



Prepared By,  
Thiru

<b>S.No</b>	<b>Table of Contents</b>
1	About Selenium WebDriver.
2	Reason for Exclusion of other tools
3	WebDriver Installation
4	Configure Eclipse with Selenium WebDriver
5	vu-premex application : Online appointments demo with the Framework design
6	Jenkins integration with Selenium WebDriver
7	Test Automation Milestones
8	Time, Resource and Costs

# About Selenium WebDriver

**Selenium WebDriver** one of the most key component of **SELENIUM** Releases and on which current Automation industry totally rely on, specifically if we say “**Open Source Community**”.

What is WebDriver in simple and easy language if we say then “**It is an API that’s easy to explore and understand, which help us to make our tests easier to read and maintain.**” WebDriver is not linked to any kind of Test Framework or Tool and this makes this API more useful as we can use the same as per our needs or knowledge of other integration open sources like JUNIT, TestNG etc.

- ✦ **A well designed Object Oriented API that provides improved support for web-app testing problems.**
- ✦ **Supports dynamic web pages where element of a Page may change without the Page itself being reloaded.**
- ✦ **All the limitations of SELENIUM RC rectified in this Selenium WebDriver**

## WebDriver Architecture

- ❖ WebDriver implemented on Layered Design, the idea behind this implementation is more and more usage of WebDriver for automation and this could be possible by fitting best fit languages.
- ❖ Implementation of WebDriver is that each browser has a language that is most natural to use when attempting to drive it. Drivers are built as per the best fit language and we can just see the wrapper around them.
- ❖ We can say that for any of browser driver if any of the features works there in one binding language then it should be easy to add support to other binding languages also.

- ❖ Web Driver is a compact Object Oriented API which can directly interacts with the Application under tests.
- ❖ WebDriver utilizes the browser native compatibility to automation without using any peripheral entity.



## Features of WebDriver

1) “Interface WebDriver”, which represents an idealized web browser used for testing. Three categories of methods in this class.

- Control to browser
- Web Elements selection
- Debugging

2) We discussed above about language bindings with browsers and it is just a thin wrapper which is exposed for us to write code as per our needs. This leads WebDriver to support “**Multiple Languages**” as well as “**Multiple Support Browsers**” which means that if any API supports multiple languages then this automatically leads to “**Multiple Platforms**”.

3) Web Page composed of Web Elements and these Drivers’s will communicate with the Web Page. For communicating with Web Page means communication with Elements present on the Web Page like for example: “Textboxes”, Buttons”, “Links” etc.

## Multiple Languages Binding support



## Multiple Browsers support



## Multiple Platform support



## Key Features:

- ✓ WebDriver: One of the most core component from Selenium with vast features where driver covers all the features and properties of explorers
- ✓ WebDriver: Gives us the opportunity to write once and execute on multiple platforms
- ✓ WebDriver: Provides speed to Selenium architecture where communication with Application under Test becomes faster
- ✓ WebDriver: Despite of working on any language or environment you need not to learn anything new it is just grab the knowledge and start producing results
- ✓ WebDriver: Gives opportunity to explore something from core and can contribute from scratch till end in Automation process
- ✓ WebDriver: Introduce AndroidDriver and IphoneDriver to explore Mobile world.

# Reason for the Exclusion of other tools

UFT	Selenium	Watir	GEB
<b>License cost</b>			
UFT is Hp licensed product available through single-seater floating or concurrent licenses	Selenium is at present the most powerful freeware of open source automation tool.	watir is an open-source (BSD) family of Ruby libraries for automating web browsers and is free	GeB is open source software based on Groovy and is free.
<b>Support</b>			
Dedicated Hp support	User and professional community support available	Limited support on open source community	Limited support on open source community
<b>Scripting language</b>			
VB Script	Java, CSharp, python, Ruby, php, Perl, JavaScript	Ruby	Groovy
<b>Object recognition</b>			
Through object spy	Fire bug, Firepath	Through webrecorder	GeB ide
<b>Learning Time</b>			
Less	Much more	More	More

<b>UFT</b>	<b>Selenium</b>	<b>Watir</b>	<b>GEB</b>
<b>Framework</b>			
Capability to build frameworks such as keyword-driven, data-driven and hybrid	Junit, Nunit, keyword-driven, data driven, TestNG and hybrid driven	Ruby supported Frameworks - RSpec, Cucumber.	Grails, Gradle, Maven
<b>Continuous Integration</b>			
Can be achieved through Jenkins	Can be achieved through Jenkins	Can be achieved through Ruby script.	Achieved using Grails, Gradle plug-in along with Jenkins Gradle plug-in
<b>Browser support</b>			
Google Chrome (until ver 23) Internet Explorer , Firefox ( ver 21)	Google Chrome , Internet Explorer , Firefox , Opera , HtmlUnit	Firefox, ie, Chrome, Opera, Safari	Firefox, ie, Chrome, Opera, Safari
<b>Environment Support</b>			
only Windows	Windows , Linux , Solaris OS X , Others (If browser & JVM or JavaScript support exists)	windows 8.1, Linux 13.10, MAC OS X 10.9, Solaris 11.1 (need JSSH compiled)	windows Xp/Vista/7, Linux, DOS (OS support depends on web-driver availability)



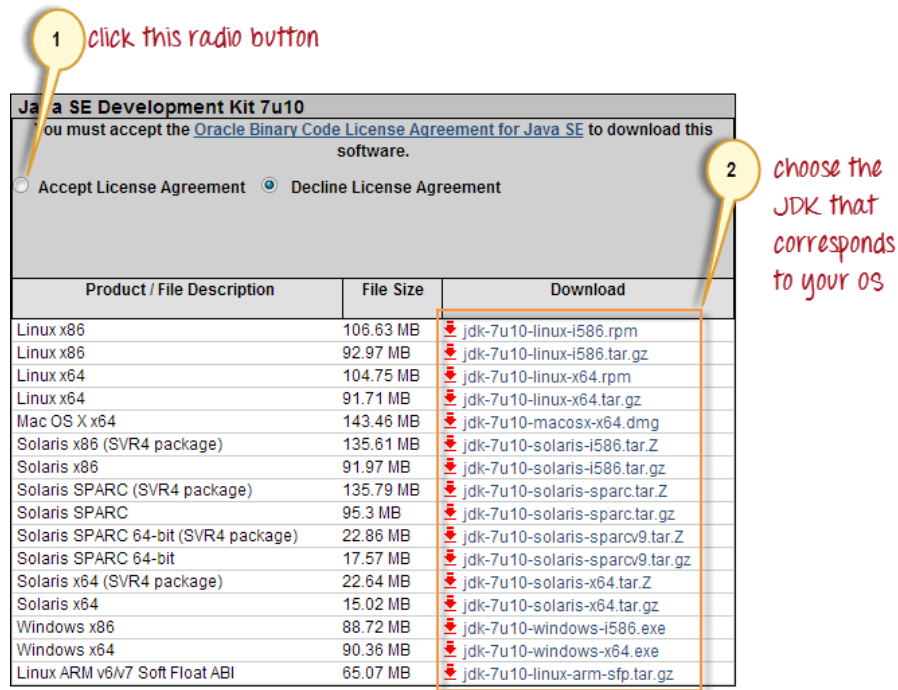
# WebDriver Installation

## Step 1 - Install Java on your computer

- ✓ Download and install the **Java Software Development Kit (JDK)**



- ✓ Then, follow the next screenshot.



- ✓ This JDK version comes bundled with Java Runtime Environment (JRE) so you do not need to download and install the JRE separately.

