Chapter 4 - Maven

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 much more advanced than ANT.

Current version of Maven is 3.

Understanding the problem without Maven

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

1) Adding a set of Jars in each project: In case of struts, spring, hibernate frameworks, we need to add a 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 is Maven?

Maven simplifies the above mentioned problems. It does mainly the 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

Difference between Ant and Maven

Ant and **Maven** both are build tools provided by Apache. The main purpose of these technologies is to ease the build process of a project.

There are many differences between ant and maven that are given below:

Ant	Maven
Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file.	Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order.	Maven is **declarative**, everything you define in the pom.xml file.
There is **no life cycle** in Ant.	There is **life cycle** in Maven.
It is **a tool** box.	It is **a framework**.
It is **mainly a build tool**.	It is **mainly a project management tool**.
The ant scripts are **not reusable**.	The maven plugins are **reusable**.
It is **less preferred** than Maven.	It is **more preferred** than Ant.

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

---

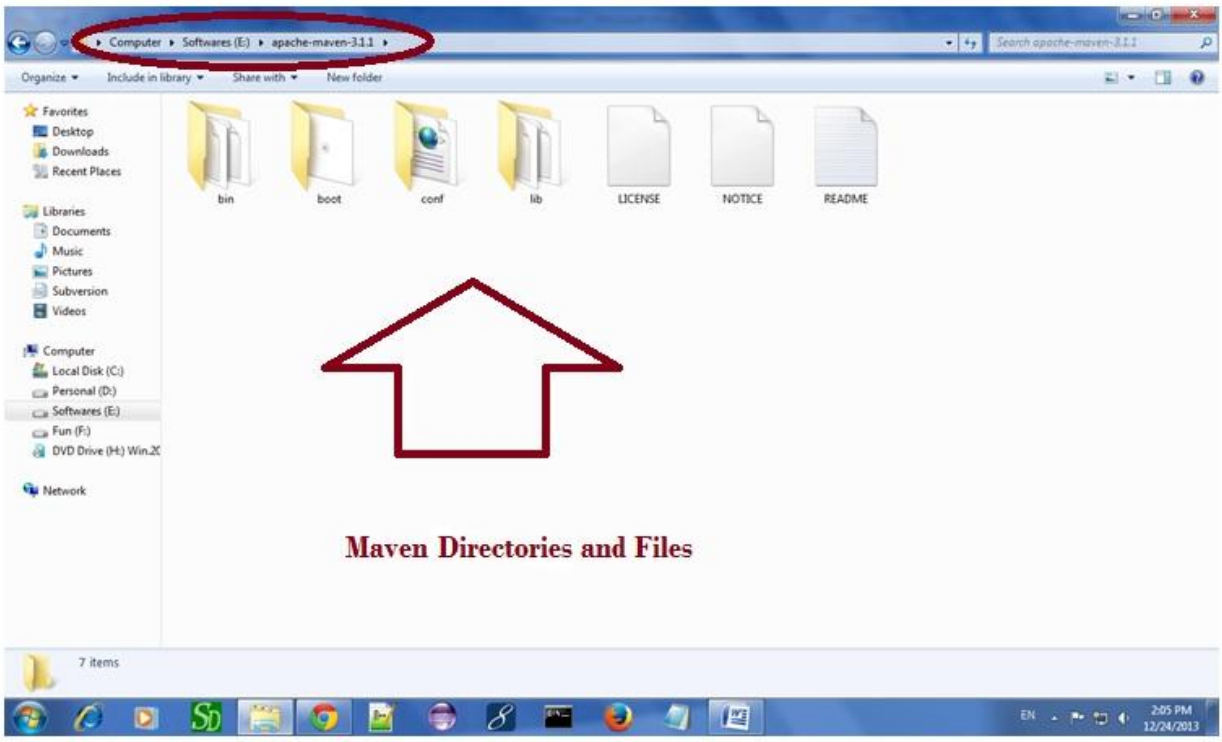
1) Download Maven

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

Download Maven latest Maven software from Download latest version of Maven

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

Extract it. Now it will look like this:

