

Explain the lesson coverage

Lesson Objectives



- In this lesson, you will learn about:
 - Overview
 - Testing your application
- Generating Java Projects using Maven in Eclipse Luna
 - Setting Maven in Eclipse Environment
 - Adding/Updating Dependencies
 - Importing Project
 - Creating Maven Module
- Creating Web application using Maven
- Creating Maven web Module

4.1.1 Unit tests the code

4.1 Testing your application



- The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application.
- By default, these files are generated at \${basedir}/target/surefire-reports.
- The Surefire Plugin has only 1 goal:
- surefire: test runs the unit tests of an application.
- By default, the Surefire plugin executes **/Test*.java,
 **/*Test.java, and **/*TestCase.java test classes
- Use the following command to execute the unit tests:
- mvn test
- To skip the entire unit test, use "-Dmaven.test.skip=true" option in command line.
- For example,
- \$ mvn install -Dmaven.test.skip=true



It generates reports in 2 different file formats:

- Plain text files (*.txt)
- XML files (*.xml)

The Failsafe Plugin is designed to run integration tests while the Surefire Plugins is designed to run unit tests

4.1.2 Integration testing

4.1 Testing your application



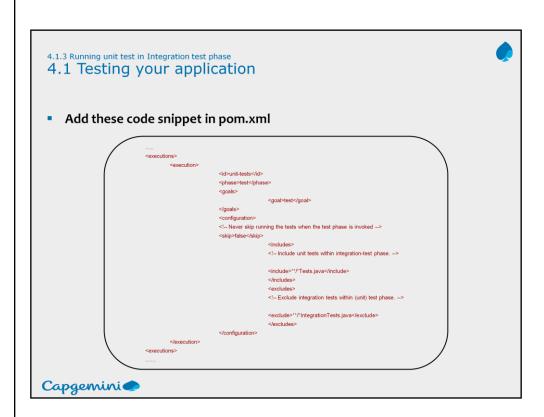
- The Failsafe Plugin is used during the integration-test and verify phases of the build lifecycle to execute the integration tests of an application
- The Maven lifecycle has four phases for running integration tests:
 - pre-integration-test for setting up the integration test environment.
 - integration-test for running the integration tests.
 - post-integration-test for tearing down the integration test environment.
 - verify for checking the results of the integration tests.
- Use the following command to run and verify the integration test.
 - mvn verify

Capgemini

The Failsafe Plugin has only 2 goals:

<u>failsafe:integration-test</u> runs the integration tests of an application. <u>failsafe:verify</u> verifies that the integration tests of an application passed.

The Failsafe plugin will look for **/IT*.java, **/*IT.java, and **/*ITCase.java.

