Section 5.1: Maven

AN INITIATIVE BY

Introduction

- Maven is a build automation tool used primarily for Java projects.
- Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.
- The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

UNICAL ACADEMY

Installing Maven in Local Machine

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

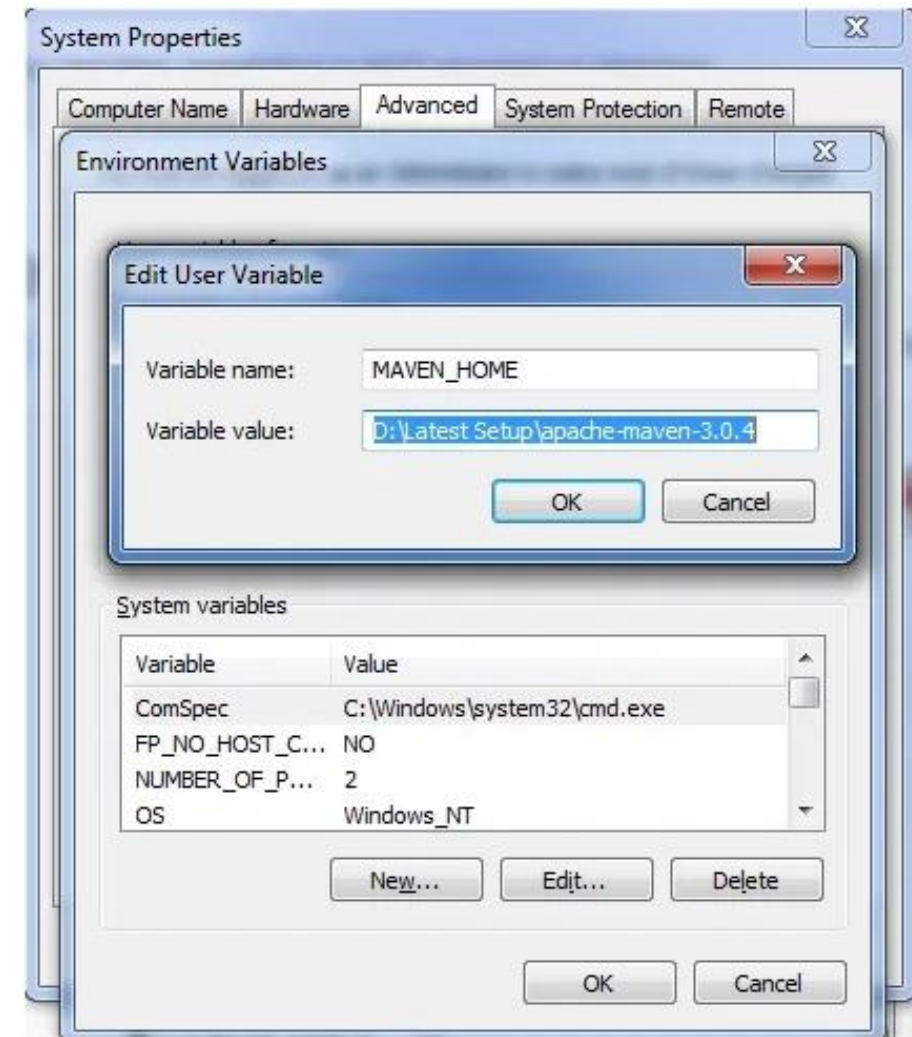**UNICAL ACADEMY**

Installing Maven in Local Machine

Creating a Script for login screen using Selenium IDE– Step-by-Step Test Procedure

Step 1:

- To install maven on windows, you need to download apache maven first.
- Download Maven latest Maven software from Download latest version of Maven
- Extract downloaded file

Step 2:

- 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

System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

Environment Variables

Edit User Variable

Variable name: MAVEN_HOME

Variable value: D:\Latest Setup\apache-maven-3.0.4

OK   Cancel

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	2
OS	Windows_NT

New...   Edit...   Delete

OK   Cancel

UNICAL ACADEMY

Installing Maven in Local Machine

**Include 'maben /bin' directory in 'PATH ' variable**

Step 3:
- The necessary step that need to follow to run maven from command prompt is,   - Update the PATH variable with 'Maven-installation/bin' directory.

Step 4:
Verify maven form console: - Once maven installation is complete test from the windows command prompt.   - go to start menu and type cmd in application location search box.   - Press enter . A new command prompt will be opened.   - Type mvn –version in command prompt   - and hit ENTER.

Email: info@unicalacademy.com | Website: www.unicalacademy.com | Phone: +91-8886978888
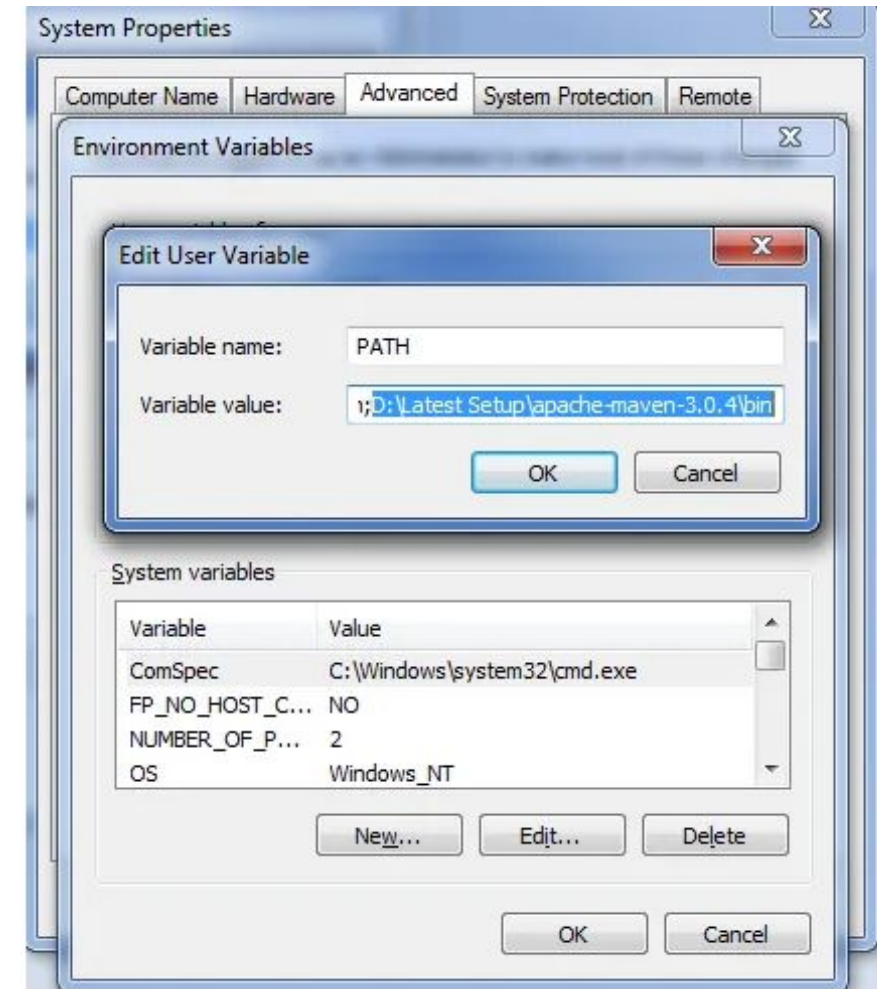
Creating Maven Project in Eclipse

Step 1:

To create Maven project in Eclipse follow the below steps:
1. Click on File menu
2. Open New project
3. Go to Maven
4. Open maven Project
5. Click on next
6. Again hit the next button
7. And press Next
8. Now write the group id, artifact id, package as shown in figure