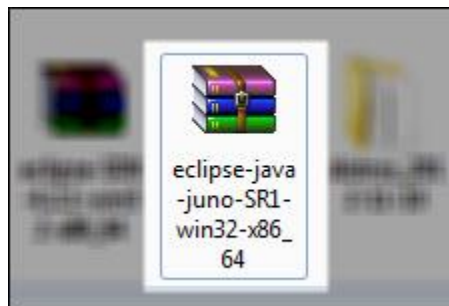
## Step 2 - Install Eclipse IDE

- ✓ Download "**Eclipse IDE for Java Developers**". Be sure to choose correctly between Windows 32 Bit and 64 Bit versions.

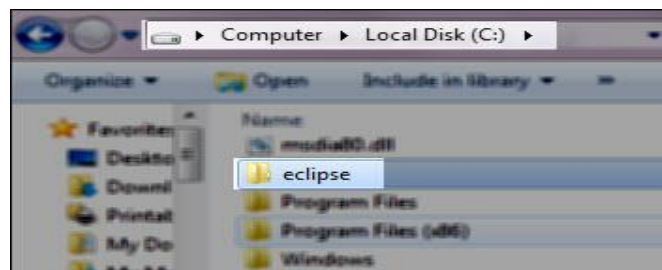
*Choose the correct version here*



- ✓ You should be able to download a ZIP file named "eclipse-java-juno-SR1-win32-x86\_64.zip" (the version number "SR1" may change over time).

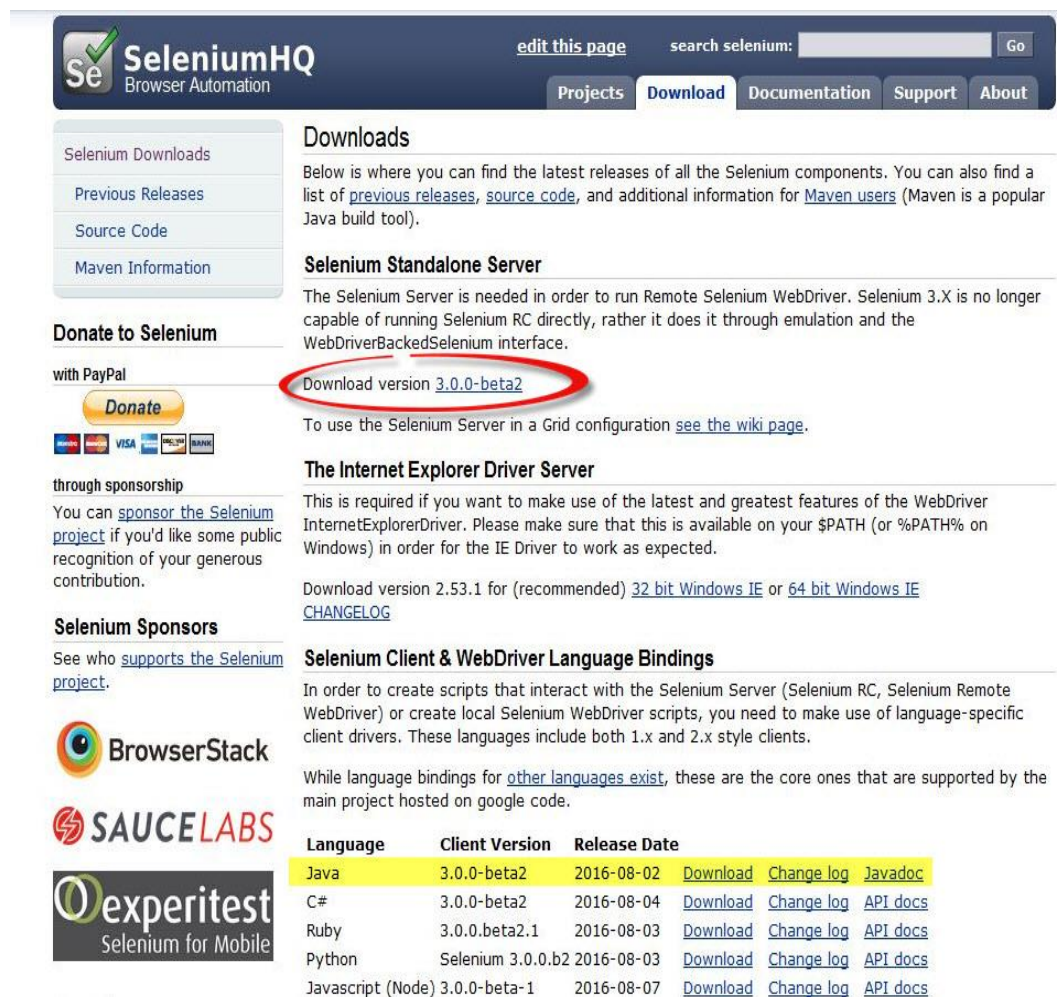


- ✓ Inside that ZIP file, there is an "eclipse" folder which contains all the application files. You can extract the "eclipse" folder anywhere you want in your PC; but for this tutorial, extract it to your C drive.



## Step 3 - Download the Selenium Java Client Driver

- ✓ You can download the **Selenium Java Client Driver**. You will find client drivers for other languages there, but only choose the one for Java.



The screenshot shows the SeleniumHQ website. The 'Downloads' section is highlighted, and the link 'Download version 3.0.0-beta2' is circled in red. Below this, there is a table of Selenium Client & WebDriver Language Bindings.

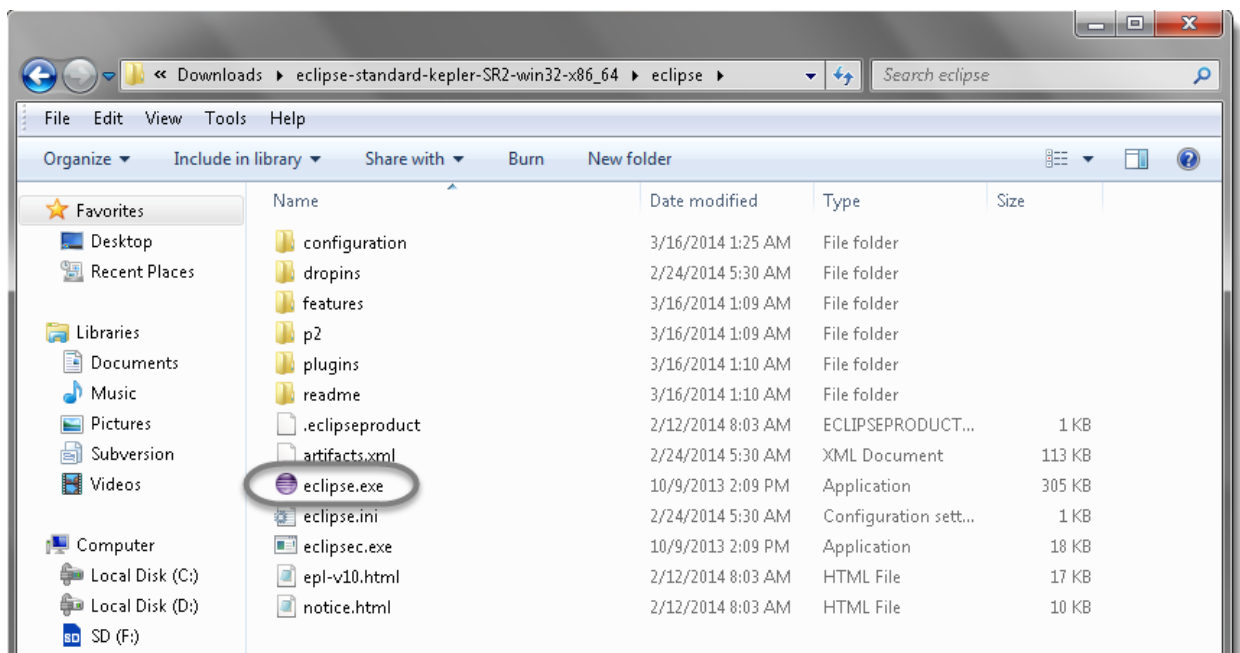
Language	Client Version	Release Date	Download	Change log	Javadoc
Java	3.0.0-beta2	2016-08-02	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">Javadoc</a>
C#	3.0.0-beta2	2016-08-04	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Ruby	3.0.0.beta2.1	2016-08-03	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Python	Selenium 3.0.0.b2	2016-08-03	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Javascript (Node)	3.0.0-beta-1	2016-08-07	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>

- ✓ This download comes as a ZIP file named "selenium-3.0.zip". For simplicity, extract the contents of this ZIP file on your C drive so that you would have the directory "C:\selenium-2.25.0\". This directory contains all the JAR files that we would later import on Eclipse.

