---------------------------------------------------------**Maven** -----------------------------------------------------------------

Build and dependency management tool..

Convention based –No need to mention where the code is.

**What are the main contents in POM.XML?(GAVMP)**

**POM** is an acronym for **Project Object Model**. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.

Before maven 2, it was named as project.xml file. But, since maven 2 (also in maven 3), it is renamed as pom.xml.

**Group ID –**Organization ID (Info about the organization)

**Artifact ID—** Name of the project

**Version---**Version of the project(**SNAPSHOT**—still under development)

**Model Version**—POM.XML version

**Package—**Type of the package(war,Jar)

## Elements of maven pom.xml file

For creating the simple pom.xml file, you need to have following elements:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **project** | It is the root element of pom.xml file. |
| **modelVersion** | It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0. |
| **groupId** | It is the sub element of project. It specifies the id for the project group. |
| **artifactId** | It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs. |
| **version** | It is the sub element of project. It specifies the version of the artifact under given group. |

*File: pom.xml*

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd"**>**
6. **<modelVersion>**4.0.0**</modelVersion>**
7. **<groupId>**com.javatpoint.application1**</groupId>**
8. **<artifactId>**my-app**</artifactId>**
9. **<version>**1**</version>**
11. **</project>**

## Maven pom.xml file with additional elements

Here, we are going to add other elements in pom.xml file such as:

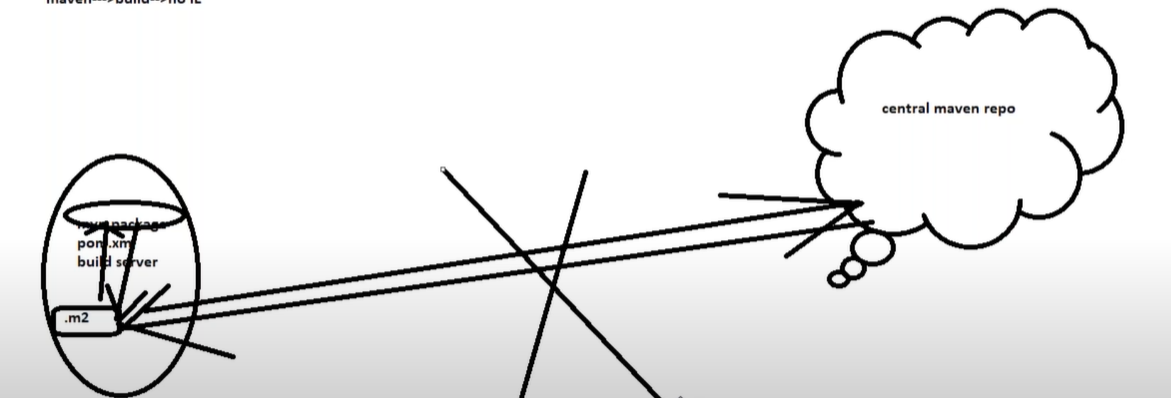
|  |  |
| --- | --- |
| **Element** | **Description** |
| **packaging** | defines packaging type such as jar, war etc. |
| **name** | defines name of the maven project. |
| **url** | defines url of the project. |
| **dependencies** | defines dependencies for this project. |
| **dependency** | defines a dependency. It is used inside dependencies. |
| **scope** | defines scope for this maven project. It can be compile, provided, runtime, test and system. |