Creating Maven Project in Eclipse

- Once you created the maven project all the files will be created automatically such as Hello Java file, pom.xml file, test case file etc.
- The directory structure of the maven project is shown in the below figure.
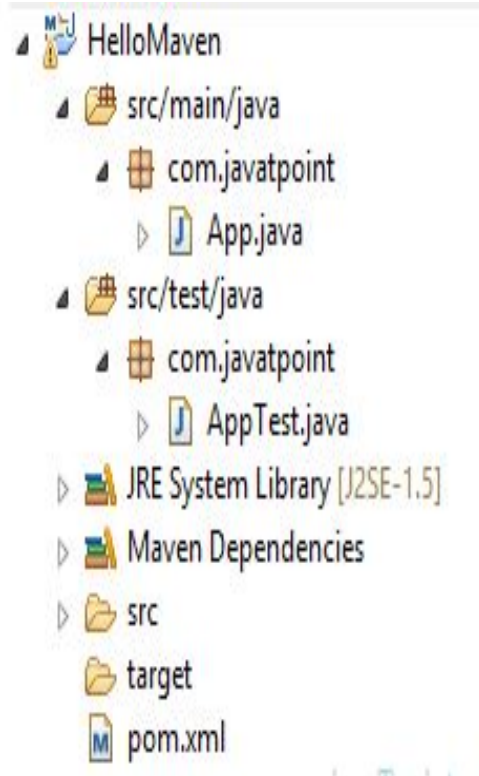
Now you can see the code of App.java file and run it. It will be like the given code:

```
package com.javatpoint;
/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

7

UNICAL ACADEMY

Creating Maven Project in Eclipse

Once you right click on your project to run the code you will be getting below options to execute the program.

m2	1 Java Applet	Alt+Shift+X, A
J	2 Java Application	Alt+Shift+X, J
Ju	3 JUnit Test	Alt+Shift+X, T
m2	4 Maven build	Alt+Shift+X, M
m2	5 Maven build...	
m2	6 Maven clean	
m2	7 Maven generate-sources	
m2	8 Maven install	
m2	9 Maven test	
	Run Configurations...	

Understanding of POM .xml

Every Maven project has a POM file that defines what the project is all about and how it should be built. POM is an acronym for project object model.

How it works:

A POM file is an XML file that is based on a specific schema.

```
<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">
```

How to do it:

Let's understand the POM file, by performing the following steps:

1 Go to a Maven project that we created in previous chapters.

2 Open the file named pom.xml .

9

Maven Integration with TestNG

- TestNG is an open-source testing framework that helps us to run before/after tests, by grouping the tests using annotations and can generate reports. It also supports Data-driven testing, Parallel execution, and Parameterization. It is easier to use.

Why We Need Maven With TestNG Integration?

- Whenever we are executing test scripts or suites using the Maven project, our dependencies are managed in the POM.xml file. However, a specific test suite cannot be selected to execute from a list of available suites.

- In TestNG, we cannot manage our dependencies, but we can select and execute particular test scripts or suites.

- Given that Maven and TestNG have different capabilities, we are integrating both using the Maven Surefire plugin.

Maven Integration with TestNG

UNICAL ACADEMY

## Work Flow Using The Maven Surefire Plugin:
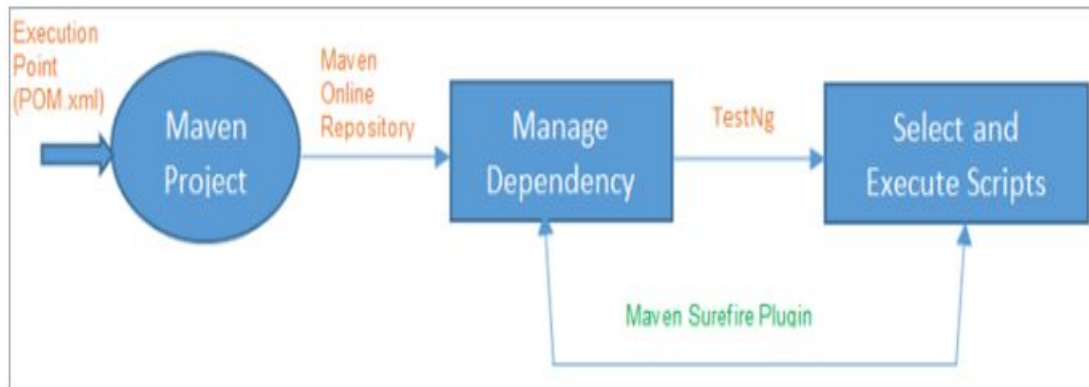
Steps to follow to Maven integration with TestNG:

1. Select the POM.xml file from the **Maven** project.

2. Add the Plugin window will be displayed

3. On clicking the OK button, the Plugin is added in POM.

4. Select any script(LoginLogoutTest), Right-click and Select **TestNG**-> Test.

• Here, execution starts from the Maven project using POM.xml. Initially, it connects to the Maven Online Repository and downloads the latest version of the dependencies.

• As TestNG has the capability to select and execute particular test scripts or suites, we are integrating this with Maven using the Maven Surefire plugin.

11

Maven Lifecycle

•Maven is based around the central concept of a build lifecycle. This means that the process for building and distributing a particular artifact (project) is clearly defined.

1. There are three built-in build lifecycles: default, clean and site.
   • The default lifecycle handles your project deployment.
   • The clean lifecycle handles project cleaning.
   • While the site lifecycle handles the creation of your   project's site documentation.
2. For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the POM will ensure they get the results they desired.

12
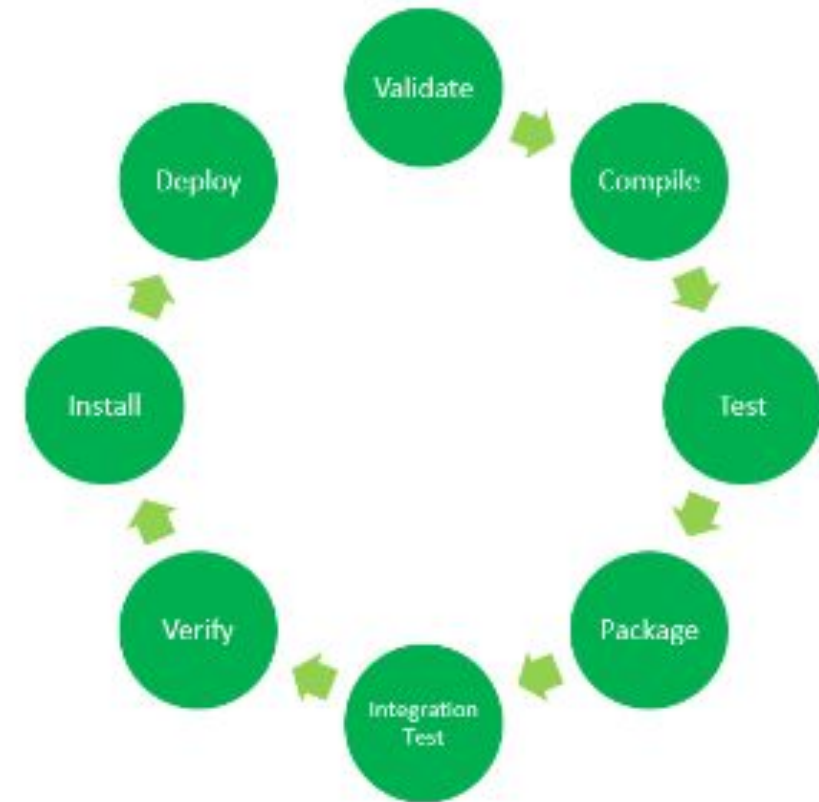
Maven Lifecycle

Maven Lifecycle Phases:

- Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.
- The default lifecycle comprises of the following phases:

✔ Validate - validate the project is correct and all necessary information is available

✔ Compile - compile the source code of the project

✔ Package - take the compiled code and package it in its distributable format, such as a

✔ Verify - run any checks on results of integration tests to ensure quality criteria are met

✔ Install - install the package into the local repository, for use as a dependency in other projects locally

✔ Deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

13

UNICAL ACADEMY

This component provides some utilities to interpret/execute some scripts for various implementations: groovy or beanshell.

Dependency declaration

```
1.    <dependency>
2.      <groupId>org.apache.maven.shared</groupId>
3.      <artifactId>maven-script-interpreter</artifactId>
4.      <version>1.3</version>
5.    </dependency>
```

maven-script-interpreter has dependency only to core interpreters library, all specific extensions should be added in your project.

For example, if you want to use **Grape** in a **Groovy** script, you must add a dependency to Ivy in your project or in the plugin that will invoke the script:

```
1.    <dependency>
2.      <groupId>org.apache.ivy</groupId>
3.      <artifactId>ivy</artifactId>
4.      <version>...</version>
5.    </dependency>
```

14

Executing Scripts Using Maven Build Tool

UNICAL ACADEMY

Using ScriptInterpreter:

Interpret BeanShell script

```
1.    ScriptInterpreter interpreter = new BeanShellScriptInterpreter();
2.    ByteArrayOutputStream out = new ByteArrayOutputStream();
3.    interpreter.evaluateScript( script content, extra classPath entries,
4.                               Map<String, ? extends Object> globalVariables, new PrintStream( out ) );
```

out.toString() returns script output.

Interpret a Groovy script

```
1.    ScriptInterpreter interpreter = new GroovyScriptInterpreter();
2.    ByteArrayOutputStream out = new ByteArrayOutputStream();
3.    interpreter.evaluateScript( script content, extra classPath entries,
4.                               Map<String, ? extends Object> globalVariables, new PrintStream( out ) );
```

out.toString() returns script output.

Executing Scripts Using Maven Build Tool

UNICAL ACADEMY

Using ScriptRunner:

ScriptRunner class will detect the script file to run based on supported extensions (.bsh, .groovy).

This class will search in the provided directory the script with the provided file Name and the supported extensions.

