Maven:

Maven Tutorial



Maven tutorial provides basic and advanced concepts of **apache maven** technology. Our maven tutorial is developed for beginners and professionals.

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.

Current version of Maven is 3.

Accessing Elements of Union

Keep Watching

Understanding the problem without Maven

There are many problems that we face during the project development. They are discussed below:

**1) Adding set of Jars in each project:** In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

**2) Creating the right project structure:** We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

**3) Building and Deploying the project:** We must have to build and deploy the project so that it may work.

What it does?

Maven simplifies the above mentioned problems. It does mainly following tasks.

1. It makes a project easy to build
2. It provides uniform build process (maven project can be shared by all the maven projects)
3. It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)
4. It is easy to migrate for new features of Maven

Apache Maven helps to manage

* Builds
* Documentation
* Reporing
* SCMs
* Releases
* Distribution

What is Build Tool

A build tool takes care of everything for building a process. It does following:

* Generates source code (if auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into JAR of ZIP file
* Installs the packaged code in local repository, server repository, or central repository

How to install Maven on windows

You can download and install maven on windows, linux and MAC OS platforms. Here, we are going to learn how to install maven on windows OS.

To install maven on windows, you need to perform following steps:

1. Download maven and extract it
2. Add JAVA\_HOME and MAVEN\_HOME in environment variable
3. Add maven path in environment variable
4. Verify Maven

1) Download Maven

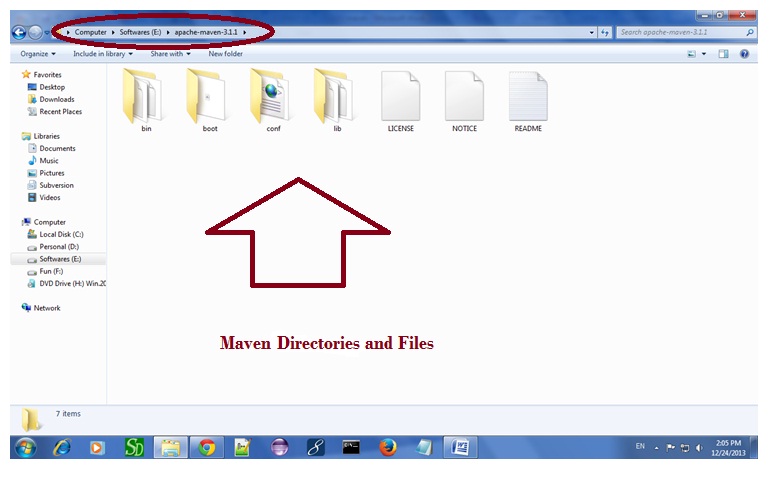
To install maven on windows, you need to download apache maven first.

Download Maven latest Maven software from [Download latest version of Maven](https://maven.apache.org/download.cgi)

[Maven – Download Apache Maven](https://maven.apache.org/download.cgi)

For example: **apache-maven-3.1.1-bin.zip**

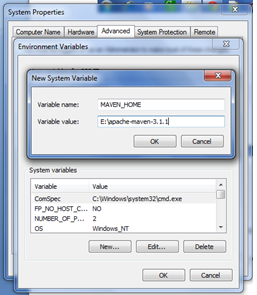
Extract it. Now it will look like this:



2) Add MAVEN\_HOME in environment variable

Right click on **MyComputer** -> **properties** -> **Advanced System Settings** -> **Environment variables** -> **click new button**

Now **add MAVEN\_HOME** in variable name and path of maven in variable value. It must be the home directory of maven i.e. outer directory of bin. For example: **E:\apache-maven-3.1.1** .It is displayed below:



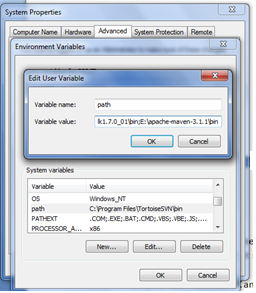
Now click on **OK** button.

3) Add Maven Path in environment variable

Click on new tab if path is not set, then set the path of maven. If it is set, edit the path and append the path of maven.

Here, we have installed JDK and its path is set by default, so we are going to append the path of maven.

The path of maven should be **%maven home%/bin**. For example, **E:\apache-maven-3.1.1\bin** .



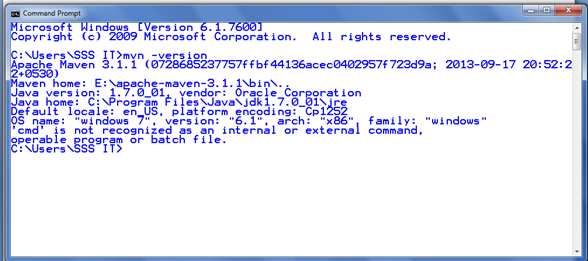
4)Verify maven

To verify whether maven is installed or not, open the command prompt and write:

1. mvn −version

Now it will display the version of maven and jdk including the maven home and java home.

Let's see the output:



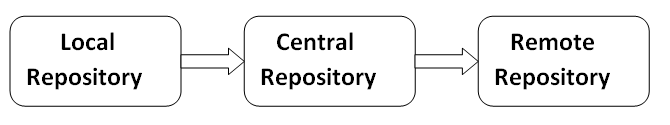
Maven Repository

A **maven repository** is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

1. Local Repository
2. Central Repository
3. Remote Repository

Maven searches for the dependencies in the following order:

**Local repository** then **Central repository** then **Remote repository**.



If dependency is not found in these repositories, maven stops processing and throws an error.

