

Instructor Notes:



Instructor Notes:

Explain the lesson coverage

Lesson Objectives



- In this lesson, you will learn about:
 - Installation and setup Maven
 - Download and Install Maven
 - Configure settings.xml file
- Creating your first project using Maven commands
 - Creating Project Template
 - Compiling project
 - Testing the sources

Instructor Notes:

3.1.1 Download and Install Maven

3.1 Installation and Setup Maven

- Download Maven from Website:
- <http://maven.apache.org/download.html>
- Unzip the distribution archive into the directory you wish to install Maven.
- Set these environment variables:
 - Maven installation directory:
 - M2_HOME=/usr/local/apache-maven/apache-maven-3.3.9.
 - Java installation directory:
 - JAVA_HOME=/usr/java/jdk1.8.0_31
 - Set the path to Java and Maven Home environment variables:
 - path:
 - path =%path%;JAVA_HOME/bin;M2_HOME/bin;

Capgemini 

Instructor Notes:

The same configuration can be made in the existing settings.xml file in apache maven home directory. For Example settings can be configured in settings.xml file present in the path C:\apache-maven-2.2.1\conf

3.1.2 Configuring Settings.xml file

3.1 Installation and Setup Maven



- Configure settings.xml file with the following content:

```
<settings>
  <proxies>
    <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.company.com</host>
      <port>8080</port>
      <username>your-username</username>
      <password>your-password</password>
    </proxy>
  </proxies>
</settings>
```

Capgemini

If you are behind a firewall, then you will have to set up Maven to understand and download required plugins from repository. To do this, create a <your-home directory>/.m2/settings.xml file with the following content:

For Example:

```
<settings>
<proxies>
  <proxy>
    <id>optional</id>
    <active>true</active>
    <protocol>http</protocol>
    <username>username</username>
    <password>password</password>
    <host>192.168.104.40.org.com</host>
    <port>8080</port>
    <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
  </proxy>
</proxies>
</settings>
```

Instructor Notes:

3.1.3 Testing Maven Installation

3.1 Installation and Setup Maven

- Testing version of Maven:
mvn -version
 - Gives information of Maven version & environment being used

```
C:\Users>mvn -version
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T22:11:47+05:30)
Maven home: D:\Softwares\apache-maven-3.3.9
Java version: 1.8.0_31, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_31\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "x86", family: "dos"
```

Capgemini 

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project



- Archetype mechanism used to create first Maven Project.
- Archetype is a Maven project templating toolkit which is combined with some user input to produce a fully functional Maven project.
- To create Maven project execute the following command and follow the instructions:
 - `mvn archetype:generate`
 - The full list of archetypes will be displayed
 - Choose a number for selecting an archetype and provide the project information.(groupid, artifactId,...)
- Example:
 - `mvn archetype:generate`
 - `DarchetypeGroupId=org.apache.maven.archetypes`
 - `DgroupId=com.mycompany.app -DartifactId=my-app`



“An archetype is defined as an original pattern or model from which all other things of the same kind are made. “

You may want to standardize J2EE development within your organization so you may want to provide archetypes for EJBs, or WARs, or for your web services. Once these archetypes are created and deployed in your organization's repository they are available for use by all developers within your organization.

Archetypes are packaged up in a JAR and they consist of the archetype metadata which describes the contents of archetype and a set of Velocity templates which make up the prototype project.

To create Maven project execute the following command and follow the instructions:

```
mvn archetype:generate
```

If this is the first time you have run the archetype plugin, you will notice that Maven spends some time downloading the archetype functionality as well as the other Java libraries that it uses. These files are stored in your local repository for re-use in the future.

Once the downloads finish from the earlier archetype command, you will be prompted (with a long list!) to select an archetype to generate the project from.

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project

```
D:\maven-demo>mvn archetype:generate -DgroupId=com.capgemini.app -DartifactId=my
-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----
[INFO]
[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources
@ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources
@ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
```



Among the many choices, the default is the quick start archetype that is used in Maven's getting started guide. This is a good choice for a minimal Maven project or an archetype from which new modules can be built (particularly for JAR projects).

Each Maven project needs some coordinates to allow it to be identified by other Maven projects. These are called the group ID, artifact ID, and version (sometimes referred to as the GAV).

Group ID: An identifier for a collection of related modules. This is usually a hierarchy that starts with the organization that produced the modules, and then possibly moves into the increasingly more specialized project or sub-projects that the artifacts are a part of. This can also be thought of as a namespace, and is structured much like the Java package system.