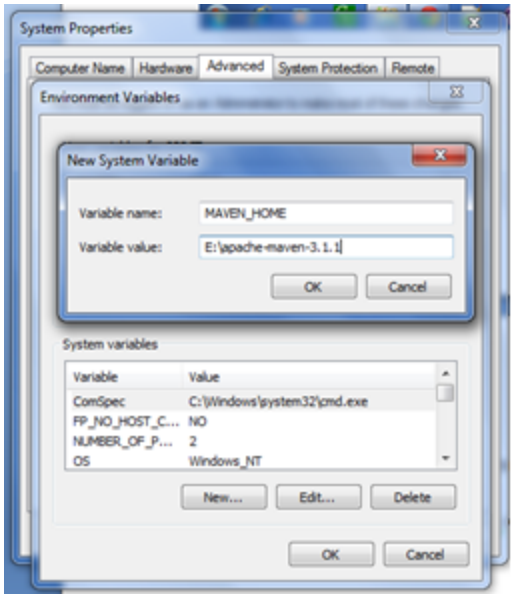


---

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.

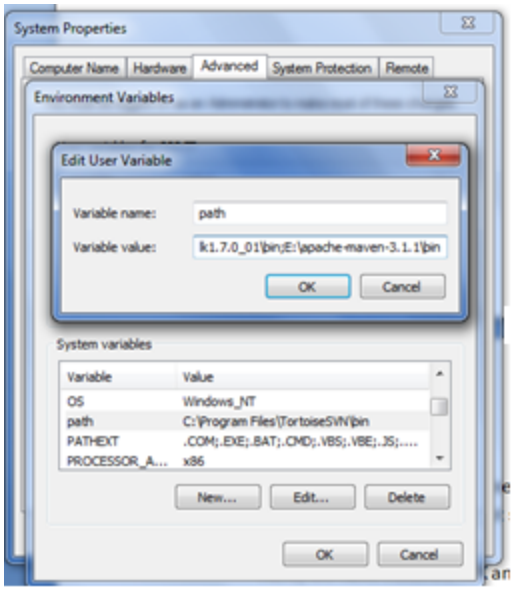
---

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** .



---

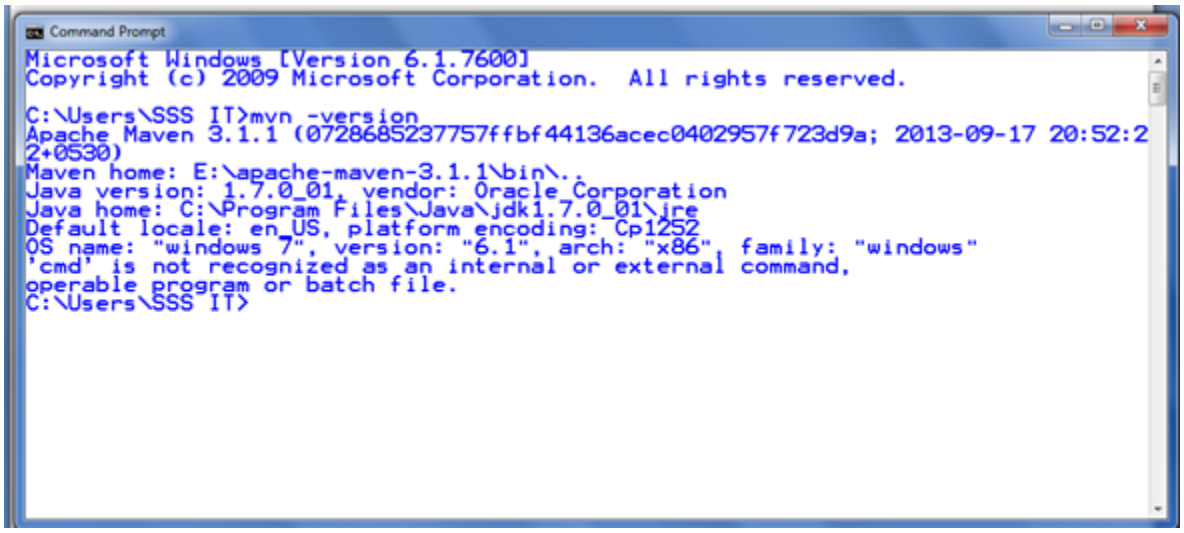
4)Verify maven

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:



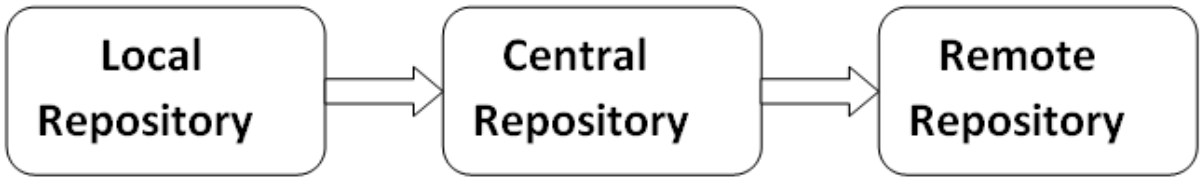
Maven Repository

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**.