```
1.    ScriptRunner scriptRunner = new ScriptRunner();
2.    scriptRunner.run( "test", new File( "src/test/resources/bsh-test" ), "verify", buildContext(),
3.                      new FileLogger( logFile ) );
```

Mirror output from script interpreter:

In order to do something more with script output, eg. log by your application you must implement FileLoggerMirrorHandler

```
1. class MyMirrorHandler implements FileLoggerMirrorHandler
2. {
3.    void consumeOutput( String message )
4.    {
5.       // this method is invoked every time when flush occurs on the underlying stream.
6.    }
7. }
```

1
6

Executing Scripts Using Maven Build Tool

Use:

```
1.      ScriptRunner scriptRunner = new ScriptRunner();
2.      scriptRunner.run( "test", new File( "src/test/resources/bsh-test" ), "verify", buildContext(),
3.                  new FileLogger( logFile, new MyMirrorHandler() ) );
```

Global variables:

Your scripts will have by default two global variables:
- •basedir: the base directory of your script
- •context: the build context (see below)

You can add more global variables as it.

```
1.      ScriptRunner scriptRunner = new ScriptRunner();
2.      scriptRunner.setGlobalVariable( name, value );
```

17

Executing Scripts Using Maven Build Tool

UNICAL ACADEMY

Build context:

You can pass some values to your script using an execution context which have the type Map<String, ? extends Object> context:

```
1. private Map<String, Object> buildContext()
2. {
3.     Map<String, Object> context = new HashMap<String, Object>();
4.     context.put( "foo", "bar" );
5.     return context;
6. }
```

Then values are available in scripts context:

```
1.     // in your bsh script
2.     String value = context.get( "foo" );
```

value will be "bar"

```
1.     // in your Groovy script
2.     context.get("foo")
```

18

Executing Scripts Using Maven Build Tool

UNICAL ACADEMY

Additional classpath entries:

You can add some additional classpath entries for your script execution