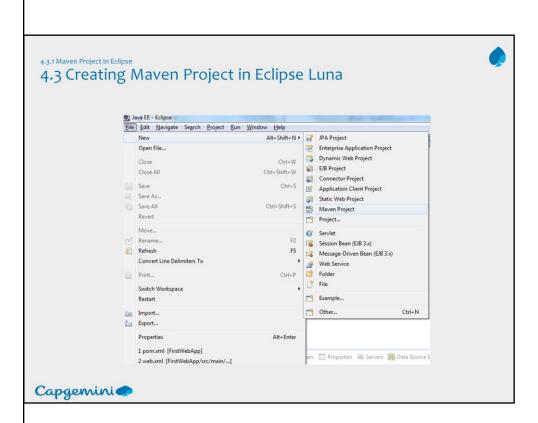


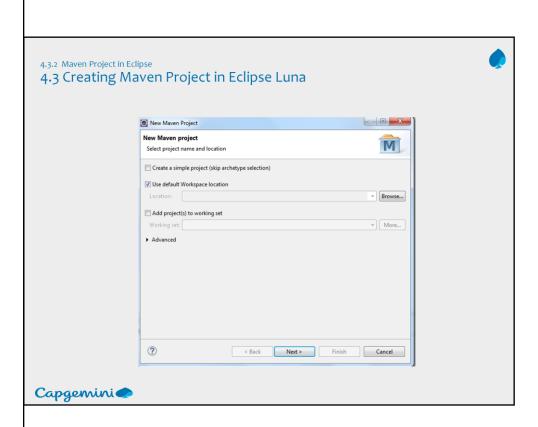
4.2.1 Starting Eclipse at JDK

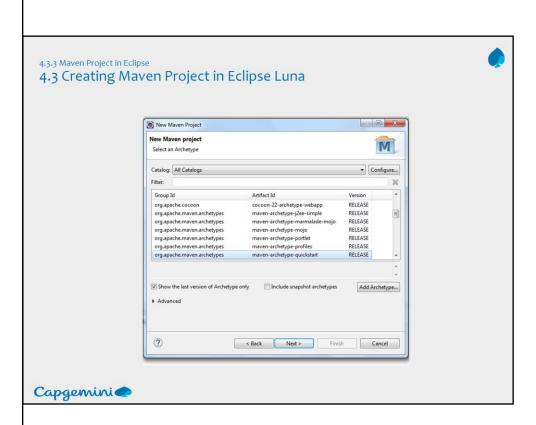


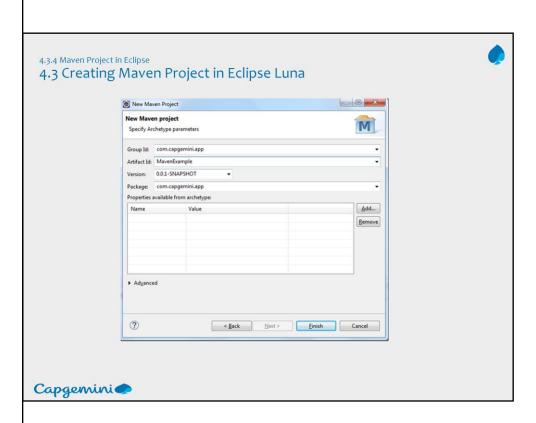
4.2 Setting Maven in Eclipse Environment

- Maven requires Eclipse using a JDK, instead of a JRE.
- To check with what Java version (JRE or JDK) Eclipse is running, do the following:
- Open the menu item "Help > About Eclipse". (On the Mac, it's in the Eclipse-menu, not the Help-menu)
- Click on "Installation Details".
- Switch to the tab "Configuration"
- Search for a line that starts with "-vm". The line following it shows which Java binary is used.

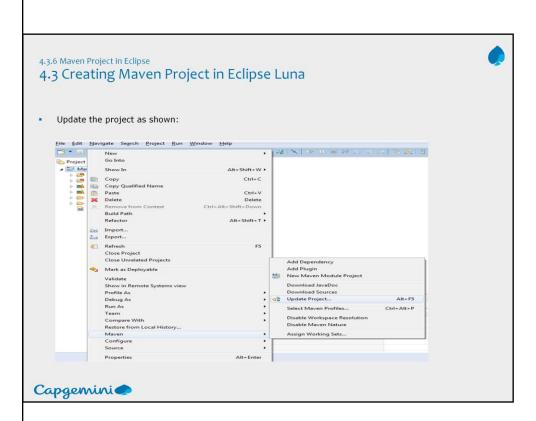












```
4.3.7 Maven Project in Eclipse
4.3 Creating Maven Project in Eclipse Luna
     Project Explorer 
☐ 🍇 😭 🌣 🗆 🗖 🚺 App.java 🖾
     1 package com.capgemini.app;
                              4 * Hello world!
5 *
6 */
7 public class App
8 {
9     public static
                                4 * Hello world!
                                      public static void main( String[] args )
                               11
                                          System.out.println( "Hello World!" );
                               12
                               13 }
                               14
                               Markers 🔲 Properties 🦚 Servers 🛍 Data Source Explorer 🔝 Snippets 📮 Console 🕄 Jtj JUnit
Capgemini
```

```
4.3.8 Maven Project in Eclipse
4.3 Creating Maven Project in Eclipse Luna
     Project Explorer 🛭 🕒 😘 💝 💆 🗖 🚺 *AppTest.java 😢

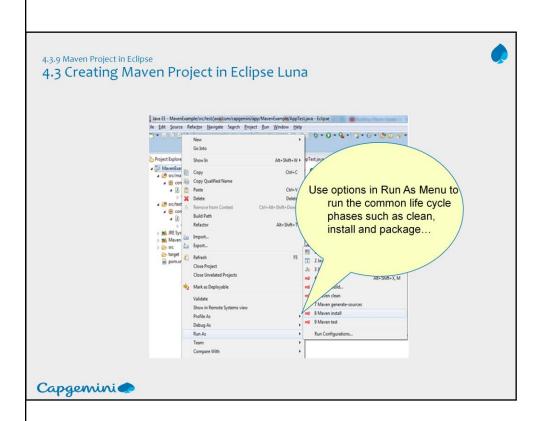
■ MavenExample

                                         1 package com.capgemini.app;
       3⊖ import static org.junit.Assert.*;
                                         5 import org.junit.Test;
                                       6 /**
8 * Unit test for simple App.
9 */
       Maven Dependencies

> Src

target

pom.xml
                                        public class AppTest
                                        12
13
                                               public void testApp() {
                                                   assertTrue(true);
                                        18
                                        19 }
Capgemini
```