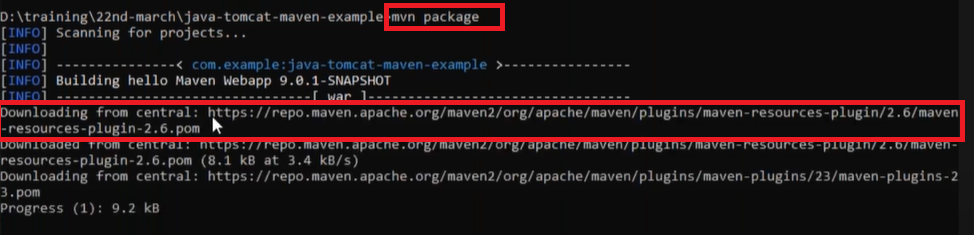
*File: pom.xml*

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd"**>**
6. **<modelVersion>**4.0.0**</modelVersion>**
8. **<groupId>**com.javatpoint.application1**</groupId>**
9. **<artifactId>**my-application1**</artifactId>**
10. **<version>**1.0**</version>**
11. **<packaging>**jar**</packaging>**
13. **<name>**Maven Quick Start Archetype**</name>**
14. **<url>**http://maven.apache.org**</url>**
16. **<dependencies>**
17. **<dependency>**
18. **<groupId>**junit**</groupId>**
19. **<artifactId>**junit**</artifactId>**
20. **<version>**4.8.2**</version>**
21. **<scope>**test**</scope>**
22. **</dependency>**
23. **</dependencies>**
25. **</project>**

**Can we do maven build without an internet?**

No we can’t build ,bcz whatever the dependencies in POM.XML it will try to find it on .M2(local repo) if not present then it’ll be downloaded from central repo and that will be stored in .M2(as a cache)





**When we provide ” mvn install “what all things happen in background?**

There are three build lifecycles: **default, clean and site**.

The default lifecycle handles project deployment,

the clean lifecycle handles project cleaning(target),

while the site lifecycle handles the creation of project's site documentation(ex :: no of dependancies,plugins ,transitive dependancies etc)

**default lifecycle** comprises of the following phases

VCT PVID

* validate - validate the project is correct and all necessary information is available.
* compile - compile the source code of the project**(Provides .Class file).**
* test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed**(.surefire reports)**
* package - take the compiled code and package it in its distributable format, such as a JAR.
* verify - run any checks on results of integration tests to ensure quality criteria are met
* install - install the package into the local repository(.m2), for use as a dependency in other projects locally**(Usually pakages will be stored in target folder)**



* deploy - done in the build environment, copies the final package to the remote repository**(like Nexus,Jfrog,Artifactory)** for sharing with other developers and projects.

**Note** :: Maven will first validate the project, then it will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

**What are the settings you need to do before running mvn deploy?**

\*\* in .m2 folder(local repo)need to have settings.xml (in that need to have username ,password for the artifactory)



\*\* In pom.xml need to mention the same



* Deploy== validate +compile +test +package +verify + install

**Why maven takes much time for 1st execution and from 2nd execution it will take less time?**

When we execute maven goals for the first time maven will download all the dependencies on to the local repo(.M2),when we execute maven goals again it’ll not download dependancies again ,so obviously it takes less time

**.M2 is local repo for maven ,now I don’t want to use .M2 folder as my local ,I want to use some other folder as my local,how can I do that?**

By default maven’s local repository exist on **‘${user.home}/.m2/repository’**. In different operating systems, these path are resolved to –

Windows 7: C:/Documents and Settings/<username>/.m2/repository

Windows 10: C:/Users/<username>/.m2/repository

Linux: /home/<username>/.m2/repository

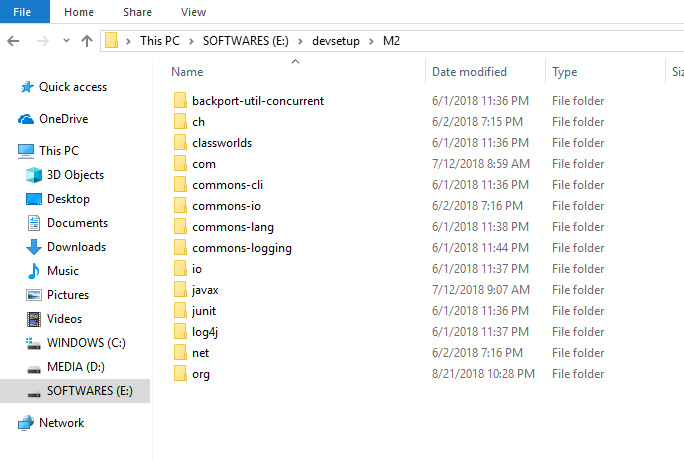
Mac: /Users/<username>/.m2/repository

We can change local repository location to some other location of our choice.

After you have downloaded the maven, follow given simple steps to change maven local repository location to some other path.