```
1.    List<String> classpathEntries = list of jar paths
2.    ScriptRunner scriptRunner = new ScriptRunner();
3.    scriptRunner.setClassPath( classpathEntries );
4.    scriptRunner.run( "test", new File( "src/test/resources/bsh-test" ), "verify", buildContext(),
5.                  new FileLogger( logFile ) );
```

19

UNICAL ACADEMY

Advantages Maven Build Tool

- Helps manage all the processes, such as building, documentation, releasing, and distribution in project management
- Helps manage all the processes, such as building, documentation, releasing, and distribution in project management
- Helps manage all the processes, such as building, documentation, releasing, and distribution in project management
- The task of downloading Jar files and other dependencies is done automatically
- The task of downloading Jar files and other dependencies is done automatically
- Makes it easy for the developer to build a project in different environments without worrying about the dependencies, processes, etc.
- In Maven, it's easy to add new dependencies by writing the dependency code in the POM file

20

Overview of cucumber

- Cucumber is a testing tool that supports Behavior Driven Development (BDD). It offers a way to write tests that anybody can understand, regardless of their technical knowledge.

- In BDD, users (business analysts, product owners) first write scenarios or acceptance tests that describe the behavior of the system from the customer's perspective, for review and sign-off by the product owners before developers write their codes.

How BDD works in Cucumber Automation?

Consider you are assigned to create Funds Transfer module in a Net Banking application.

Multiple ways to test application in Cucumber Testing framework:

- There are multiple ways to test it in Cucumber Testing framework

- Fund Transfer should take place if there is enough balance in source account
- Fund Transfer should take place if the destination a/c details are correct