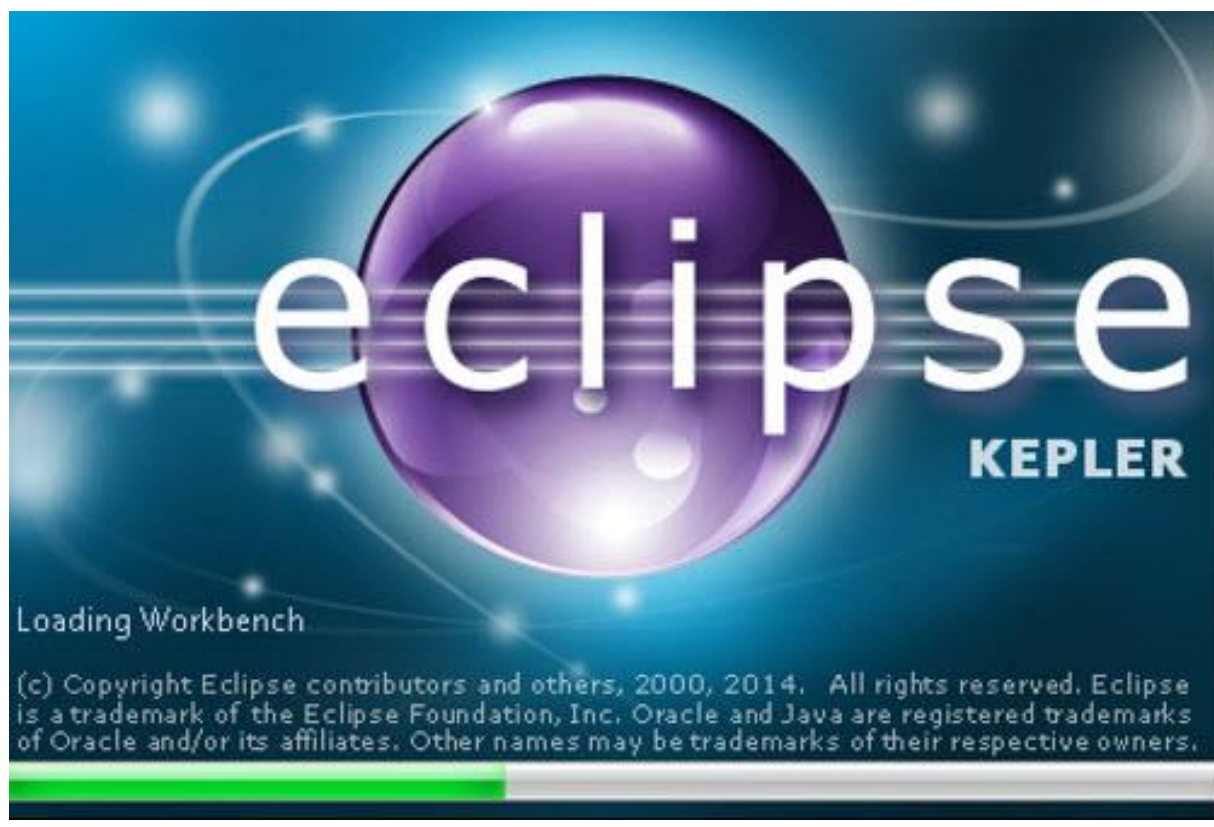
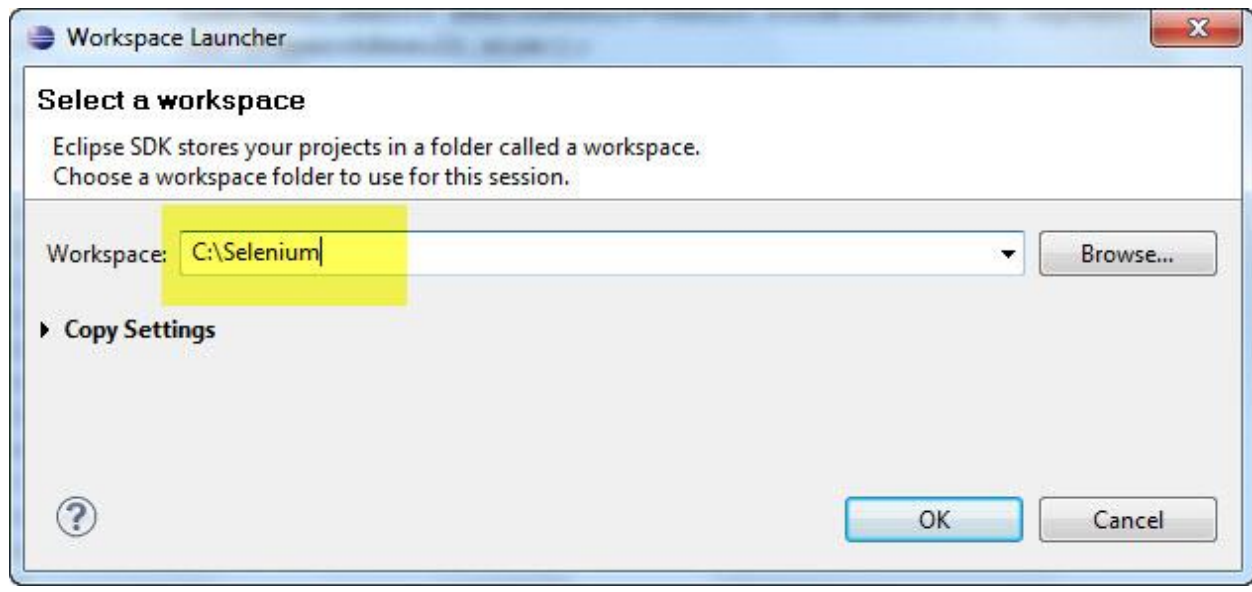
# Configure Eclipse with Selenium WebDriver

## Launch the Eclipse IDE & Create a Workspace

- Double click on '**eclipse.exe**' to start eclipse. First time when you start eclipse, it will ask you to select your **workspace** where your work will be stored as shown in below image.

