenium</artifactId>  <version>0.0.1-SNAPSHOT</version>  </project> |

**Step 5**: Add dependencies for Selenium.

All selenium Maven artifacts are available in below central repository

[http://repo1.maven.org/maven2/org/seleniumhq/selenium/](https://repo1.maven.org/maven2/org/seleniumhq/selenium/)

Add following dependencies in *pom.xml* for selenium

|  |
| --- |
| <dependency>         <groupId>org.seleniumhq.selenium</groupId>         <artifactId>selenium-java</artifactId>         <version>2.41.0</version>   </dependency> |

Similarly, following is the dependency for Junit :

|  |
| --- |
| <dependency>        <groupId>junit</groupId>        <artifactId>junit</artifactId>        <version>4.4</version>   </dependency> |

If you want to add other third-party jars then add those dependencies in *pom.xml*

**Step 6**: Final *pom.xml* will be like below:

|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.softwaretestinghelp.test</groupId> <artifactId>com.softwaretestinghelp.selenium</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>         <groupId>org.seleniumhq.selenium</groupId>         <artifactId>selenium-java</artifactId>         <version>2.41.0</version>     </dependency>     </dependencies>  </project> |

**Step 7**: Maven will download all the dependency jars into the local repository called .m2.

M2 folder is basically inside Users->username->m2

All the jars will be placed in a folder called repository which is inside .m2 folder. Maven will create separate folders for the different version and different group id.

**Step 8**: If an m2 folder does not populate in Maven dependencies, then you can populate those jars manually.

– Eclipse Windows ->Preference  
– Navigate Java->Build Path->Classpath Variables

– Click New Button ->Define M2\_REPO and provide the path of the m2 folder.

**Step 9**: Upon successful setup, you will find Maven Dependencies folder like below which will have the required dependency jar for the project

**Build the Project:**

The project can be built by both using IDE and command prompt.

Using IDE you have to right click on POM-*Run As-Maven* Build

Enter goals like clean install etc. and click Run

|  |
| --- |
| [INFO] Scanning for projects...  [INFO]  [INFO] ------------------------------------------------------------------------  [INFO] Building com.softwaretestinghelp.0.0.1-SNAPSHOT  [INFO] ------------------------------------------------------------------------  [INFO]   [INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ com.softwaretestinghelp ---[INFO] Deleting C:\Users\rshwus\WORKSPACE\com.softwaretestinghelp\target  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time: 0.702s  [INFO] Finished at: Sat May 24 18:58:22 IST 2014  [INFO] Final Memory: 2M/15M  [INFO] ---------------------------------------------------------------------- |