- Fund Transfer should take place if transaction password / rsa code / security authentication for the transaction entered by user is correct
- Fund Transfer should take place even if it's a Bank Holiday
- Fund Transfer should take place on a future date as set by the account holder

Cucumber Installation and Setup with Eclipse

UNICAL ACADEMY

1) Launch the *Eclipse IDE* and from Help menu, click "***Install New Software***".
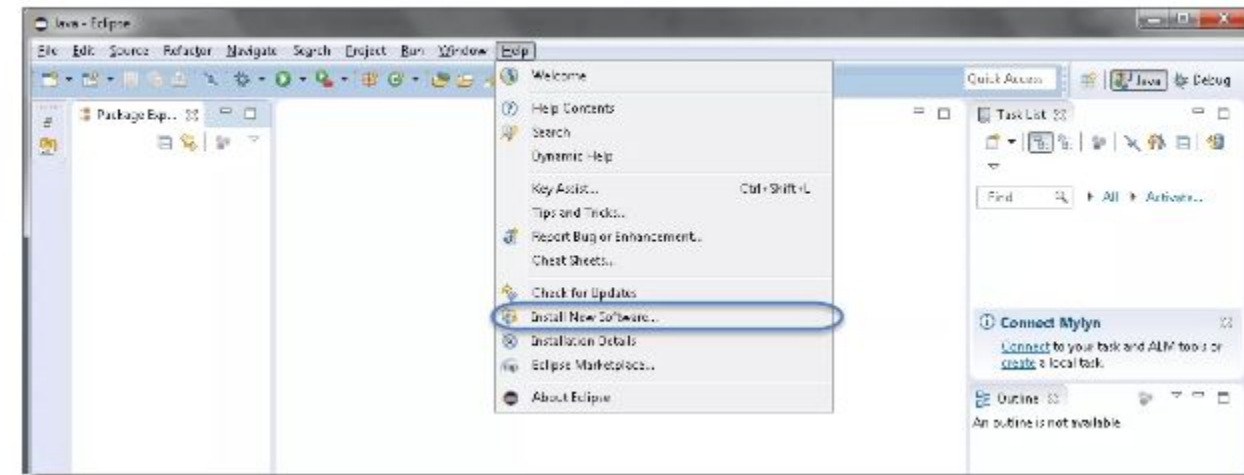
You will see a dialog window, click "***Add***" button.

Cucumber Installation and Setup with Eclipse
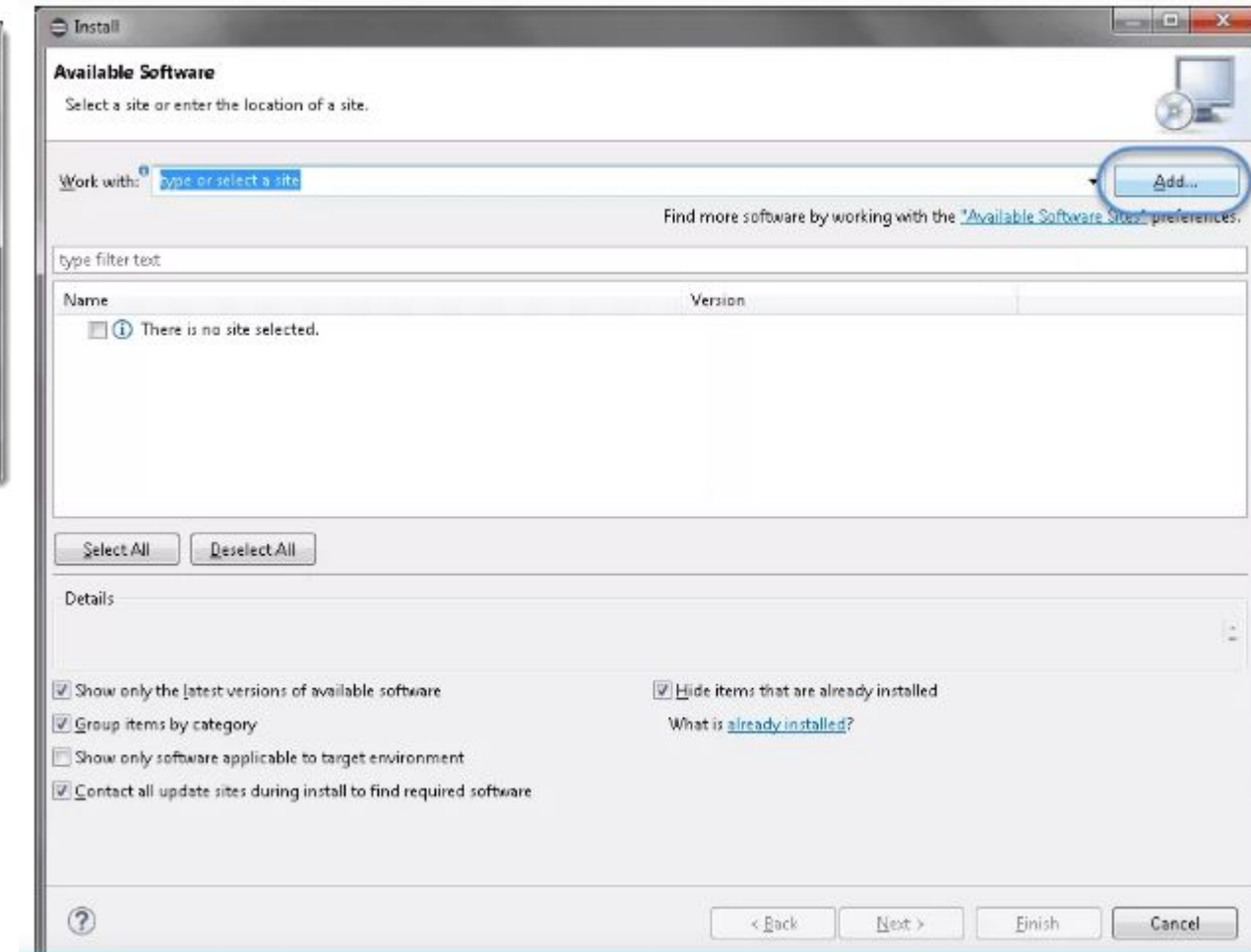
Click on **Next**.
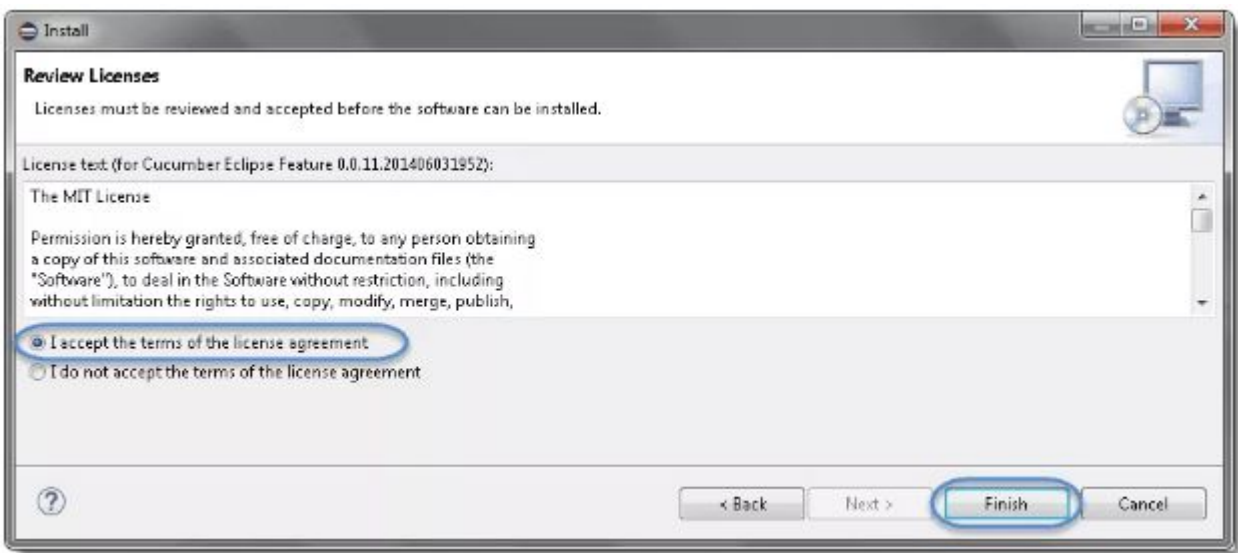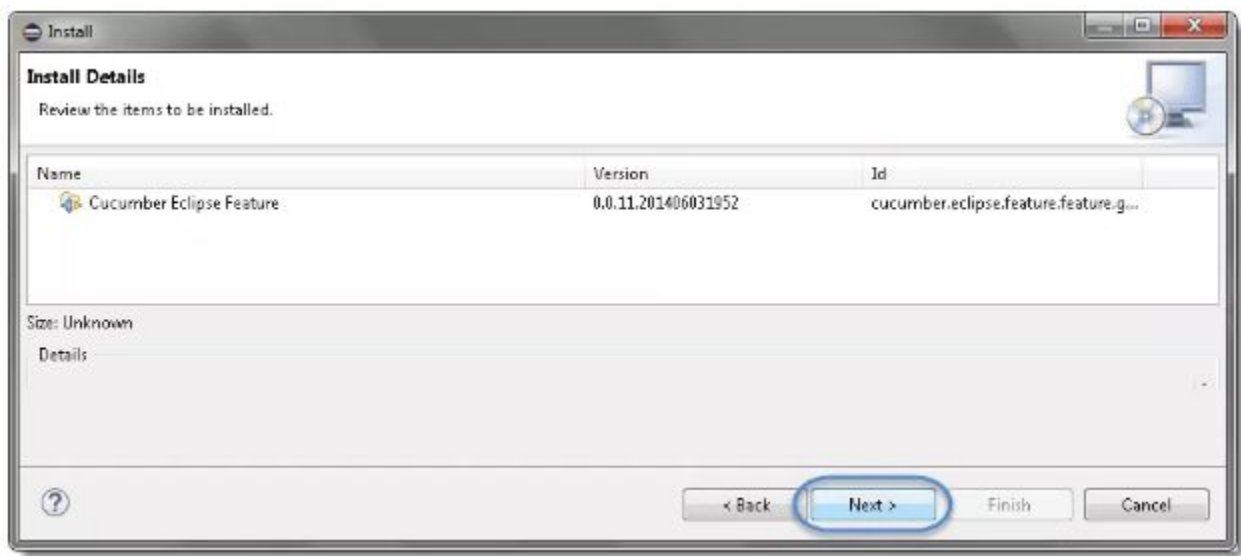
Click "**I accept the terms of the license agreement**" then click **Finish**.

UNICAL ACADEMY

Cucumber Installation and Setup with Eclipse

Let it install, it will take few seconds to complete.
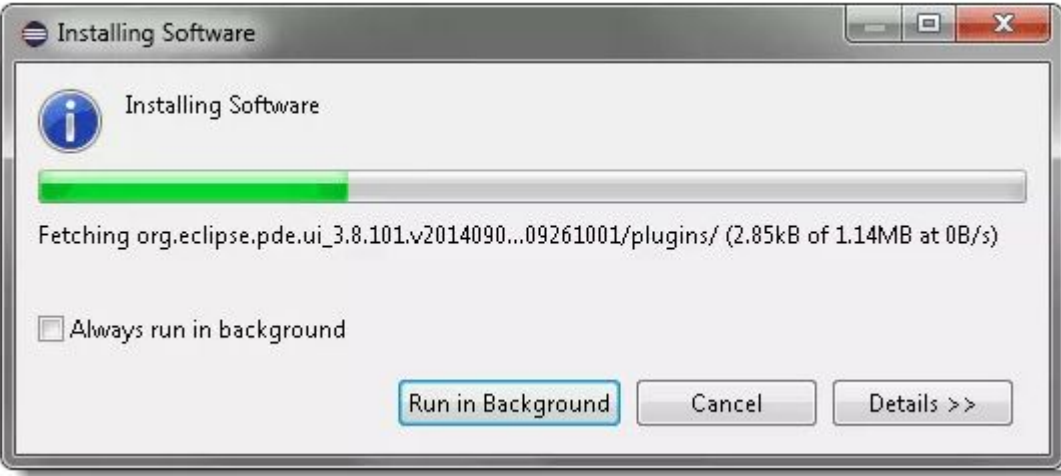
You may or may not encounter a Security warning, if in case you do just click **OK**.

25

UNICAL ACADEMY

Cucumber Installation and Setup with Eclipse

You are all done now, just click **Yes**.

26

UNICAL ACADEMY

Overview of Gherkin keywords

- Gherkin is primarily used to write structured tests which can later be used as project documentation.

- The property of being structured gives us the ability to automate them. This automation is done by Cucumber/SpecFlow.

Create Feature File

- There are several approaches to create a feature file in different IDEs, here we are creating it in Eclipse IDE.

- We can create a feature file with the ".feature" extension.

Following are the steps to create a feature file by using eclipse IDE:

1. In order to create a feature file in eclipse, go to the **File** option at the left side of the window then select **New**. When you click the **New**, you will get the following three options:

    o **Project**

    o **Example**

    o **Other**

    Select **Other** option from these three options.

Create Feature File

Consider the following image: to create file.
