**MAVEN TUTORIAL**

**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.

**Maven Usage:**

Maven is used to build the project

It provide the project info like dependency list, unit test reports

It is easy to migrate for new features of Maven

Apache Maven, is an innovative software project management tool, provides new concept of a project object model (POM) file to manage project’s build, dependency and documentation. The most powerful feature is able to download the project dependency libraries automatically.

**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

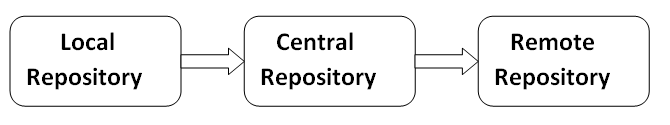
**Maven Repository:**

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.***

**Setting the 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**.

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

**Setting the 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/>.

The central repository contains a lot of common libraries that can be viewed by this url <http://search.maven.org/#browse>

**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*

[copy to clipboard](http://www.javatpoint.com/maven-repository)

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. <modelVersion>4.0.0</modelVersion>
6. <groupId>com.javatpoint.application1</groupId>
7. <artifactId>my-application1</artifactId>
8. <version>1.0</version>
9. <packaging>jar</packaging>
10. <name>Maven Quick Start Archetype</name>
11. <url>http://maven.apache.org</url>
12. <dependencies>
13. <dependency>
14. <groupId>junit</groupId>
15. <artifactId>junit</artifactId>
16. <version>4.8.2</version>
17. <scope>test</scope>
18. </dependency>
19. </dependencies>
20. </project>

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*

[copy to clipboard](http://www.javatpoint.com/maven-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">
5. <modelVersion>4.0.0</modelVersion>
6. <groupId>com.javatpoint.application1</groupId>
7. <artifactId>my-app</artifactId>
8. <version>1</version>
9. </project>

<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.javatpoint.application1</groupId>

<artifactId>my-app</artifactId>

<version>1</version>

</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*

[copy to clipboard](http://www.javatpoint.com/maven-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">
5. <modelVersion>4.0.0</modelVersion>
6. <groupId>com.javatpoint.application1</groupId>
7. <artifactId>my-application1</artifactId>
8. <version>1.0</version>
9. <packaging>jar</packaging>
10. <name>Maven Quick Start Archetype</name>
11. <url><http://maven.apache.org> </url>
12. <dependencies>
13. <dependency>
14. <groupId>junit</groupId>
15. <artifactId>junit</artifactId>
16. <version>4.8.2</version>
17. <scope>test</scope>
18. </dependency>
19. </dependencies>
20. </project>

# Maven Example

We can create a simple maven example by executing the **archetype:generate** command of **mvn tool**.

To create a simple java project using maven, you need to open command prompt and run the **archetype:generate** command of mvn tool.

#### Syntax

The **syntax** to generate the project architecture is given below:

[copy to clipboard](http://www.javatpoint.com/maven-example)

1. mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid
2. -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=booleanValue

mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=booleanValue

#### Example

The **example** to generate the project architecture is given below:

[copy to clipboard](http://www.javatpoint.com/maven-example)

1. mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator
2. -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

#### Note: Here, we are using maven-archetype-quickstart to create simple maven core project. if you use maven-archetype-webapp, it will generate a simple maven web application.

#### Output

***Now it will generate following code in the command prompt:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=Cub

#### eGenerator -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=fa

#### lse

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (5 KB at 3.5 KB/se

#### c)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 0.9 KB/s

#### ec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-quickstart:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGenerator

#### [INFO] Parameter: basedir, Value: C:\Users\SSS IT

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: C:\Users\SSS IT\CubeGene

#### rator

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.913s

#### [INFO] Finished at: Thu Dec 26 16:45:18 IST 2013

#### [INFO] Final Memory: 9M/25M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=Cub

#### eGenerator -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=fa

#### lse

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (5 KB at 3.5 KB/se

#### c)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 0.9 KB/s

#### ec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-quickstart:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGenerator

#### [INFO] Parameter: basedir, Value: C:\Users\SSS IT

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: C:\Users\SSS IT\CubeGene

#### rator

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.913s

#### [INFO] Finished at: Thu Dec 26 16:45:18 IST 2013

#### [INFO] Final Memory: 9M/25M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### Generated Directory Structure

***Now go to the current directory from where you have executed the mvn command. For example: C:\Users\SSS IT\CubeGenerator. You will see that a simple java project is created that has the following directory:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### CubeGenerator

#### -src

#### --main

#### ---java

#### ----com

#### -----javatpoint

#### ------App.java

#### --test

#### ---java

#### ----com

#### -----javatpoint

#### ------AppTest.java

#### -pom.xml

#### CubeGenerator

#### -src

#### --main

#### ---java

#### ----com

#### -----javatpoint

#### ------App.java

#### --test

#### ---java

#### ----com

#### -----javatpoint

#### ------AppTest.java

#### -pom.xml

***As you can see, there are created 3 files pom.xml, App.java and AppTest.java. Let's have a quick look at these files:***

#### 1) Automatically Generated pom.xml file

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### <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/maven-v4\_0\_0.xsd">

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGenerator</artifactId>

#### <packaging>jar</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGenerator</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### </project>

#### <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/maven-v4\_0\_0.xsd">

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGenerator</artifactId>

#### <packaging>jar</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGenerator</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### </project>

#### 2) Automatically Generated App.java file

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### **package** com.javatpoint;

#### /\*\*

#### \* Hello world!

#### \*

#### \*/

#### **public** **class** App

#### {

#### **public** **static** **void** main( String[] args )

#### {

#### System.out.println( "Hello World!" );

#### }

#### }

#### package com.javatpoint;

#### /\*\*

#### \* Hello world!

#### \*

#### \*/

#### public class App

#### {

#### public static void main( String[] args )

#### {

#### System.out.println( "Hello World!" );

#### }

#### }

#### 3) Automatically Generated AppTest.java file

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### **package** com.javatpoint;

#### **import** junit.framework.Test;

#### **import** junit.framework.TestCase;

#### **import** junit.framework.TestSuite;

#### /\*\*

#### \* Unit test for simple App.

#### \*/

#### **public** **class** AppTest

#### **extends** TestCase

#### {

#### /\*\*

#### \* Create the test case

#### \*

#### \* @param testName name of the test case

#### \*/

#### **public** AppTest( String testName )

#### {

#### **super**( testName );

#### }

#### /\*\*

#### \* @return the suite of tests being tested

#### \*/

#### **public** **static** Test suite()

#### {

#### **return** **new** TestSuite( AppTest.**class** );

#### }

#### /\*\*

#### \* Rigourous Test :-)

#### \*/

#### **public** **void** testApp()

#### {

#### assertTrue( **true** );

#### }

#### }

#### package com.javatpoint;

#### import junit.framework.Test;

#### import junit.framework.TestCase;

#### import junit.framework.TestSuite;

#### /\*\*

#### \* Unit test for simple App.

#### \*/

#### public class AppTest

#### extends TestCase

#### {

#### /\*\*

#### \* Create the test case

#### \*

#### \* @param testName name of the test case

#### \*/

#### public AppTest( String testName )

#### {

#### super( testName );

#### }

#### /\*\*

#### \* @return the suite of tests being tested

#### \*/

#### public static Test suite()

#### {

#### return new TestSuite( AppTest.class );

#### }

#### /\*\*

#### \* Rigourous Test :-)

#### \*/

#### public void testApp()

#### {

#### assertTrue( true );

#### }

#### }

## *Compile the Maven Java Project*

***To compile the project, go to the project directory,***

***for example: C:\Users\SSS IT\CubeGenerator and write the following command on the command prompt:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### mvn clean compile

#### mvn clean compile

***Now, you will see a lot of execution on the command prompt. If you check your project directory, target directory is created that contains the class files.***

## *Run the Maven Java Project*

***To run the project, go to the project directory\target\classes,***

***for example: C:\Users\SSS IT\CubeGenerator\target\classes and write the following command on the command prompt:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### java com.javatpoint.App

#### java com.javatpoint.App

***Now, you will see the output on the command prompt:***

## *Output of the maven example*

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### Hello World!