Artifact ID: The Artifact ID is a unique identifier for a given module within a group.

Version: The version is used to identify the release or build number of the project.

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project



```
[INFO] Parameter: basedir, Value: D:\maven-demo
[INFO] Parameter: package, Value: com.capgemini.app
[INFO] Parameter: groupId, Value: com.capgemini.app
[INFO] Parameter: artifactId, Value: my-app
[INFO] Parameter: packageName, Value: com.capgemini.app
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: D:\maven-demo\my-app
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.301 s
[INFO] Finished at: 2016-05-13T02:18:12+05:30
[INFO] Final Memory: 10M/26M
[INFO] -----
```



In Maven, there are two types of versions: releases and snapshots (those that end in -SNAPSHOT). A snapshot should always be used by the projects during development as it has a special meaning to Maven to indicate that development is still occurring and that the project may change. A release is assumed never to change, so release versions (such as 1.1, 2.0, or 3.0-beta-5) should only be used for a single state of the project when it is released, and then updated to the next snapshot.

The final prompt for a **package** is the Java package that will be used for source code. It is not the packaging type that to take by the project as that is provided by the archetype itself.

After entering the values above, prompt will be displayed to confirm that the values are as you intended, and then the project will be created. If the below highly desirable banner displayed, then congratulations—Maven project created successfully!

```
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
```


Instructor Notes:

3.2.2 Viewing Project structure

3.2 Creating your First Project

- To view the project structure which is created based on archetype, execute the following command:
 - tree application-name

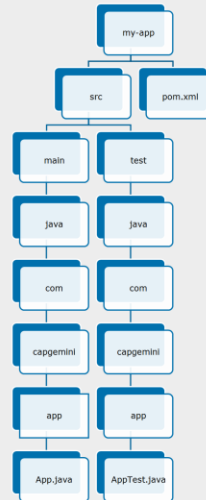
```
D:\maven-demo>tree my-app
Folder PATH listing
Volume serial number is 00000002 8A01:5839
D:\MAVEN-DEMO\MY-APP
├── src
│   ├── main
│   │   └── java
│   │       ├── com
│   │       │   └── capgemini
│   │       │       └── app
│   └── test
│       └── java
│           ├── com
│           │   └── capgemini
│           │       └── app
```

- The src/main/java directory contains the basic java application, while pom.xml specifies the information Maven needs to know.
- This directory structure has followed the Maven convention, which requires a minimal amount of configuration to construct a functional build.



Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Folder Structure

Capgemini

Instructor Notes:

3.2.3 Compiling Project

3.2 Creating your First Project

- For Compiling your application sources, execute the following command:
 - mvn compile

```
D:\maven-demo\my-app>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory D:\maven-demo\my-app\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. b
uild is platform dependent!
[INFO] Compiling 1 source file to D:\maven-demo\my-app\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 5.699 s
[INFO] Finished at: 2016-05-14T22:40:51+05:30
[INFO] Final Memory: 10M/26M
[INFO] -----
```



For Compiling your application sources, execute the following command:

- mvn compile

After compilation, the compiled classes generated in the default target directory i.e) target/classes.

The compiler plugin was used to compile the application sources, that maps to the Maven's default build life cycle.

Instructor Notes:

3.2.4 Testing Project

3.2 Creating your First Project

- After successfully compiling application's sources, testing phase in the lifecycle will be performed.
- Use the following simple command to test:
 - mvn test
- Before compiling and executing the tests, Maven compiles the main code.
- mvn test will always run the compile and test-compile phases.
 - compile : Compiles Application sources.
 - test-compile: Compile tests
- Test logs are written under target/surefire-reports.
- Debugging an application can be done by viewing the test logs.

Capgemini 

- After successfully compiling application's sources, testing phase in the lifecycle will be performed.
- Use the following simple command to test:
 - mvn test
- Maven downloads more dependencies this time. These are the dependencies and plugins necessary for executing the tests.
- Before compiling and executing the tests, Maven compiles the main code.
- mvn test will always run the compile and test-compile phases.
 - compile - Compiles the application source codes.
 - test-compile - compiles the test codes.