41M

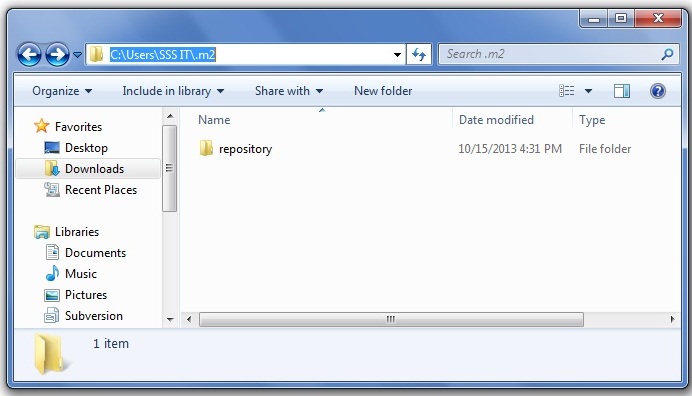
774

Difference between JDK, JRE, and JVM

1) Maven Local Repository

Maven **local repository** is located in your local system. It is created by the maven when you run any maven command.

By default, maven local repository is %USER\_HOME%/.m2 directory. For example: **C:\Users\SSS IT\.m2**.



Update location of Local Repository

We can change the location of maven local repository by changing the **settings.xml** file. It is located in **MAVEN\_HOME/conf/settings.xml**, for example: **E:\apache-maven-3.1.1\conf\settings.xml**.

Let's see the default code of settings.xml file.

*settings.xml*

1. ...
2. **<settings** xmlns="http://maven.apache.org/SETTINGS/1.0.0"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd"**>**
5. <!-- localRepository
6. | The path to the local repository maven will use to store artifacts.
7. |
8. | Default: ${user.home}/.m2/repository
9. **<localRepository>**/path/to/local/repo**</localRepository>**
10. --**>**
12. ...
13. **</settings>**

Now change the path to local repository. After changing the path of local repository, it will look like this:

*settings.xml*

1. ...
2. **<settings** xmlns="http://maven.apache.org/SETTINGS/1.0.0"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd"**>**
5. **<localRepository>**e:/mavenlocalrepository**</localRepository>**
7. ...
8. **</settings>**

As you can see, now the path of local repository is e:/mavenlocalrepository.

2) Maven Central Repository

Maven **central repository** is located on the web. It has been created by the apache maven community itself.

The path of central repository is: [http://repo1.maven.org/maven2/](https://repo1.maven.org/maven2/).

The central repository contains a lot of common libraries that can be viewed by this url [http://search.maven.org/#browse](https://search.maven.org/#browse).

3) Maven Remote Repository

Maven **remote repository** is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

Let's see the code to add the jUnit library in pom.xml 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>**

You can search any repository from Maven official website **mvnrepository.com**.

Maven pom.xml file

**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.

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. |

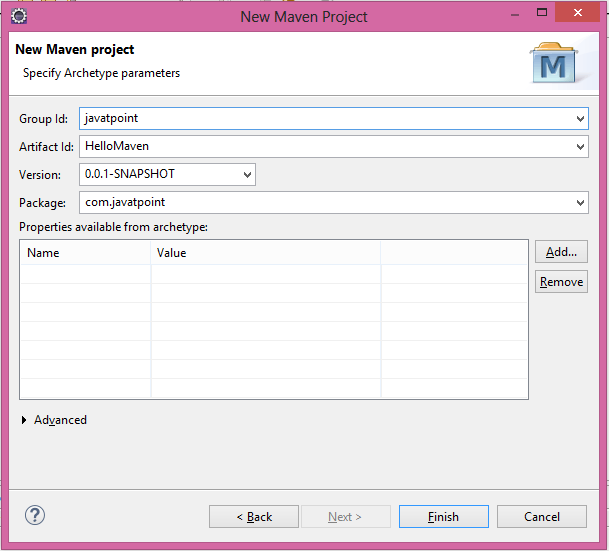
*File: pom.xml*

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>**

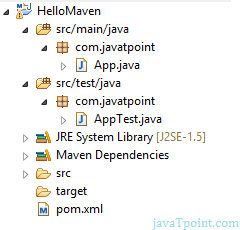
# Maven Eclipse Example

Maven eclipse tutorial explains how to create maven example in eclipse.

In eclipse, click on File menu → New → Project → Maven → Maven Project. → Next → Next → Next. Now write the group Id, artifact Id, Package as shown in below figure → finish.



Now you will see a maven project with complete directory structure. All the files will be created automatically such as Hello Java file, pom.xml file, test case file etc. The directory structure of the maven project is shown in the below figure.



Now you can see the code of App.java file and run it. It will be like the given code:

Competitive questions on Structures in HindiKeep Watching

1. **package** com.javatpoint;
2. /\*\*
3. \* Hello world!
4. \*
5. \*/
6. **public** **class** App
7. {
8. **public** **static** **void** main( String[] args )
9. {
10. System.out.println( "Hello World!" );
11. }
12. }

If you right click on the project → Run As, you will see the maven options to build the project.