* Open file **settings.xml** in edit mode in some text editor.
* Fine the tag **<localRepository>**
* Update the desired path in value of this tag. Save the file.

|  |
| --- |
| setting.xml (under .M2) |
| **<!-- localRepository**  **| The path to the local repository maven will use to store artifacts.**  **|**  **| Default: ${user.home}/.m2/repository -->**  **<localRepository>E:/devsetup/M2</localRepository>**  **</settings>** |



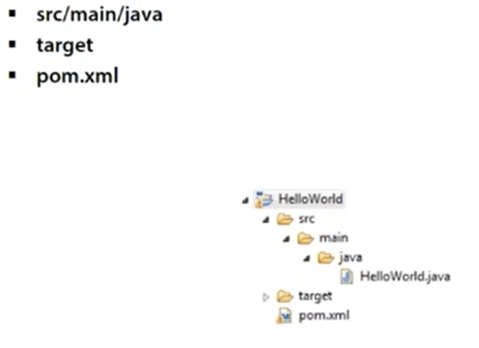
**Maven follows convention over configuration that means it assumes code should be there under src/main/java .test cases under src/tests and many more. Here my requirement is I don’t want to follow that conventions I need to use different folder structure is that possible in maven?**

Super POM

The Super POM is Maven's default POM. All POMs extend the Super POM unless explicitly set, meaning the configuration specified in the Super POM is inherited by the POMs you created for your projects.

**Where does maven stores in dependencies?**

Homefolder/.M2/repositories

**Maven folder structure?**

**In which tag we will mention output artifact type?**

POM.XML-🡪 packaging

1. <project xmlns="http://maven.apache.org/POM/4.0.0"
2. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4. http://maven.apache.org/xsd/maven-4.0.0.xsd">
5. ...
6. <packaging>war</packaging>
7. ...
8. </project>

When no packaging is declared, Maven assumes the packaging is the default: jar. If we want specific type then we need to explicitly mention on the POM ,packaging section.

**Command to skip the test cases in maven?**

mvn package -Dmaven.test.skip=true

Or defined in pom.xml

pom.xml

<properties>

<maven.test.skip>true</maven.test.skip>

</properties>

mvn package –DskipTests

**What is Maven Site used for ?**

It generates site documents for the project ,it has 4 phases those are pre-site ,site ,post site and site deploy .It generates project reports which include project license ,project summary ,about details,dependancies source repo ,CI report and it’ll create html doc

**In maven we’ve local,central and remote repo ,when we execute any maven goals it will take less time right and it will store all the dependancies in ,M2 ? i do want to store it on this location ,so is it customizable?**

1)edit **settings.xml file** where in we need to edit **local repo tag**

2)mvn install –d local.repo command

## Local repository

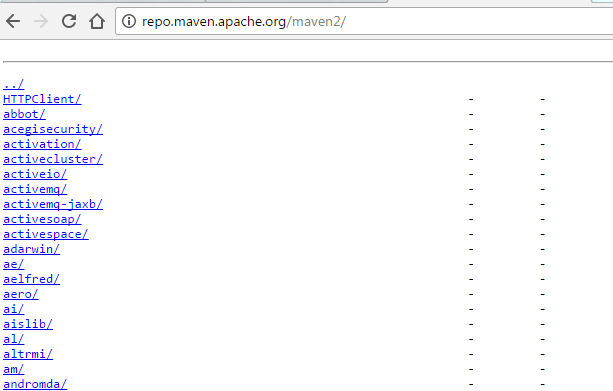
**Maven local repository** reside in the developer’s machine. Whenever you run maven goals which require these dependencies, maven will download the dependencies from remote servers and store them into developer’s machine.

By default, Maven create the local repository inside user home directory i.e. C:/Users/superdev/.m2 directory. You can change the location of the local repository in [setting.xml](https://howtodoinjava.com/maven/maven-settings-file/) file using localRepository tag.

|  |
| --- |
| <settings>      <localRepository>          C:\M2      </localRepository>  </settings> |

## Central repository

**Maven central repository** is located at [http://repo.maven.apache.org/maven2/](https://repo.maven.apache.org/maven2/). Whenever you run build job, maven first try to find dependency from local repository. If it is not there, then, by default, maven will trigger the download from this central repository location.