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project



```
C:\Windows\system32\cmd.exe
D:\maven-demo\my-app>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building mv-app 1.0-SNAPSHOT
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ my-app ---
[INFO] Surefire report directory: D:\maven-demo\my-app\target\surefire-reports
[INFO]
[INFO] -----
T E S T S
[INFO] -----
Running com.capgemini.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.015 sec
Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

Instructor Notes:

3.2.5 Packaging and Installation to Local Repository

3.2 Creating your First Project

- **Packaging an Application**
- Artifact creation can be accomplished by executing the following command:
 - `mvn package`
- Artifact will be created based on the packaging element value in the pom.xml file.
- ../target directory contains generated artifact of a project.
- **Installation of artifact into local repository:**
- The directory <home directory>/.m2/repository is the default location of the repository
- Once artifact installed, it can be used by other projects as a dependency.
- To install, execute the following command:
 - `mvn install`

Capgemini **Packaging an Application:**

- Artifact creation can be accomplished by executing the following command:
 - `mvn package`
- Artifact will be created based on the packaging element value in the pom.xml file.
- ../target directory contains generated artifact of a project.

Installation of artifact into local repository:

- The directory <home directory>/.m2/repository is the default location of the repository
- Once artifact installed, it can be used by other projects as a dependency.
- To install, execute the following command:
 - `mvn install`
- This command can be used instead most of the time, as it will process all of the previous phases of the build life cycle (generate sources, compile, compile tests, run tests, etc.).

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project



```
D:\maven-demo>cd my-app
D:\maven-demo\my-app>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory D:\maven-demo\my-app\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build
is platform dependent!
[INFO] Compiling 1 source file to D:\maven-demo\my-app\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ my
-app ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
```

Instructor Notes:

3.2.1 Archetype Mechanism

3.2 Creating your First Project



```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ my-app ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.760 s
[INFO] Finished at: 2016-05-13T02:20:58+05:30
[INFO] Final Memory: 7M/19M
[INFO] -----

D:\maven-demo\my-app>java -cp target/my-app-1.0-SNAPSHOT.jar com.capgemini.app.A
pp
Hello World!
```

Capgemini 

Instructor Notes:

3.2.6 Implementation of rich features in Maven

3.2 Creating your First Project


- Creating a simple site for a project can be done by running a command:
 - mvn site
- To remove target directory and remains project with old build data, execute the following command:
 - mvn clean
- Help command:
 - mvn help
- For an example, to view the options for the maven-compiler-plugin, use the following command:
 - mvn help:describe -DgroupId=org.apache.maven.plugins \
 - DartifactId=maven-compiler-plugin -Dfull=true

Capgemini 

Instructor Notes:

3.2.3 Generated POM.xml

3.2 Creating your First Project



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/POM/4.0.0/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.capgemini.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</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>
```

Capgemini

The archetype has substituted the coordinates that it prompted for earlier in the **pom.xml** file. It has also added the jar packaging type to indicate that the build structure for the project is a Java application. A default name has been added, which you would typically change to something which describes the project better.

Maven's declarative project model defines a number of well-known properties that the whole build can reuse for a number of different purposes. For Example, even javadoc can be produced if requested.

The POM contains important pieces of information about the project, which include:

- The Maven coordinate of the project for reuse by other Maven projects
- The project name, description and license
- Project resource information such as the location of source control, issue tracking, and continuous integration
- The developers, contributors and organizations participating in the project

The POM will also include information about how the project should be built, such as:

- The source directory layout
- Dependencies on other projects
- Build requirements (by means of Maven plugins) and configuration

Instructor Notes:

3.2.7 Demo on Project creation

3.2 Creating your First Project



- Creating and executing the "Hello World!" program

Instructor Notes:

3.2.8 Demo on Project creation

Lab: 1

- 1.1 : Getting Started With Maven
- 1.2 : Configuring Maven Settings
- 1.3 : Creating first standalone Maven application using archetype

Capgemini 

Instructor Notes:

Explain the lesson coverage

Summary



- In this lesson, you have learn about:
- Installation and setup Maven
 - Download and Install Maven
 - Configure settings.xml file
- Creating your first project using Maven commands
 - Creating Project Template
 - Compiling project
 - Testing the sources

Instructor Notes:

Answers for the Review Questions:

Question 1: mvn clean

Question 2: True

Review Question



- Question1: _____ command execution will remove target directory and remains project with old build data.
 - mvn compile
 - mvn clean
 - mvn stage
 - mvn verify
- Question2: Maven stores all the software libraries or artifacts in stores called a repository.
 - True
 - False