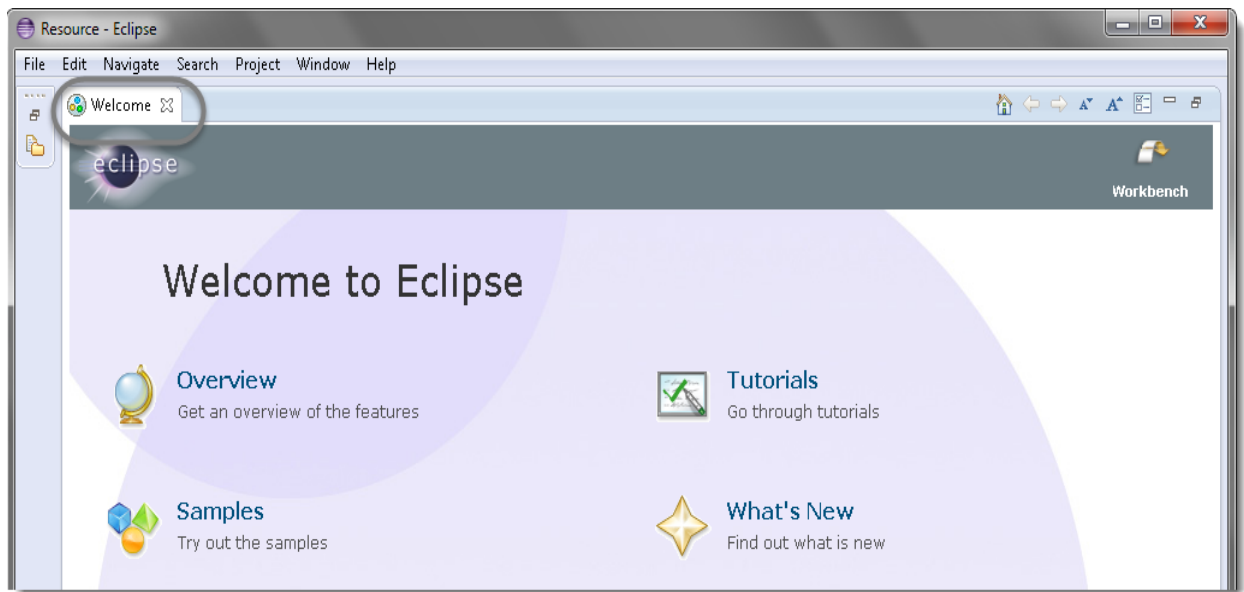


- Create a '**working directory**' for all of your projects. Think of it like '**My Documents**' in the Windows operating system. It's a folder which contains a lot of your documents, but there's nothing to prevent you from creating another folder called 'My Other Documents' (for instance) to house other documents.
- You can change it later on from '**Switch Workspace**' under '**File**' menu of eclipse. After selecting workspace folder, Eclipse will be open.



- You may see the window like this, this is the Welcome window for Eclipse. You may close this window.



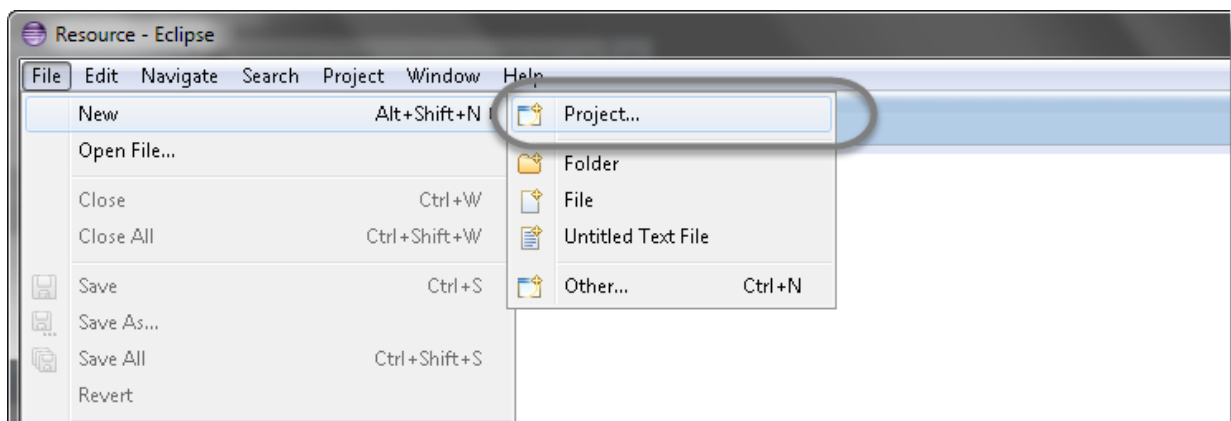


## Create a new Project

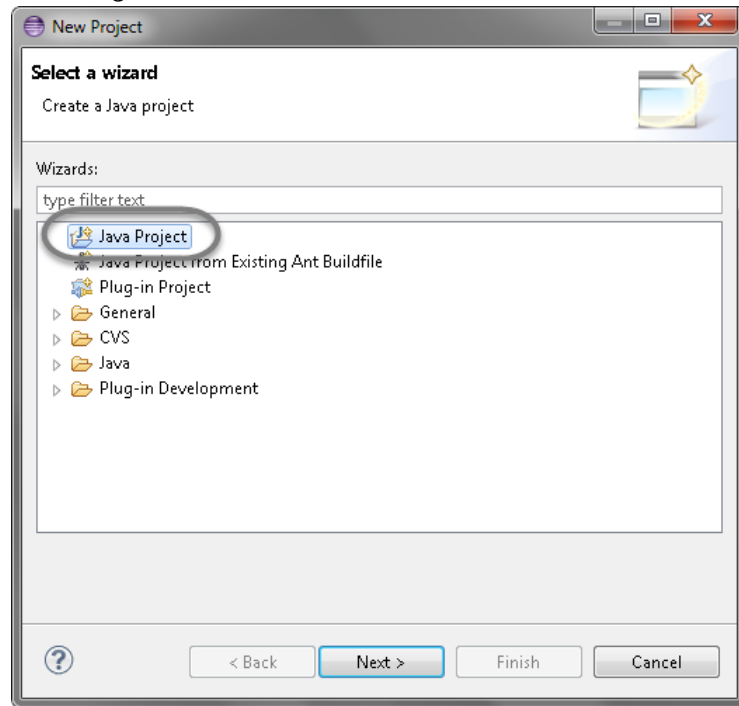
**Projects:** A collection of related code. Generally speaking, each project encompasses one independent program. Each programming assignment you do will typically require its own project.

Once you've established your workspace, you'll want to create a project and begin writing code. In Eclipse, projects are the next-smallest functional unit after workspaces, but where you might have only one workspace, you will usually have several projects inside one workspace.

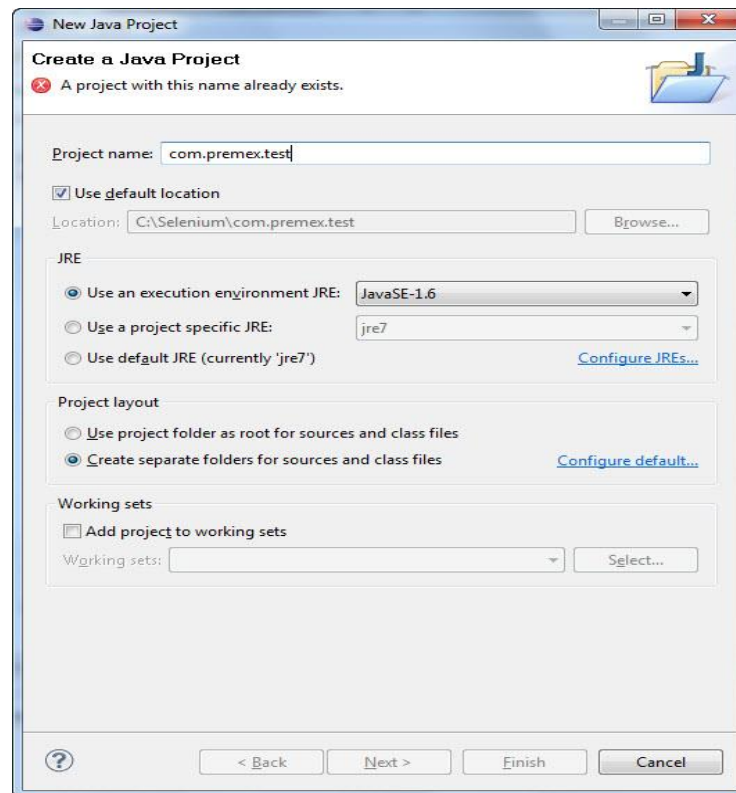
➤ Create new Java Project from **File > New > Project**.



- Select **Java Project** and click **Next**.