## Remote repository

Apart from central repository, you may have needed artifacts deployed on other remote locations. For example, in your corporate office there may be projects or modules specific to organization only. In this cases, organization can create remote repository and deploy these **private artifacts**. This remote repository will be accessible only inside organization.

These **maven remote repository** work exactly same way as maven’s central repository. Whenever an artifact is needed from these repositories, it is first downloaded to developer’s local repository and then it is used.

You can configure a remote repository in the POM file or super POM file in remote repository itself.

|  |
| --- |
| <repositories>     <repository>         <id>org.source.repo</id>         <url>[http://maven.orgName.com/maven2/](http://maven.orgname.com/maven2/)</url>     </repository>  </repositories> |

**How to save username and password in POM.XML file?**

If you have authentication to Maven repos in your organization, you normally store the username and password in the Maven settings file located by default at ~/.m2/settings.xml.

For example, I might have something like this in my settings.xml:

<servers>

<server>

<id>myorg-internal-repo</id>

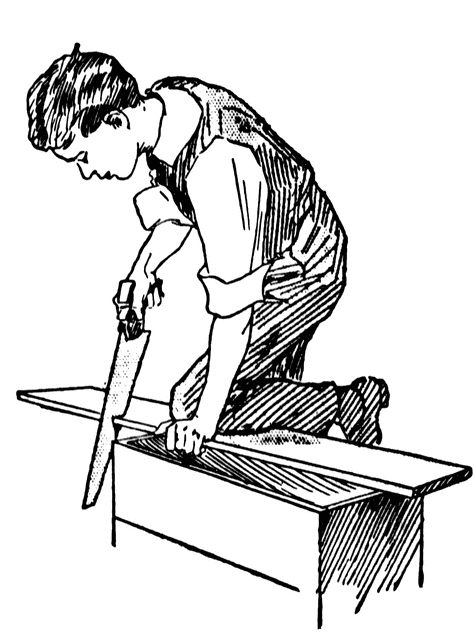
<username>scott.shipp</username>

<password>notMyRealPasswordForAnything</password>

</server>

</servers>

Obviously, storing a password in a clear-text file like this is foolish.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--OEfeoIpQ--/c_limit,f_auto,fl_progressive,q_66,w_880/http:/code.scottshipp.com/wp-content/uploads/2017/09/carpenter_1_md.gif)

**How to encrypt and replace the password**

To remedy this issue, follow these steps:

**Create a master password**

1. First, you must create a master password that is used to encrypt all the other Maven passwords. Start by opening a terminal.
2. Type:

$ mvn --encrypt-master-password

1. You will be prompted for a master password. Enter the password here.
2. Maven will spit out a big long string like this:
3. {w5+NYEttGTAHV3FanFoel4N5uUmbcvtzRoWZHI5N97jtssbo0O/93W/XLlm0caeM}

Keep this terminal window open while you do the next step.

**Store the master password**

1. Create a file called settings-security.xml in the ~/.m2 directory.
2. Copy/paste the following block into the new file:

<settingsSecurity>

<master></master>

</settingsSecurity>

1. Copy/paste the big long encrypted string that Maven spit out in the previous steps in between the <master> tags. You'll end with something like this:

<settingsSecurity>

<master>{w5+NYEttGTAHV3FanFoel4N5uUmbcvtzRoWZHI5N97jtssbo0O/93W/XLlm0caeM}</master>

</settingsSecurity>

1. Save the security-settings.xml file, obviously!

**Encrypt your password**

1. In the given example, the settings.xml server entry has a password of 'notMyRealPasswordForAnything'. This is what we want to encrypt. So open a terminal if you aren't already in one.

Type:

$ mvn --encrypt-password

1. Enter the password you want to encrypt (in our fake example scenario, it's 'notMyRealPasswordForAnything').
2. Maven will spit out an encrypted string that looks similar to the encrypted string it spit out for the master password.
3. Copy the new string it spit out.
4. Open the settings.xml file.
5. Delete the current password between the <password> tags.
6. Paste in the new encrypted version.
7. Save the file.
8. Verify that Maven can still access the repo in question.