4.3.10 Adding and Updating Dependencies

4.3 Creating Maven Project in Eclipse Luna

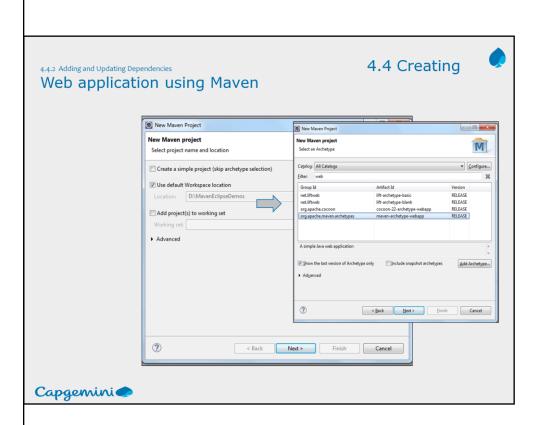


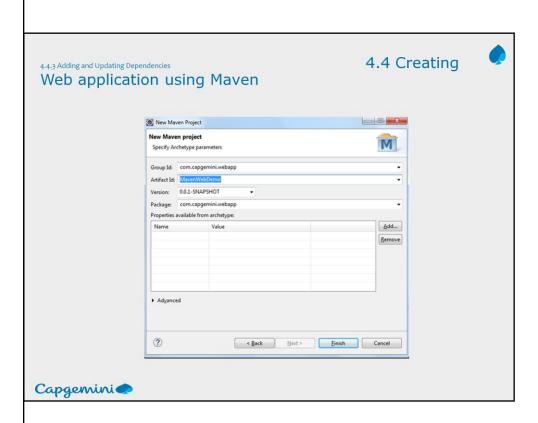
- Eclipse offers two options for adding dependencies to a project.
- By manually editing the POM file to type in the XML to add the dependency.
 - Open the pom.xml using XML Editor
 - Edit the pom by adding /updating dependency and save the file.
- Searching for a dependency using groupId
 - Open the pom.xml using Maven POM Editor
 - Click on Dependency tab to add the dependency.
 - Search the dependency by entering groupId or artifactID.
 - Eclipse queries the dependency in repository indexes and even shows a version of the artifact that is currently in local Maven repository

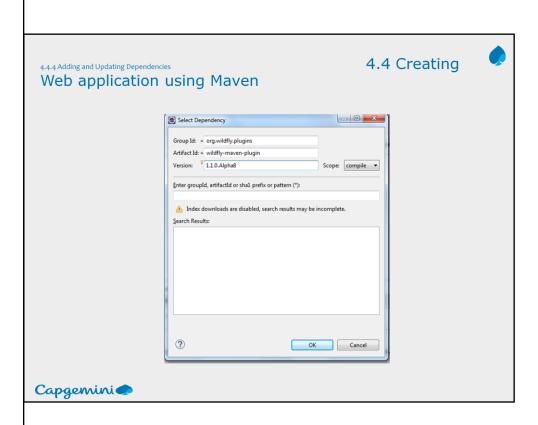


- 4.3 Creating Maven Projects in Eclipse Luna
- Creating and executing the "Maven Example!" program

4.4 4.4.1 Adding and Updating Dependencies Creating Web application using Maven Add server credentials in settings.xml <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"</pre> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 https://maven.apache.org/xsd/settings-1.0.0.xsd"> cprofile> <id>wildfly-local</id> properties> <wildfly-home>D:/Softwares/WildFly8.1.0/WildFly8.1.0</wildfly-home> <wildfly-hostname>localhost</wildfly-hostname> <wildfly-port>8090</wildfly-port> <wildfly-username>test</wildfly-username> <wildfly-password>test@123</wildfly-password> </properties> </profile> </profiles> Capgemini

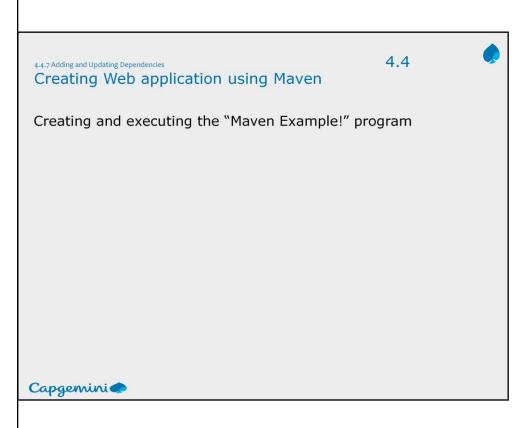






```
4.4
4.4.5 Adding and Updating Dependencies
Creating Web application using Maven
       <dependency>
           <groupId>org.wildfly.plugins
           <artifactId>wildfly-maven-plugin</artifactId>
           <exclusions>
               <exclusion>
                   <artifactId>jconsole</artifactId>
                   <groupId>sun.jdk
               </exclusion>
           </exclusions>
           <version>1.1.0.Alpha11
           <scope>runtime</scope>
       </dependency>
Capgemini
```





Capgemini

2.1: Configure environment to run Maven project
2.2: Creating a sample Maven project

Explain the lesson coverage

Summary



- In this lesson, you have learn about:
 - Overview
 - Testing your application
- Generating Java Projects using Maven
 - Using archetype
 - Creating Maven Module
- Generating Java web application Projects using Maven

Answers for the Review Questions: Question 1: True Question 2: pom.xml

Review Question



- Question 1: mvn deploy command executes all the build phases in the default lifecycle
 - True
 - False
- Question 2: Configuration of dependencies, is specified in which of the following files?
 - pom.xml
 - setting.xml
 - .m2 folder
 - test files

Answers for the Review Questions:
Question 3: Internal and Snapshot
Question 4: groupId and artifactId may be same in a project.

Review Question



- Question 3: ______repository for containing released artifacts and ______ repository for storing deployed snapshots.
- Question 4: Which of the following statements is true?
 - groupId and artifactId must be same in a project.
 - groupId and artifactId may be same in a project.
 - groupId and artifactId must not be same in a project.