Maven - Creating Project

Maven uses **archetype** plugins to create projects. To create a simple java application, we'll use maven-archetype-quickstart plugin. In example below, We'll create a maven based java application project in C:\MVN folder.

Let's open command console, go the C:\MVN directory and execute the following **mvn** command.

C:\MVN>mvn archetype:generate

-DgroupId=com.companyname.bank

-DartifactId=consumerBanking

-DarchetypeArtifactId=maven-archetype-quickstart

-DinteractiveMode=false

Maven will start processing and will create the complete java application project structure.

INFO] Scanning for projects...

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

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

[INFO] Building Maven Default Project

[INFO] task-segment: [archetype:generate] (aggregator-style)

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

[INFO] Preparing archetype:generate

[INFO] No goals needed for project - skipping

[INFO] [archetype:generate {execution: default-cli}]

[INFO] Generating project in Batch mode

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

[INFO] Using following parameters for creating project

from Old (1.x) Archetype: maven-archetype-quickstart:1.0

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

[INFO] Parameter: groupId, Value: com.companyname.bank

[INFO] Parameter: packageName, Value: com.companyname.bank

[INFO] Parameter: package, Value: com.companyname.bank

[INFO] Parameter: artifactId, Value: consumerBanking

[INFO] Parameter: basedir, Value: C:\MVN

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

[INFO] project created from Old (1.x) Archetype in dir: C:\MVN\consumerBanking

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

[INFO] BUILD SUCCESSFUL

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

[INFO] Total time: 14 seconds

[INFO] Finished at: Tue Jul 10 15:38:58 IST 2012

[INFO] Final Memory: 21M/124M

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

Now go to C:/MVN directory. You'll see a java application project created named consumerBanking (as specified in artifactId). Maven uses a standard directory layout as shown below:



Using above example, we can understand following key concepts

|  |  |
| --- | --- |
| **Folder Structure** | **Description** |
| consumerBanking | contains src folder and pom.xml |
| src/main/java | contains java code files under the package structure (com/companyName/bank). |
| src/main/test | contains test java code files under the package structure (com/companyName/bank). |
| src/main/resources | it contains images/properties files (In above example, we need to create this structure manually). |

If you see, Maven also created a sample Java Source file and Java Test file. Open C:\MVN\consumerBanking\src\main\java\com\companyname\bank folder, you will see App.java.

package com.companyname.bank;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

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

}

}

Open C:\MVN\consumerBanking\src\test\java\com\companyname\bank folder, you will see AppTest.java.

package com.companyname.bank;

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 );

}

}

Developers are required to place their files as mentioned in table above and Maven handles the all the build related complexities.

What we learnt in Project Creation chapter is how to create a Java application using Maven. Now we'll see how to build and test the application.

Go to C:/MVN directory where you've created your java application. Open *consumerBanking* folder.You will see the **POM.xml** file with following contents.

<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.companyname.projectgroup</groupId>

<artifactId>project</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

</dependency>

</dependencies>

</project>

Here you can see, Maven already added Junit as test framework. By default Maven adds a source file **App.java** and a test file **AppTest.java** in its default directory structure discussed in previous chapter.

Let's open command console, go the C:\MVN\consumerBanking directory and execute the following **mvn** command.

C:\MVN\consumerBanking>mvn clean package

Maven will start building the project.

[INFO] Scanning for projects...

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

[INFO] Building consumerBanking

[INFO] task-segment: [clean, package]

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

[INFO] [clean:clean {execution: default-clean}]

[INFO] Deleting directory C:\MVN\consumerBanking\target

[INFO] [resources:resources {execution: default-resources}]

[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,

i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory C:\MVN\consumerBanking\src\main\

resources

[INFO] [compiler:compile {execution: default-compile}]

[INFO] Compiling 1 source file to C:\MVN\consumerBanking\target\classes

[INFO] [resources:testResources {execution: default-testResources}]

[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,

i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory C:\MVN\consumerBanking\src\test\

resources

[INFO] [compiler:testCompile {execution: default-testCompile}]

[INFO] Compiling 1 source file to C:\MVN\consumerBanking\target\test-classes

[INFO] [surefire:test {execution: default-test}]

[INFO] Surefire report directory: C:\MVN\consumerBanking\target\

surefire-reports

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running com.companyname.bank.AppTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.027 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] [jar:jar {execution: default-jar}]

[INFO] Building jar: C:\MVN\consumerBanking\target\

consumerBanking-1.0-SNAPSHOT.jar

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

[INFO] BUILD SUCCESSFUL

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

[INFO] Total time: 2 seconds

[INFO] Finished at: Tue Jul 10 16:52:18 IST 2012

[INFO] Final Memory: 16M/89M

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

You've built your project and created final jar file, following are the key learning concepts

* We give maven two goals, first to clean the target directory (clean) and then package the project build output as jar(package).
* Packaged jar is available in consumerBanking\target folder as consumerBanking-1.0-SNAPSHOT.jar.
* Test reports are available in consumerBanking\target\surefire-reports folder.
* Maven compiled source code file(s) and then test source code file(s).
* Then Maven run the test cases.
* Finally Maven created the package.

Now open command console, go the C:\MVN\consumerBanking\target\classes directory and execute the following java command.

C:\MVN\consumerBanking\target\classes>java com.companyname.bank.App

You will see the result

Hello World!

Adding Java Source Files

Let's see how we can add additional Java files in our project. Open C:\MVN\consumerBanking\src\main\java\com\companyname\bank folder, create Util class in it as Util.java.

package com.companyname.bank;

public class Util

{

public static void printMessage(String message){

System.out.println(message);

}

}

Update App class to use Util class.

package com.companyname.bank;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

Util.printMessage("Hello World!");

}

}

Now open command console, go the C:\MVN\consumerBanking directory and execute the following **mvn** command.

C:\MVN\consumerBanking>mvn clean compile

After Maven build is successful, go the C:\MVN\consumerBanking\target\classes directory and execute the following java command.

C:\MVN\consumerBanking\target\classes>java -cp com.companyname.bank.App

You will see the result

Hello World!