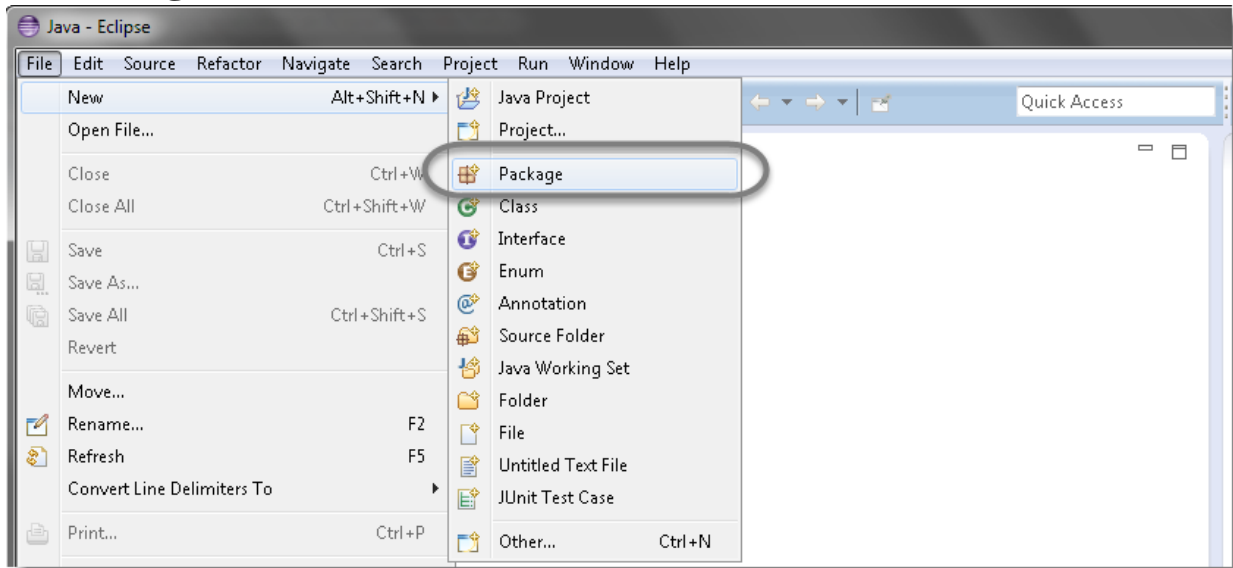


- Give your Project name “com.premex.test” as shown in below given figures. Click on **Finish** button.

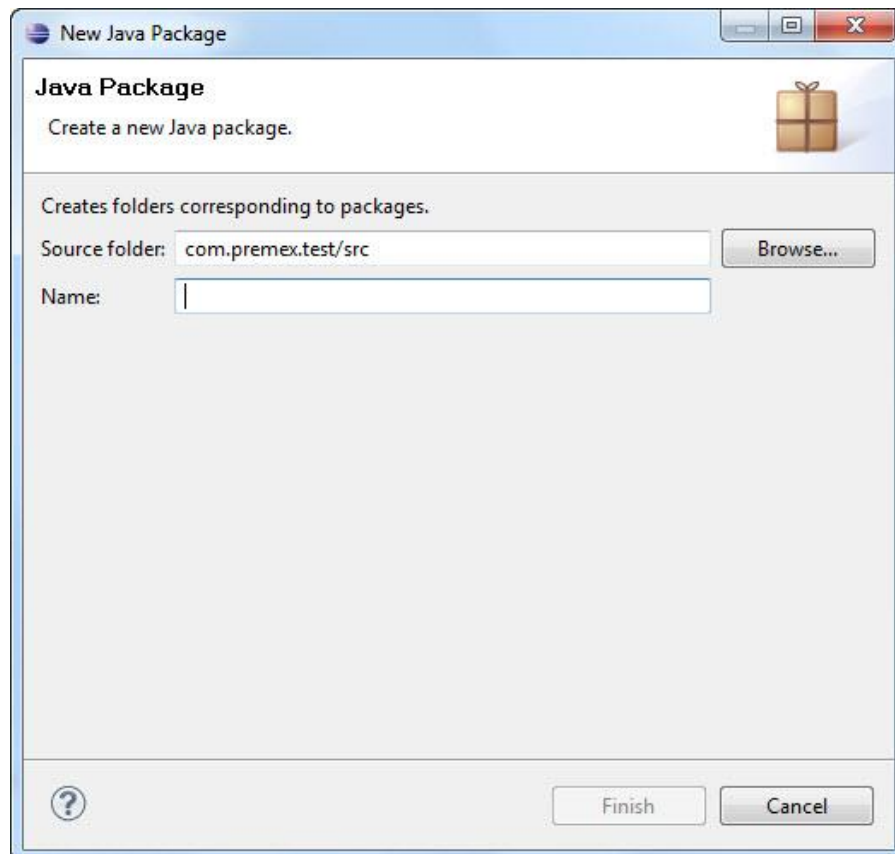


## Create a new Package

- Right click on Project name '**com.premex.test**' and select **New > Package**.



- Give your Package name '**ExecutionEngine**' and click on **Finish** button.

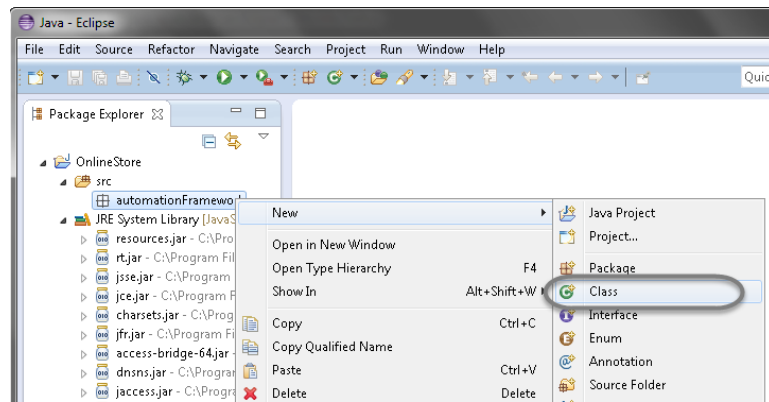




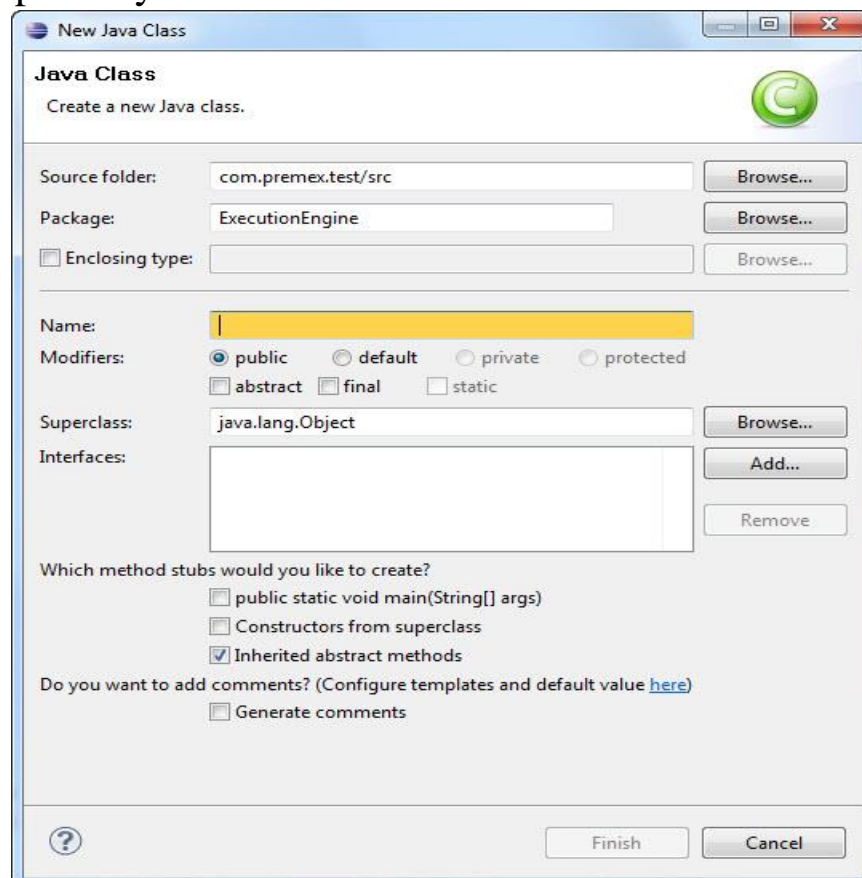
## Create a new Class

Now that you have a project set up, you're going to want start writing some new classes.