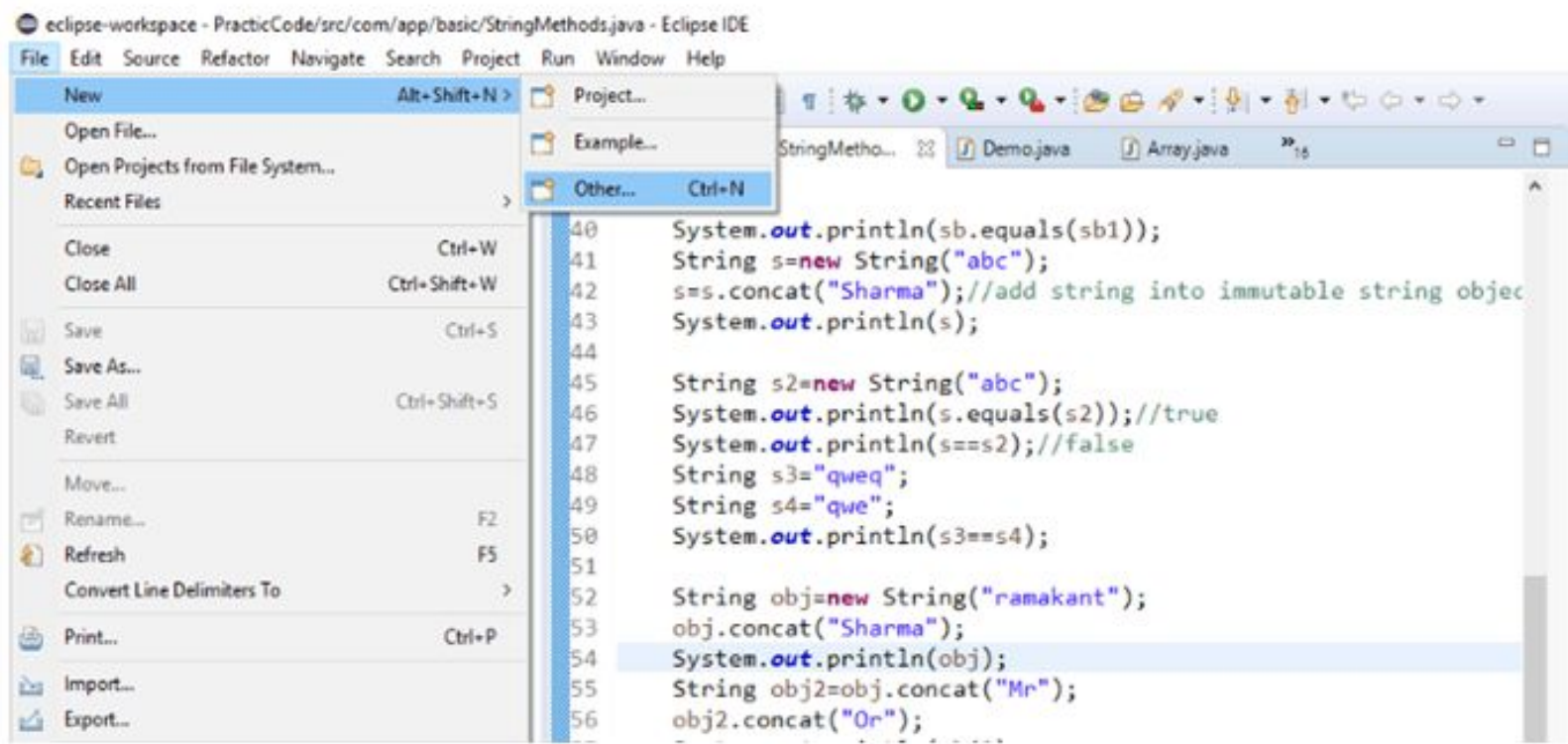
Create Feature File

After selecting Other option, you will get several options, from these options, select General < File option, and then click Next.
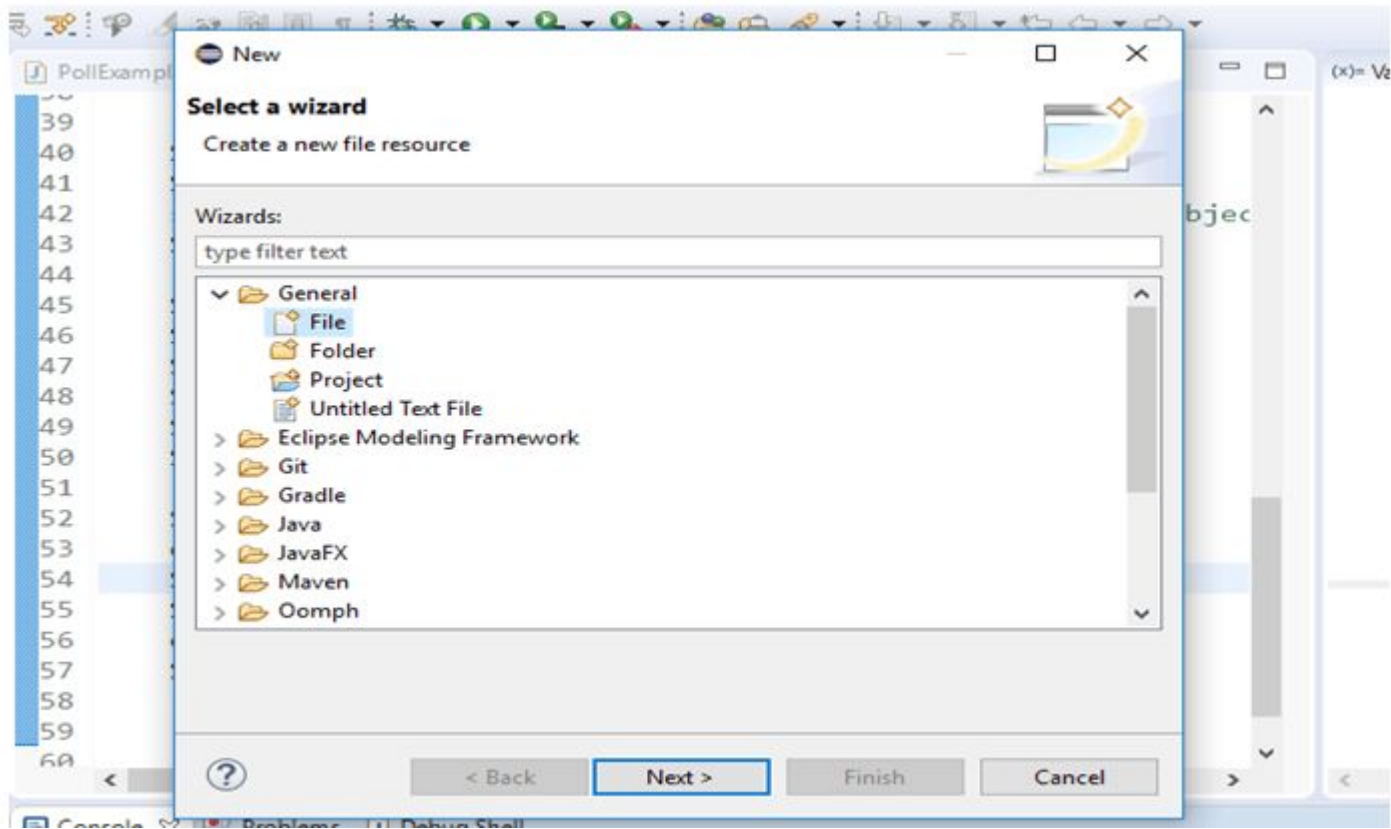
Create Feature File

- After clicking the Next, select the project inside which you want to create a feature file.

- After selecting the project, you can create a feature file by giving a name and ".feature" extension. You can provide any name to the feature file on your own choice. After providing a name, click on the Finish button.
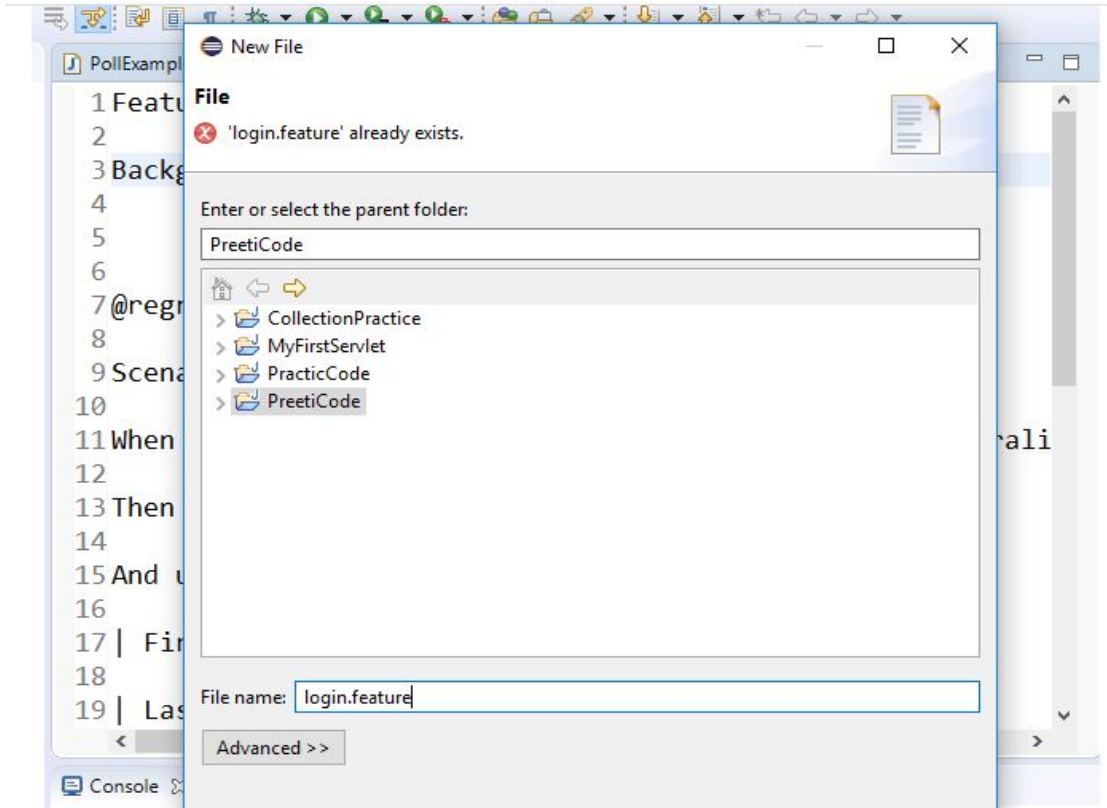
Now, the created feature file will appear inside the project.

UNICAL ACADEMY
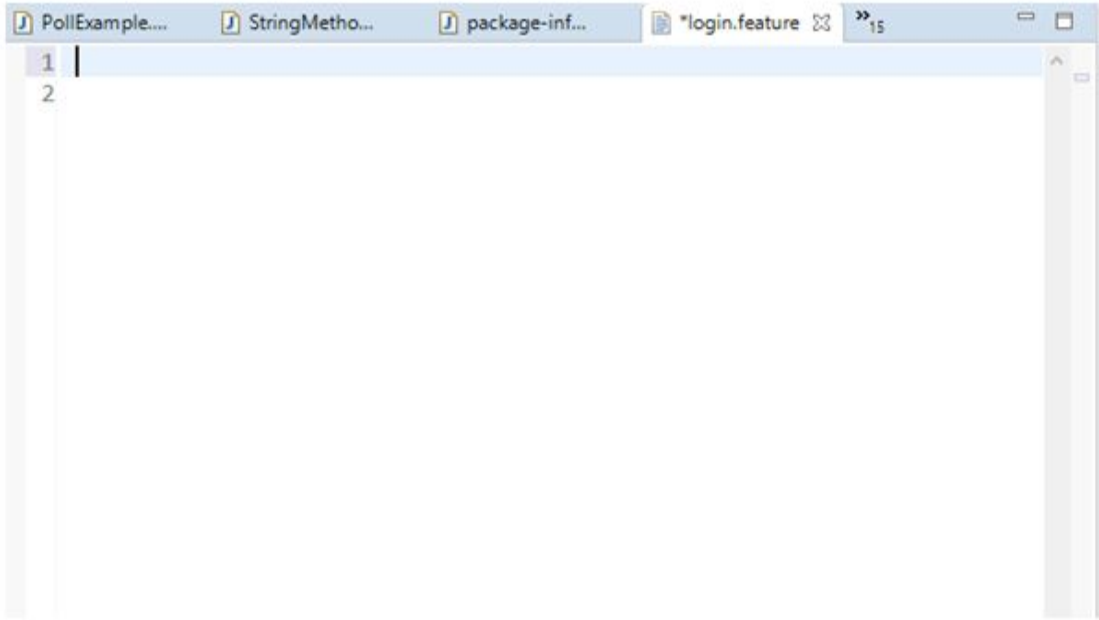
Generate Step Definition File

- While editing the .feature file, type a reference to a step definition. ...
- Press Alt+Enter to show the Create Step Definition intention action:
- Select the target step definition file from the list: ...
- In the selected step definition file that opens in the editor, enter the desired code.

UNICAL ACADEMY

Integrate Cucumber with Selenium WebDriver

- Step 1- Create Java project
- Step 2- Add Selenium jars and add Cucumber jars
- Step 3- Create feature file and write scenarios.
- Step 4- Write TestRunner and execute the same.
- Step 5- Once TestRunner execution complete, you will get all methods with default implementation (Copy all of them).
- Step 6- Create Java program and paste the code which we have copied in Step 5
- Step 7- Write the code for methods which has no body
- Step 8- Execute the Test Runner and check the result.

Integrate Cucumber with Selenium WebDriver

UNICAL ACADEMY

Email: info@unicalacademy.com | Website: www.unicalacademy.com | Phone: +91-8886978888

JUnit Test Runner Class
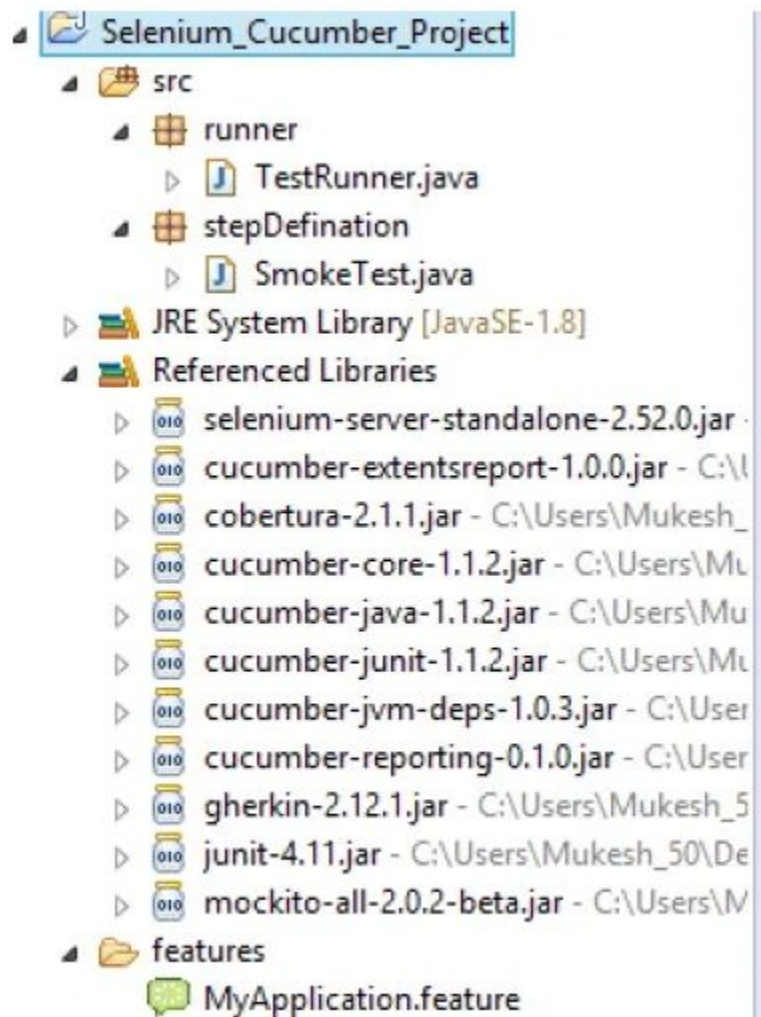
- A JUnit Runner is class that extends JUnit's abstract Runner class. Runners are used for running test classes. The Runner that should be used to run a test can be set using the @RunWith annotation.