***Hello World!***

## *How to build the maven project or how to package maven project?*

***The mvn package command completes the build life cycle of the maven project such as:***

#### validate

#### compile

#### test

#### package

#### integration-test

#### verify

#### install

#### deploy

***Visit this link to know more about build life cycle*** [***http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html***](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)

***You need to execute the following command on the command prompt to package the maven project:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### mvn **package**

***mvn package***

***Now you will see that a jar file is created inside the project/target directory.***

***You can also run the maven project by the jar file. To do so, go to the maven project directory, for example: C:\Users\SSS IT\CubeGenerator and execute the following command on the cmd:***

#### [copy to clipboard](http://www.javatpoint.com/maven-example)

#### java -classpath target\CubeGenerator-1.0-SNAPSHOT.jar;.; com.javatpoint.App

**OUTPUT:**

***java -classpath target\CubeGenerator-1.0-SNAPSHOT.jar;.; com.javatpoint.App***

#### Hello World!

# Maven Web Application

We can create a simple maven web application example by executing the **archetype:generate** command of **mvn tool**.

To create a simple java project using maven, you need to open command prompt and run the **archetype:generate** command of mvn tool.

#### Syntax

The **syntax** to generate the project architecture is given below:

[copy to clipboard](http://www.javatpoint.com/maven-web-application)

1. mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid
2. -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=booleanValue

mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=booleanValue

#### Example

The **example** to generate the project architecture is given below:

[copy to clipboard](http://www.javatpoint.com/maven-web-application)

1. mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb
2. -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

#### Note: Here, we are using maven-archetype-webapp to create simple maven web application. if you use maven-archetype-quickstart, it will generate a simple maven core project.

#### Output

***Now it will generate following code in the command prompt:***

#### [copy to clipboard](http://www.javatpoint.com/maven-web-application)

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWe

#### b -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar (4 KB at 3.8 KB/sec)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom (533 B at 0.8 KB/sec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-webapp:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGeneratorWeb

#### [INFO] Parameter: basedir, Value: D:\

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: D:\CubeGeneratorWeb

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.273s

#### [INFO] Finished at: Thu Dec 26 19:25:04 IST 2013

#### [INFO] Final Memory: 10M/24M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWe

#### b -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

#### [INFO] Scanning for projects...

#### [INFO]

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Building Maven Stub Project (No POM) 1

#### [INFO] ------------------------------------------------------------------------

#### [INFO]

#### [INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>

#### >

#### [INFO]

#### [INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<

#### <

#### [INFO]

#### [INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --

#### -

#### [INFO] Generating project in Batch mode

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar (4 KB at 3.8 KB/sec)

#### Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav

#### en-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom

#### Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave

#### n-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom (533 B at 0.8 KB/sec)

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Using following parameters for creating project from Old (1.x) Archetype:

#### maven-archetype-webapp:1.0

#### [INFO] -------------------------------------------------------------------------

#### ---

#### [INFO] Parameter: groupId, Value: com.javatpoint

#### [INFO] Parameter: packageName, Value: com.javatpoint

#### [INFO] Parameter: package, Value: com.javatpoint

#### [INFO] Parameter: artifactId, Value: CubeGeneratorWeb

#### [INFO] Parameter: basedir, Value: D:\

#### [INFO] Parameter: version, Value: 1.0-SNAPSHOT

#### [INFO] project created from Old (1.x) Archetype in dir: D:\CubeGeneratorWeb

#### [INFO] ------------------------------------------------------------------------

#### [INFO] BUILD SUCCESS

#### [INFO] ------------------------------------------------------------------------

#### [INFO] Total time: 10.273s

#### [INFO] Finished at: Thu Dec 26 19:25:04 IST 2013

#### [INFO] Final Memory: 10M/24M

#### [INFO] ------------------------------------------------------------------------

#### 'cmd' is not recognized as an internal or external command,

#### operable program or batch file.

#### Generated Directory Structure

***Now go to the current directory from where you have executed the mvn command. For example: d:\CubeGeneratorWeb. You will see that a simple java project is created that has the following directory:***