- 1) Right click on Package '**ExecutionEngine**' and select **New > Class**.



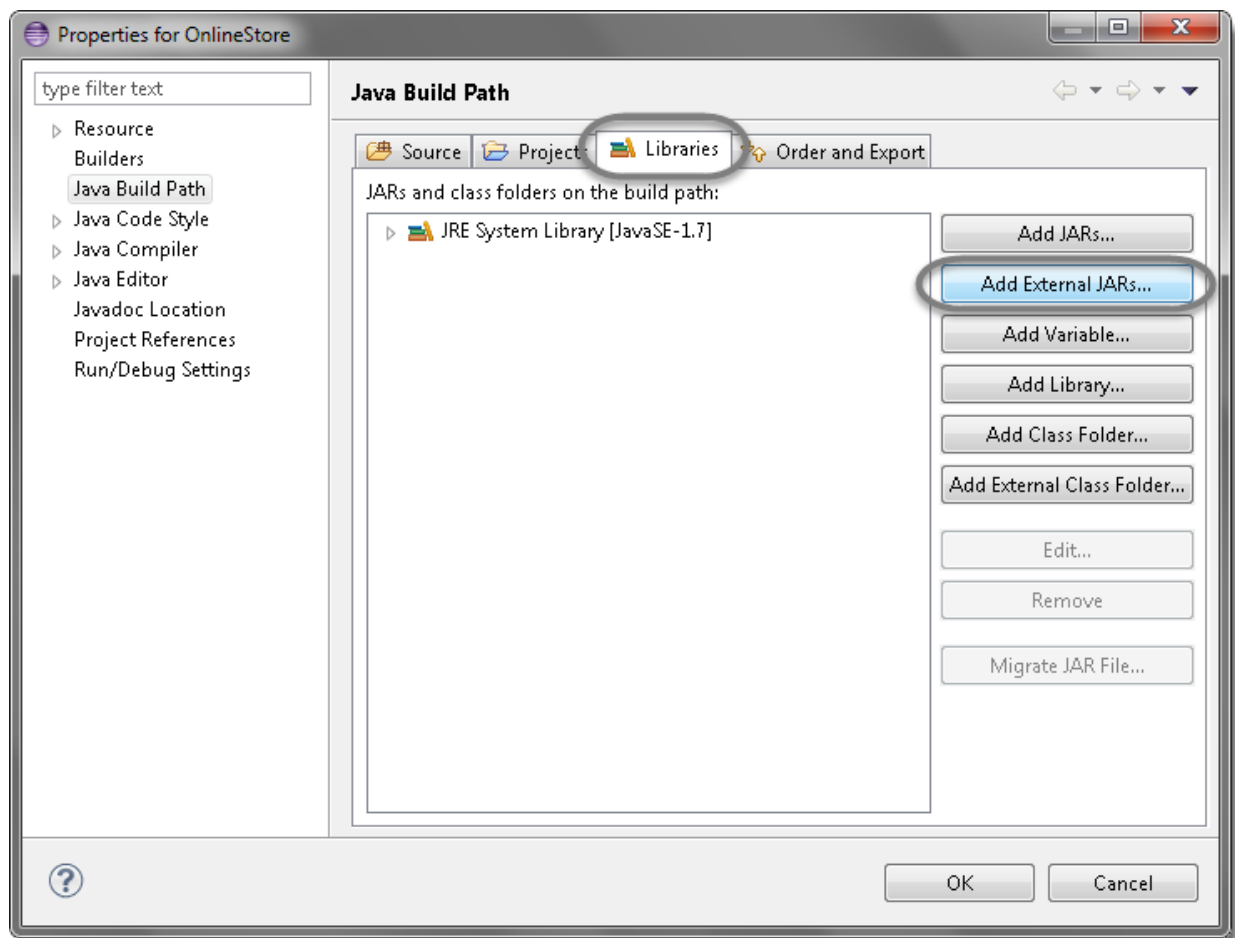
- Give your Class name '**OnlineAppointments**', check the option '**public static void main**' and click on **Finish** button. This will bring up totally a sweet class creation window.



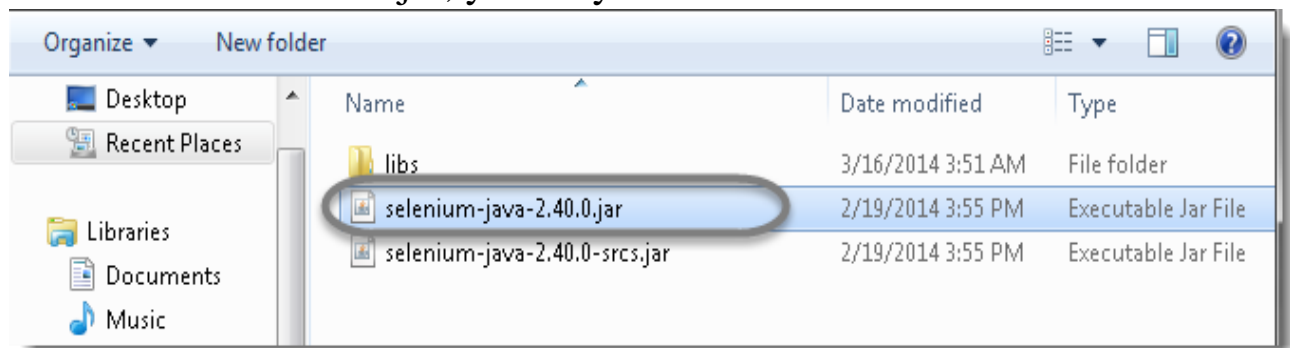
## Add External Jars to Java build path

Now you need to add Selenium WebDriver's Jar files in to Java build path.

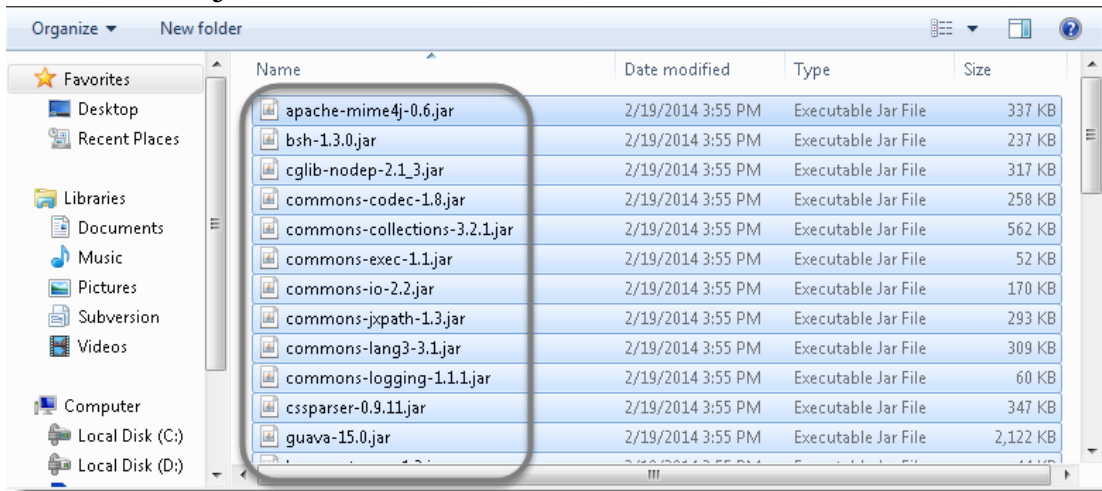
- 1) Right click on Project '**com.premex.test**' > **Select Properties** > Java build path. Then navigate to **Libraries** tab and click **Add External JARs**.



