**Downaload and install maven**

[**https://maven.apache.org/download.cgi**](https://maven.apache.org/download.cgi)

[**https://maven.apache.org/install.html**](https://maven.apache.org/install.html)

**Maven is from Apache**

It is automation build tool.

Maven help in building your project and documentation of project.

Maven depends on **pom.xml** file. This file has information about how to build your project.

WEB-INF -> web app

pom.xml -> Maven project (Project Object Model)

**Maven has 3 lifecycle**

* Clean
* default (build)
* site

1) The **clean** lifecycle handles project cleaning, ie cleaning of target folder

2) The **default** lifecycle handles your project deployment, i.e creation of jar or wars

3) the **site** lifecycle handles the creation of your project's web site or documentation.

**Each lifecycle has multiple phases.**

**Clean -> preClean, clean and PostClean**

**Each phase can have one or more goals (Specified by plugins)**

ie.

**Each phases has multiple goals**

**Each goal is a plugin in maven**

<https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

**Maven uses a standard directory layout and a default build lifecycle.**

Create project and show (skip archetype selection)

Maven uses **Convention** over **Configuration**, which means developers are not required to create build process themselves.

When a Maven project is created, Maven creates default project structure. Developer is only required to place files accordingly and he/she need not to define any configuration in pom.xml.

As an example, following table shows the default values for project source code files, resource files and other configurations. Assuming, **${basedir}** denotes the project location −

|  |  |
| --- | --- |
| **Item** | **Default** |
| source code | ${basedir}/src/main/java |
| Resources | ${basedir}/src/main/resources |
| Tests | ${basedir}/src/test |
| Complied byte code | ${basedir}/target/classes |
| distributable JAR | ${basedir}/target |

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<properties>

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

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

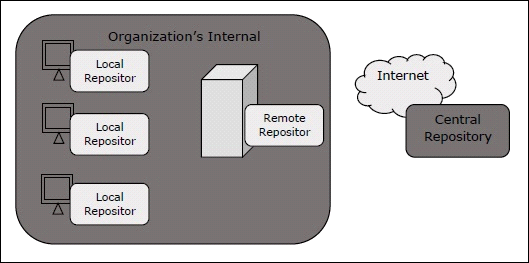
</properties>

**Maven Repository?**

In Maven terminology, a repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

Maven repository are of three types. The following illustration will give an idea regarding these three types.

* local
* central
* remote



Local Repository

Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time.

Maven local repository by default get created by Maven in %USER\_HOME% directory. To override the default location, mention another path in Maven **settings.xml** file available at %M2\_HOME%\conf directory.

<settings xmlns = "<http://maven.apache.org/SETTINGS/1.0.0>"

xmlns:xsi = "<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation = "<http://maven.apache.org/SETTINGS/1.0.0>

<http://maven.apache.org/xsd/settings-1.0.0.xsd>">

<localRepository>${user.home}/.m2/repository</localRepository>

</settings>

**Central Repository**

Maven central repository is repository provided by Maven community. It contains a large number of commonly used libraries.

When Maven does not find any dependency in local repository, it starts searching in central repository using following URL − <https://repo1.maven.org/maven2/>

Key concepts of Central repository are as follows −

* This repository is managed by Maven community.
* It is not required to be configured.
* It requires internet access to be searched.

To browse the content of central maven repository, maven community has provided a URL − [https://search.maven.org/#browse](https://search.maven.org/). Using this library, a developer can search all the available libraries in central repository.

**Remote Repository**

Sometimes, Maven does not find a mentioned dependency in central repository as well. It then stops the build process and output error message to console. To prevent such situation, Maven provides concept of **Remote Repository**, which is organization/company own custom repository containing required libraries or other project jars.

Maven Dependency Search Sequence

When we execute Maven build commands, Maven starts looking for dependency libraries in the following sequence −

* **Step 1** − Search dependency in local repository, if not found, move to step 2 else perform the further processing.
* **Step 2** − Search dependency in central repository, if not found and remote repository/repositories is/are mentioned then move to step 4. Else it is downloaded to local repository for future reference.
* **Step 3** − If a remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency).
* **Step 4** − Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference. Otherwise, Maven stops processing and throws error (Unable to find dependency).

**settings.xml** ->

1) Local repo configuration

2) From where we need to download jars (proxy).

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.15</version>

</dependency>

<!-- <https://mvnrepository.com/artifact/org.springframework/spring-aop> -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.15</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.2.16.RELEASE</version>

</dependency>

**Site lifecycle**

<build>

<plugins>

<plugin>

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

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

</plugins>

</build>

A **settings.xml** file is usually found in a couple of places:

**Global settings** in Mavens home directory: ${maven.home}/conf/settings.xml

C:\Program Files\Apache\maven\

**User settings in** the user’s home: ${user.home}/.m2/settings.xml

If both files exist, their contents are merged. Configurations from the user settings take precedence.

C:\Program Files\Apache\maven\conf