```
@RunWith(MyTestRunner.class)
public class MyTestClass {

@Test
public void myTest() {
..
}
}
```

- JUnit tests are started using the JUnitCore class. This can either be done by running it from command line or using one of its various run() methods

```
JUnitCore.runClasses(MyTestClass.class);
```

- JUnitCore then uses reflection to find an appropriate Runner for the passed test classes. One step here is to look for a @RunWith annotation on the test class. If no other Runner is found the default runner (BlockJUnit4ClassRunner) will be used. The Runner will be instantiated and the test class will be passed to the Runner. Now it is Job of the Runner to instantiate and run the passed test class.

How do Runners work?

Runners functions:

- It will parse the Gherkin feature file.
- It will execute the functions written in the step definition file according to feature file statements.
- JUnit will combine the test case result.
- It will build the test report in the specified format (which can be html/JSON).

Data Driven Testing in Cucumber

UNICAL ACADEMY
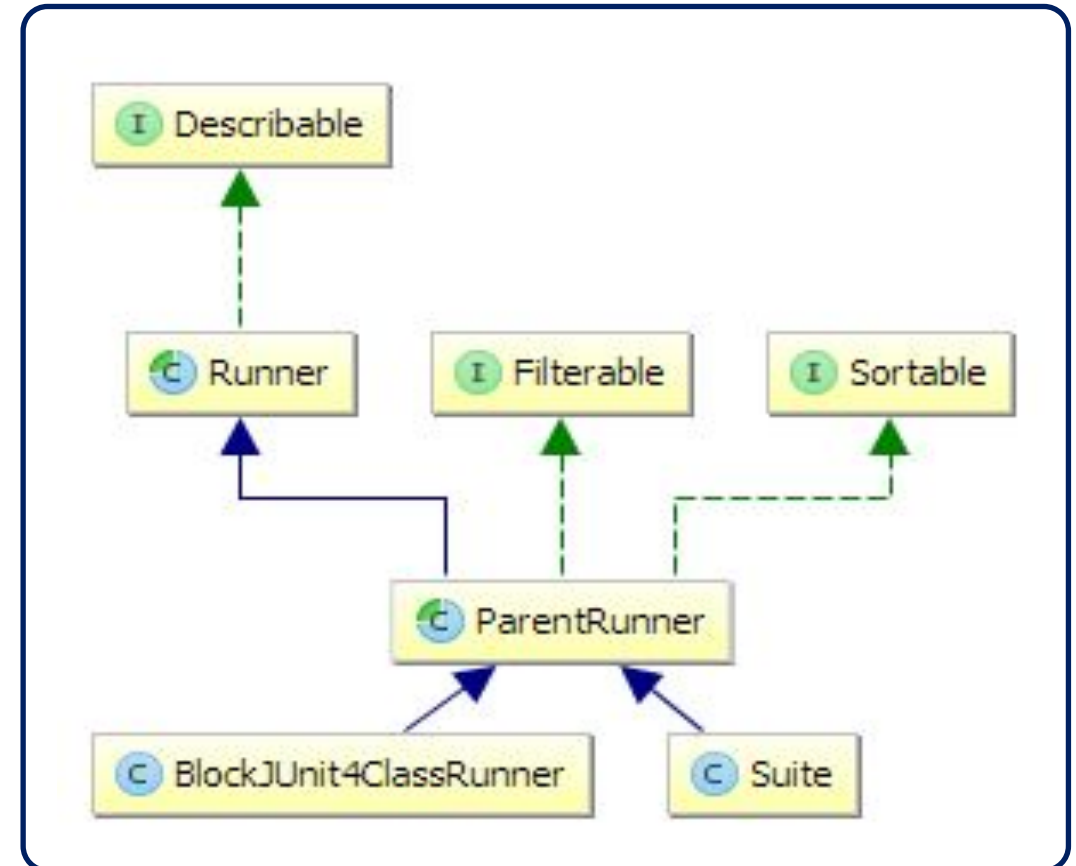
- Most commercial automated software tools on the market support some sort of Data Driven Testing, which allows to automatically run a test case multiple times with different input and validation values.

- There are different ways to use the data insertion within the Cucumber and outside the Cucumber with external files.

1. **Parameterization in Cucumber**

2. **Data-Driven Testing Using Examples Keyword**

3. **Data Tables in Cucumber**

4. **Maps in Data Tables**

Configure Cucumber with Maven and Jenkins

1. Access Jenkins URL. The default is port 8080.