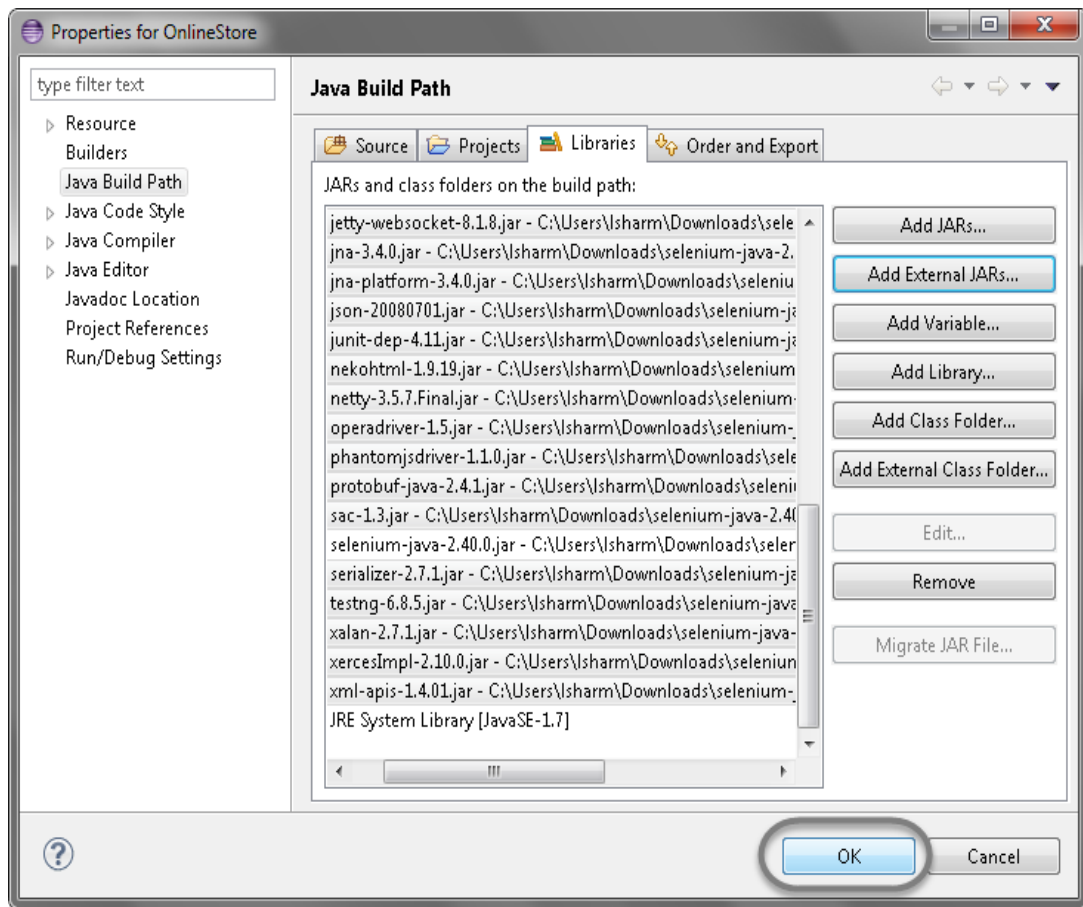
- Add Selenium Java jar, you may add the source file too.



- Add all the jars from the **libs** folder as well.



- Click **OK**.



- That's all about configuration of WebDriver with eclipse. Now you are ready to write your test script in eclipse and run it in WebDriver.

# **vu-premex application : Online appointments demo**

## **Scenario:**

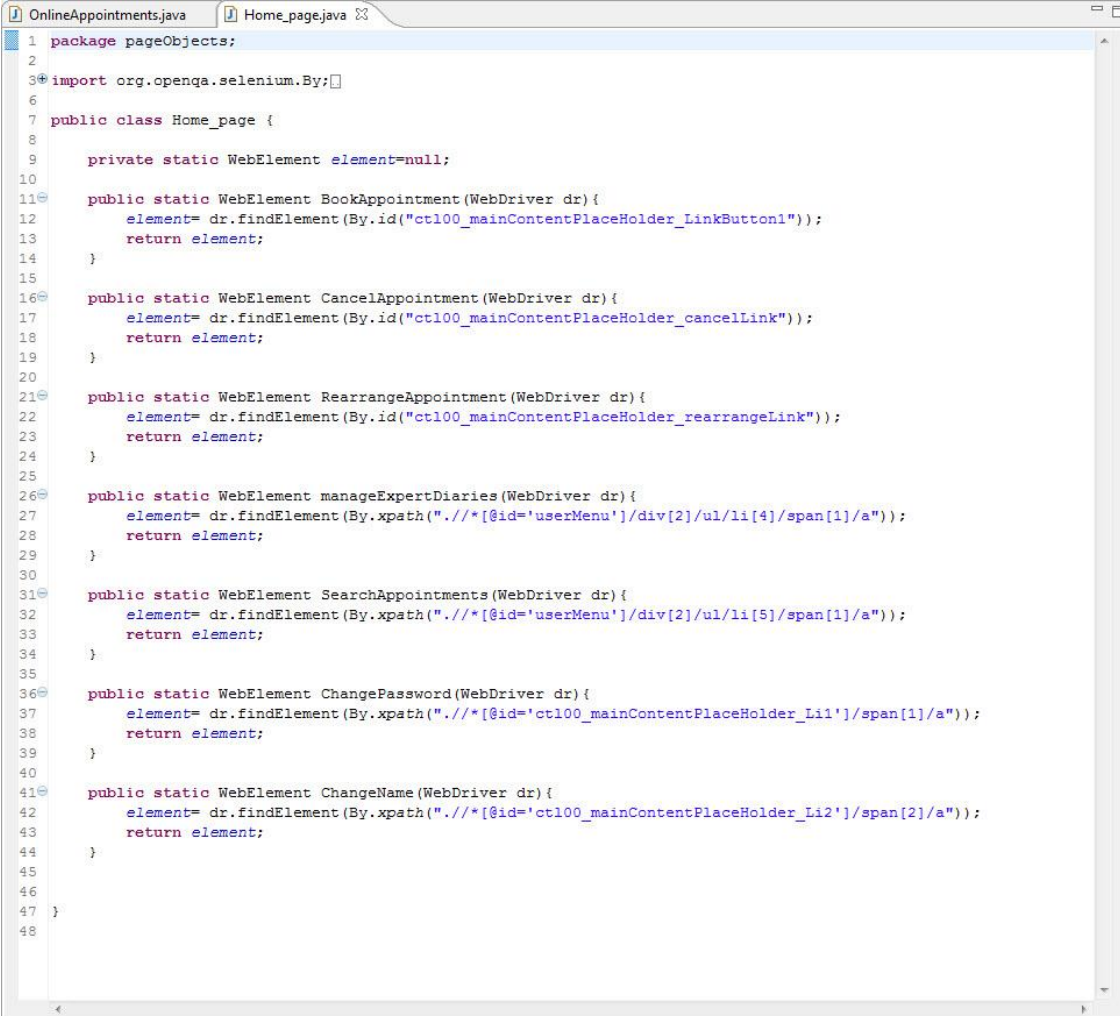
- ✓ Launch a new Firefox browser.
- ✓ Open <https://vuqa.premex.com/vu/Login.aspx>
- ✓ Login with the Premex user credential.
- ✓ Book one appointment.
- ✓ Search the Booked appointment.
- ✓ Rearrange the appointment.
- ✓ Cancel the appointment.
- ✓ Add appointment in Manage Expert diaries.
- ✓ Cancel appointment in Manage Expert diaries.
- ✓ Close the Browser.