#### [copy to clipboard](http://www.javatpoint.com/maven-web-application)

#### CubeGenerator

#### -src

#### --main

#### ---resources

#### ---webapp

#### ----WEB-INF

#### -----web.xml

#### ----index.jsp

#### -pom.xml

#### CubeGenerator

#### -src

#### --main

#### ---resources

#### ---webapp

#### ----WEB-INF

#### -----web.xml

#### ----index.jsp

#### -pom.xml

***As you can see, there are created 3 files pom.xml, index.jsp and web.xml. Let's have a quick look at these files:***

#### 1) Automatically Generated pom.xml file

#### [copy to clipboard](http://www.javatpoint.com/maven-web-application)

#### <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/maven-v4\_0\_0.xsd">

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGeneratorWeb</artifactId>

#### <packaging>war</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGeneratorWeb Maven Webapp</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### <build>

#### <finalName>CubeGeneratorWeb</finalName>

#### </build>

#### </project>

#### <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/maven-v4\_0\_0.xsd">

#### <modelVersion>4.0.0</modelVersion>

#### <groupId>com.javatpoint</groupId>

#### <artifactId>CubeGeneratorWeb</artifactId>

#### <packaging>war</packaging>

#### <version>1.0-SNAPSHOT</version>

#### <name>CubeGeneratorWeb Maven Webapp</name>

#### <url>http://maven.apache.org</url>

#### <dependencies>

#### <dependency>

#### <groupId>junit</groupId>

#### <artifactId>junit</artifactId>

#### <version>3.8.1</version>

#### <scope>test</scope>

#### </dependency>

#### </dependencies>

#### <build>

#### <finalName>CubeGeneratorWeb</finalName>

#### </build>

#### </project>

#### 2) Automatically Generated index.jsp file

#### [copy to clipboard](http://www.javatpoint.com/maven-web-application)

#### <html>

#### <body>

#### <h2>Hello World!</h2>

#### </body>

#### </html>

#### <html>

#### <body>

#### <h2>Hello World!</h2>

#### </body>

#### </html>

#### 3) Automatically Generated web.xml file

#### [copy to clipboard](http://www.javatpoint.com/maven-web-application)

#### <!DOCTYPE web-app PUBLIC

#### "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

#### "http://java.sun.com/dtd/web-app\_2\_3.dtd" >

#### <web-app>

#### <display-name>Archetype Created Web Application</display-name>

#### </web-app>

#### <!DOCTYPE web-app PUBLIC

#### "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

#### "http://java.sun.com/dtd/web-app\_2\_3.dtd" >

#### <web-app>

#### <display-name>Archetype Created Web Application</display-name>

#### </web-app>

## *Deploy and Run the Maven Web Project*

***Now you need to deploy the project on the server and access it by the following url:***

***http://<host-name>:<portnumber>/projectname, for example: http://localhost:8888/CubeGeneratorWeb***

#### maven webapp output

## *Maven Webapp in Eclipse*

***You can import the maven web project in eclipse. To do so, perform following steps:***

***1) Open eclipse IDE***

***2) Import the maven project***

***File Menu -> Import -> Maven -> Existing Maven Projects***

#### maven webapp eclipse import

***-> Next -> Browse Project***

#### maven webapp eclipse project

***-> Finish.***

***3) Run the maven web project***

***Right click on project -> Run As -> Run on Server***

## *Directory Structure of Maven Webapp in Eclipse*

#### maven webapp eclipse directory structure

#### **Next Topic**[Maven Plugin](http://www.javatpoint.com/maven-plugin)

Maven dependency mechanism, how it works

By [mkyong](http://www.mkyong.com/author/mkyong/) | November 26, 2009 | Updated : December 13, 2012

Maven’s dependency mechanism help to download all the necessary dependency libraries automatically, and maintain the version upgrade as well.