2. Go to "**Manage Jenkins**" –> "**Manage Plugins**."

Configure Cucumber with Maven and Jenkins

UNICAL ACADEMY
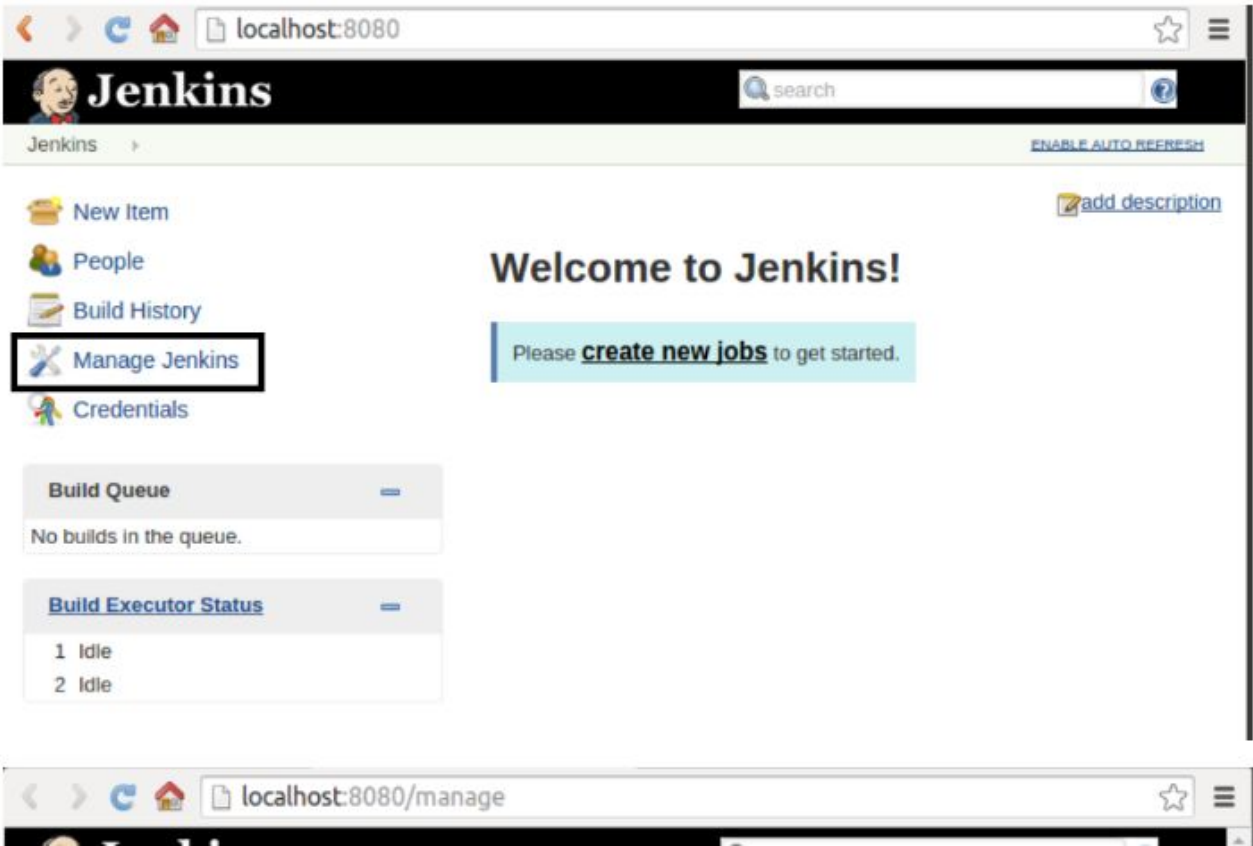
Install the following plugins
- Cucumber Test Results plugin
- Cucumber Reports
- Cucumber perf plugin

Make sure while installing these plugins, all the dependent plugins are also successfully installed. This is to ensure the plugins work as expected without glitches.

39

UNICAL ACADEMY

Configure Cucumber with Maven and Jenkins

The next step is to navigate to "**Manage**" –> "**Global Tool Configuration**." Under "**JDK**," set the path for JDK.

Under "**Maven**," set the path for Maven.

Configure Cucumber with Maven and Jenkins

UNICAL ACADEMY

Create a "**New Item**" as a Maven Project as shown below. Enter any arbitrary name.

Once the job is created, click on the "**Configure**" link on the left-hand side panel.
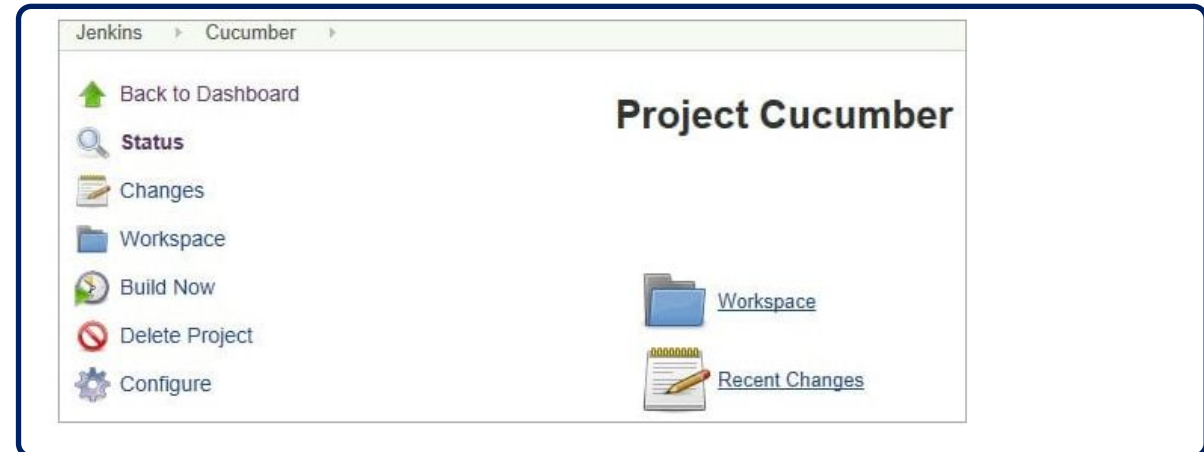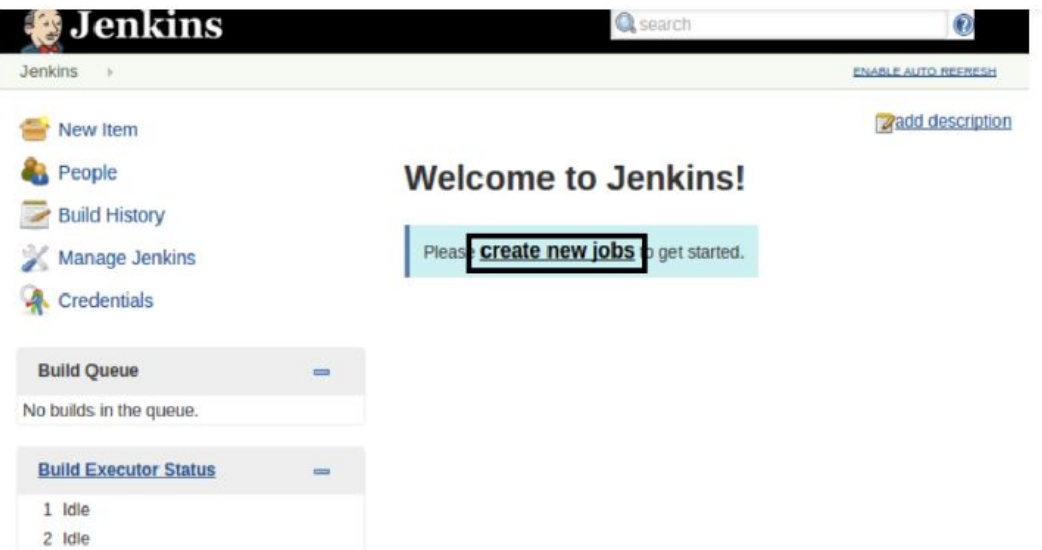
Scroll down the page. Set the full path of pom.xml under "**Root POM**" and "Goal and options" as 'test' and proceed to save the configuration.

41

Configure Cucumber with Maven and Jenkins

UNICAL ACADEMY
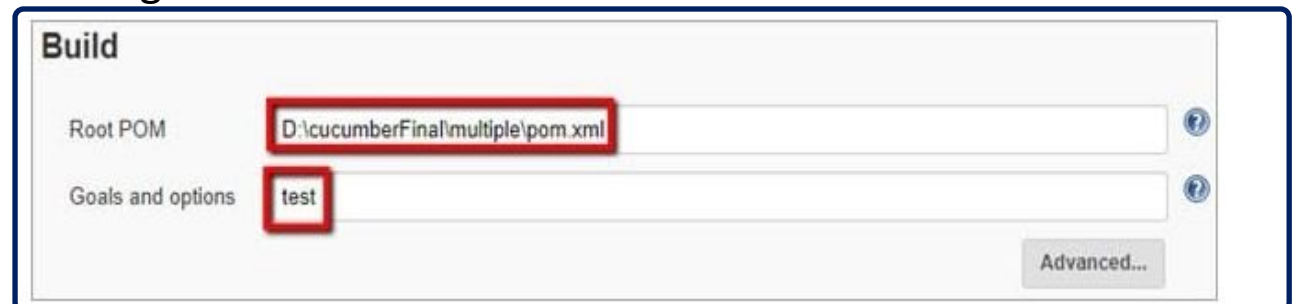
Once all the above steps are completed, click on the "Build Now" button.
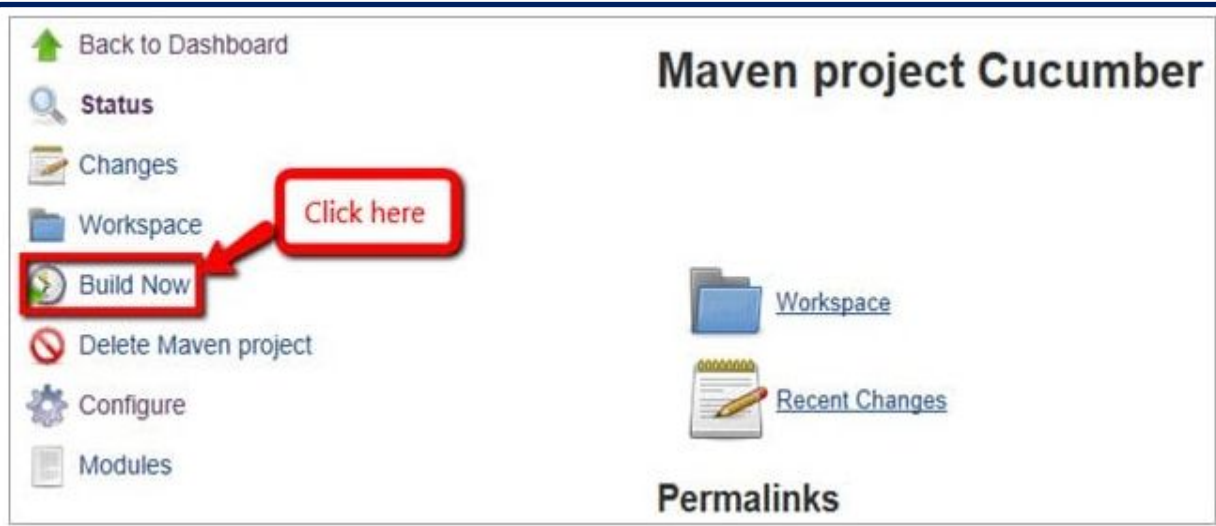
The build will be executed, and the corresponding testing.xml file, which is the pom.xml, will get executed.

42

UNICAL ACADEMY

How to generate reports in CUCUMBER

Step 1 − Create a Maven project named cucumber Report in Eclipse.

Step 2 − Create a package named CucumberReport under src/test/java.

Step 3 − Create a feature file named cucumberReport.feature.