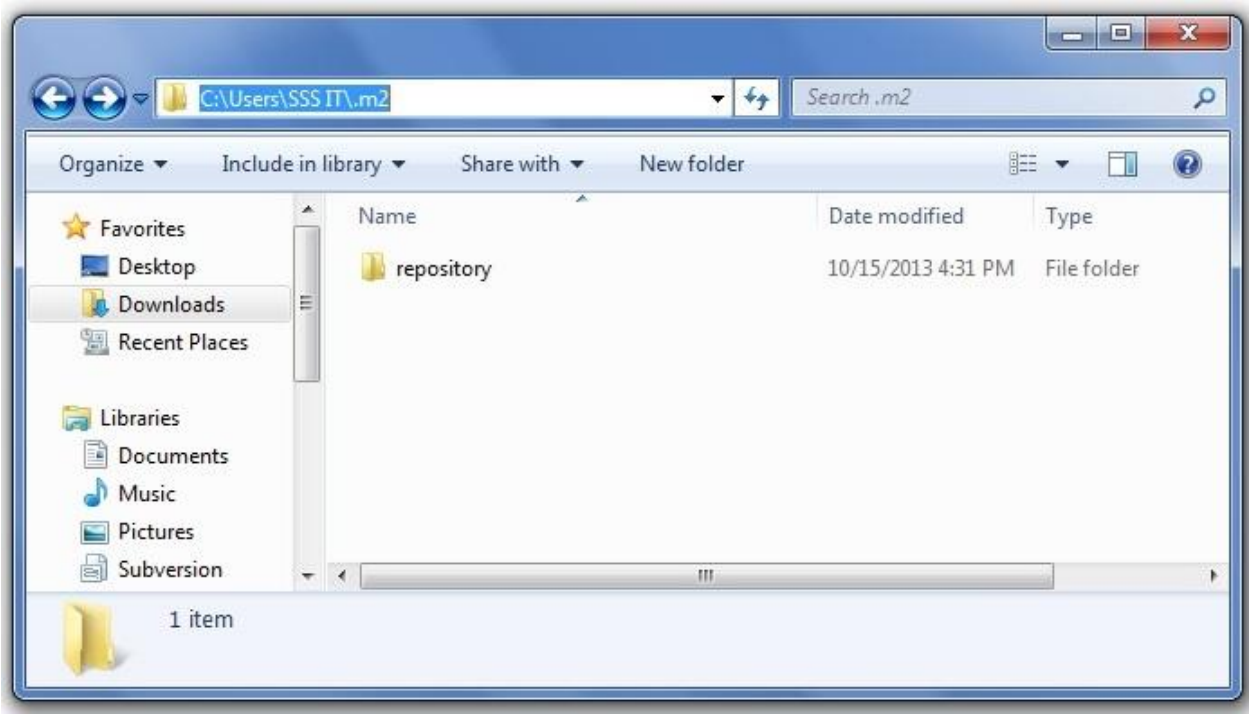
If dependency is not found in these repositories, maven stops processing and throws an error.

---

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

```
...
<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
  | The path to the local repository maven will use to store artifacts.
  |
  | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
```

```
  -->

...

</settings>
```

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

settings.xml

```
...

<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>e:/mavenlocalrepository</localRepository>


...

</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/.

The central repository contains a lot of common libraries that can be viewed by this url http://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

```
<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-application1</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
```

```
</dependencies>
</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

```
<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

```
<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-application1</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>4.8.2</version>
```

```
    <scope>test</scope>
  </dependency>
 </dependencies>

</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:

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

Example

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

```
1. mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator
2. -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**:

```
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:

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

```
<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

```
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

```
package com.javatpoint;

import junit.framework.Test;
```

```java
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 );
    }
}
```