Case study

Let see a case study to understand how it works. Assume you want to use Log4J as your project logging mechanism. Here is what you do…

1. In traditional way

1. Visit <http://logging.apache.org/log4j/>
2. Download the Log4j jar library
3. Copy jar to project classpath
4. Include it into your project dependency manually
5. All manages by yourself, you need to do everything

If there is a Log4j version upgrade, you need to repeat above steps again.

2. In Maven way

1. You need to know the log4j Maven coordinates, for example
2. <groupId>log4j</groupId>
3. <artifactId>log4j</artifactId>
4. <version>1.2.14</version>

It will download the log4j version 1.2.14 library automatically. If the “version” tag is ignored, it will upgrade the library automatically when there is a newer version.

1. Declares Maven coordinates into pom.xml file.
2. <dependencies>
3. <dependency>
4. <groupId>log4j</groupId>
5. <artifactId>log4j</artifactId>
6. <version>1.2.14</version>
7. </dependency>
8. </dependencies>
9. When Maven is compiling or building, the log4j jar will be downloaded automatically and put it into your Maven local repository.
10. All manages by Maven.

Explanation

See the different? So what just happened in Maven? When you build a Maven’s project, the pom.xml file will be parsed, if it see the log4j Maven coordinate, then Maven search the log4j library in this order :

1. Search log4j in Maven local repository.
2. Search log4j in Maven central repository.
3. Search log4j in Maven remote repository (if defined in pom.xml).

This Maven dependency library management is a very nice tool, and save you a lot of work.

How to include custom library into maven local repository?

By mkyong | May 12, 2009 | Updated : December 13, 2012

There are 2 cases that you need to issue Maven’s command to include a jar into the Maven local repository manually.

1. The jar you want to use doesn’t exist in the Maven center repository.
2. You created a custom jar, and need to use for another Maven project.

**P.S Trust me, there are still many jars that doesn’t support Maven.**

Case study

For example, [kaptcha](http://code.google.com/p/kaptcha/), a popular third party Java library, which is used to generate “captcha” image to stop spam, but it’s not available in the Maven center repository.

In this tutorial, we will show you how to install the “kaptcha” jar into your Maven’s local repository.

1. mvn install

Download the “[kaptcha](http://code.google.com/p/kaptcha/downloads/list)“, extract it and copy the kaptcha-version.jar to somewhere else, for example, c drive. Issue following command :

mvn install:install-file -Dfile=c:\kaptcha-{version}.jar -DgroupId=com.google.code

-DartifactId=kaptcha -Dversion={version} -Dpackaging=jar

Demo.

D:\>mvn install:install-file -Dfile=c:\kaptcha-2.3.jar -DgroupId=com.google.code

-DartifactId=kaptcha -Dversion=2.3 -Dpackaging=jar

[INFO] Scanning for projects...

[INFO] Searching repository for plugin with prefix: 'install'.

[INFO] ------------------------------------------------------------------------

[INFO] Building Maven Default Project

[INFO] task-segment: [install:install-file] (aggregator-style)

[INFO] ------------------------------------------------------------------------

[INFO] [install:install-file]

[INFO] Installing c:\kaptcha-2.3.jar to

D:\maven\_repo\com\google\code\kaptcha\2.3\kaptcha-2.3.jar

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESSFUL

[INFO] ------------------------------------------------------------------------

[INFO] Total time: < 1 second

[INFO] Finished at: Tue May 12 13:41:42 SGT 2009

[INFO] Final Memory: 3M/6M

[INFO] ------------------------------------------------------------------------

Now, the "**kaptcha**" jar is copied to your Maven local repository.

2. pom.xml

After installed, just declares the kaptcha coordinate in pom.xml.

<dependency>

<groupId>com.google.code</groupId>

<artifactId>kaptcha</artifactId>

<version>2.3</version>

</dependency>

3. Done

Build it, now the "**kaptcha**" jar is able to retrieve from your Maven local repository.