**What is the default tag for packaging?**

Ans :: Jar

**Diff b/w maven and Ant**

|  |  |
| --- | --- |
| **Maven** | **Ant** |
| No need to provide information about the project in POM.xml | Need to provide information about the project in Build.xml |
| Maven will work with pom.xml and it has targets. | Ant will work with build.xml and it has goals. |
| Maven has life cycle | Ant doesn’t have life cycle |
| Maven plugins are reusable | Ant scripts are not reusable |
| It is more preffered | Less Preffered |

**What is Maven?**

Maven is a build automation tool used primarily for Java projects.

**Features of maven?**

\*Simple and fast \*Easy to learn \* Dependancy management \* Multiple projects \* Extensibility

**What is an artifact?**

It is a file ,usually a jar that’s get deployed onto a maven repo

**What is a snapshot?**

Snapshot indicates current development copy.

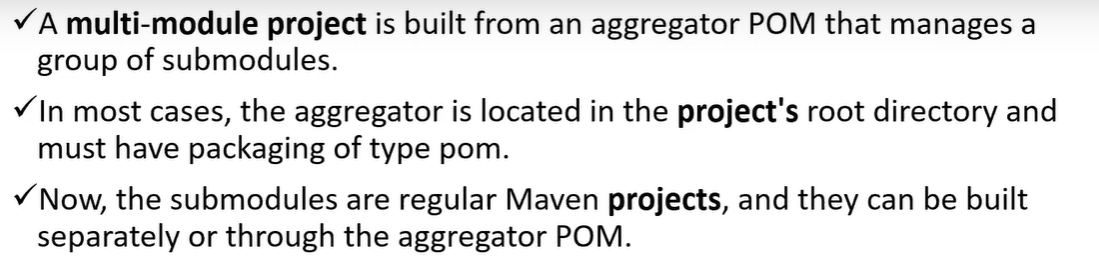
**What is Settings.XML**

Most commonly, it is used to define a local repository location, alternate remote repository servers, and authentication information for private repositories

**Difference b/w JAR,EAR,WAR**

|  |  |  |
| --- | --- | --- |
| **JAR** | **WAR** | **EAR** |
| JAVA Archive | WEB Archive | Enterprise Archive |
| Collection of .class files | It contains Servelets, JSP,CSS,HTML and many other things (Only web related technologies we can use) | J2EE |
|  | Web application | Enterprise application |

**What is multi mode project?Configurantions you need to do in parent and child project? What is dependancy management?**

Ans :: 

**What are dependency and plugin in maven?Give one example for each?**

Ans :: Both plugins and dependencies are Jar files.

But the difference between them is, most of the work in maven is done using plugins; whereas dependency is just a Jar file which will be added to the classpath while executing the tasks.

For example, you use a compiler-plugin to compile the java files. You can't use compiler-plugin as a dependency since that will only add the plugin to the classpath, and will not trigger any compilation. The Jar files to be added to the classpath while compiling the file, will be specified as a dependency.

Same goes with your scenario. You have to use spring-plugin to execute some spring executables [ I'm not sure what spring-plugins are used for. I'm just taking a guess here ]. But you need dependencies to execute those executables. And Junit is tagged under dependency since it is used by surefire-plugin for executing unit-tests.

So, we can say, plugin is a Jar file which executes the task, and dependency is a Jar which provides the class files to execute the task.

**What is distribution management?**

distributionManagement specifies where (and how) this project will get to a remote repository when it is deployed.

**How to create maven project through commandline?**

**What is the main file in Maven? (pom.xml) important configuration file in maven :settings.xml**

**What are the scope types in maven?**

**how we get back deleted data in remote repository?**

**how to call multiple pom.xml at a time using Jenkins?**

**how can we call pom.xml variable?**

**Are you involved in a maven script?**

My dev team provides required pom.xml . in the same pom.xml we will have required goals to be run. During the meeting will discuss with dev team to run required goal. Accordingly will create a Jenkins job to create artifacts.

**What is convention and configuration?**

Configuration::Developers need to create build process manually.

Convention based : Ones Developer creates a project in maven ,it’ll automatically create a structure.