## **Page Object Model**

Creating Selenium test cases can result in an unmaintainable project. One of the reasons is that too many duplicated code is used. Duplicated code could be caused by duplicated functionality and this will result in duplicated usage of locators. The disadvantage of duplicated code is that the project is less maintainable. If some locator will change, you have to walk through the whole test code to adjust locators where necessary. By using the page object model we can make non-brittle test code and reduce or eliminate duplicate test code. Beside of that it improves the readability and allows us to create interactive documentation. Last but not least, we can create tests with less keystroke. An implementation of the page object model can be achieved by separating the abstraction of the test object and the test scripts.

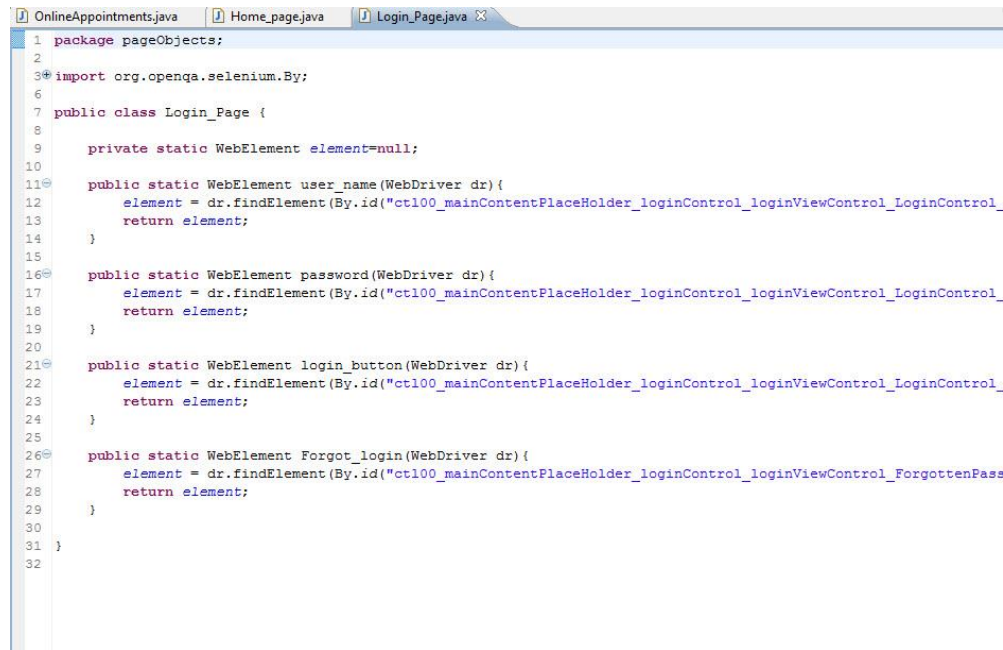
## Steps

- Create a 'New Package' file and name it as '**pageObjects**', by right click on the Project and select **New > Package**. It is always recommended to use this structure, as it is easy to understand, easy to use and easy to maintain.
- Create a 'New Class' file and refer the name to the actual page from the test object, by right click on the above created Package and select **New > Class**. In our case it is **Home\_page** and **Login\_Page**.
- Now create a **Static Method** for each **Element (Object)** in the Home Page. Each method will have an **Argument (dr)** and a **Return value (element)**.



```
1 package pageObjects;
2
3 import org.openqa.selenium.By;
4
5
6 public class Home_page {
7
8     private static WebElement element=null;
9
10
11     public static WebElement BookAppointment(WebDriver dr){
12         element= dr.findElement(By.id("ctl00_mainContentPlaceholder_LinkButton1"));
13         return element;
14     }
15
16     public static WebElement CancelAppointment(WebDriver dr){
17         element= dr.findElement(By.id("ctl00_mainContentPlaceholder_cancelLink"));
18         return element;
19     }
20
21     public static WebElement RearrangeAppointment(WebDriver dr){
22         element= dr.findElement(By.id("ctl00_mainContentPlaceholder_rearrangeLink"));
23         return element;
24     }
25
26     public static WebElement manageExpertDiaries(WebDriver dr){
27         element= dr.findElement(By.xpath(".*//*[@id='userMenu']/div[2]/ul/li[4]/span[1]/a"));
28         return element;
29     }
30
31     public static WebElement SearchAppointments(WebDriver dr){
32         element= dr.findElement(By.xpath(".*//*[@id='userMenu']/div[2]/ul/li[5]/span[1]/a"));
33         return element;
34     }
35
36     public static WebElement ChangePassword(WebDriver dr){
37         element= dr.findElement(By.xpath(".*//*[@id='ctl00_mainContentPlaceholder_Li1']/span[1]/a"));
38         return element;
39     }
40
41     public static WebElement ChangeName(WebDriver dr){
42         element= dr.findElement(By.xpath(".*//*[@id='ctl00_mainContentPlaceholder_Li2']/span[2]/a"));
43         return element;
44     }
45
46
47 }
48
```

- Driver is being passed as an Argument so that Selenium is able to locate the element on the browser (dr).
- Element is returned, so that an Action can be performed on it.
- Method is declared as **Public Static**, so that it can be called in any other method without instantiate the class.
- Follow the same rule for creating **Login\_Page** class.

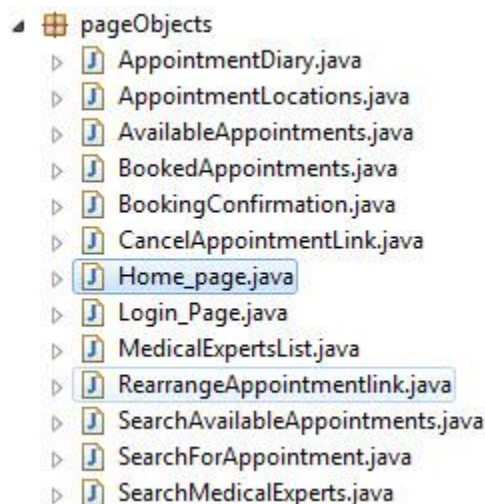


```

1 package pageObjects;
2
3 import org.openqa.selenium.By;
4
5
6 public class Login_Page {
7
8     private static WebElement element=null;
9
10
11     public static WebElement user_name(WebDriver dr){
12         element = dr.findElement(By.id("ctl00_mainContentPlaceHolder_loginControl_loginViewControl_LoginControl_
13     return element;
14 }
15
16     public static WebElement password(WebDriver dr){
17         element = dr.findElement(By.id("ctl00_mainContentPlaceHolder_loginControl_loginViewControl_LoginControl_
18     return element;
19 }
20
21     public static WebElement login_button(WebDriver dr){
22         element = dr.findElement(By.id("ctl00_mainContentPlaceHolder_loginControl_loginViewControl_LoginControl_
23     return element;
24 }
25
26     public static WebElement Forgot_login(WebDriver dr){
27         element = dr.findElement(By.id("ctl00_mainContentPlaceHolder_loginControl_loginViewControl_ForgottenPass
28     return element;
29 }
30 }
31
32

```

- Create page objects class for all the pages like **Home\_page** and **Login\_page** class.
- Your Project explorer window will look like this now.



```

pageObjects
├── AppointmentDiary.java
├── AppointmentLocations.java
├── AvailableAppointments.java
├── BookedAppointments.java
├── BookingConfirmation.java
├── CancelAppointmentLink.java
├── Home_page.java
├── Login_Page.java
├── MedicalExpertsList.java
├── RearrangeAppointmentlink.java
├── SearchAvailableAppointments.java
├── SearchForAppointment.java
└── SearchMedicalExperts.java

```

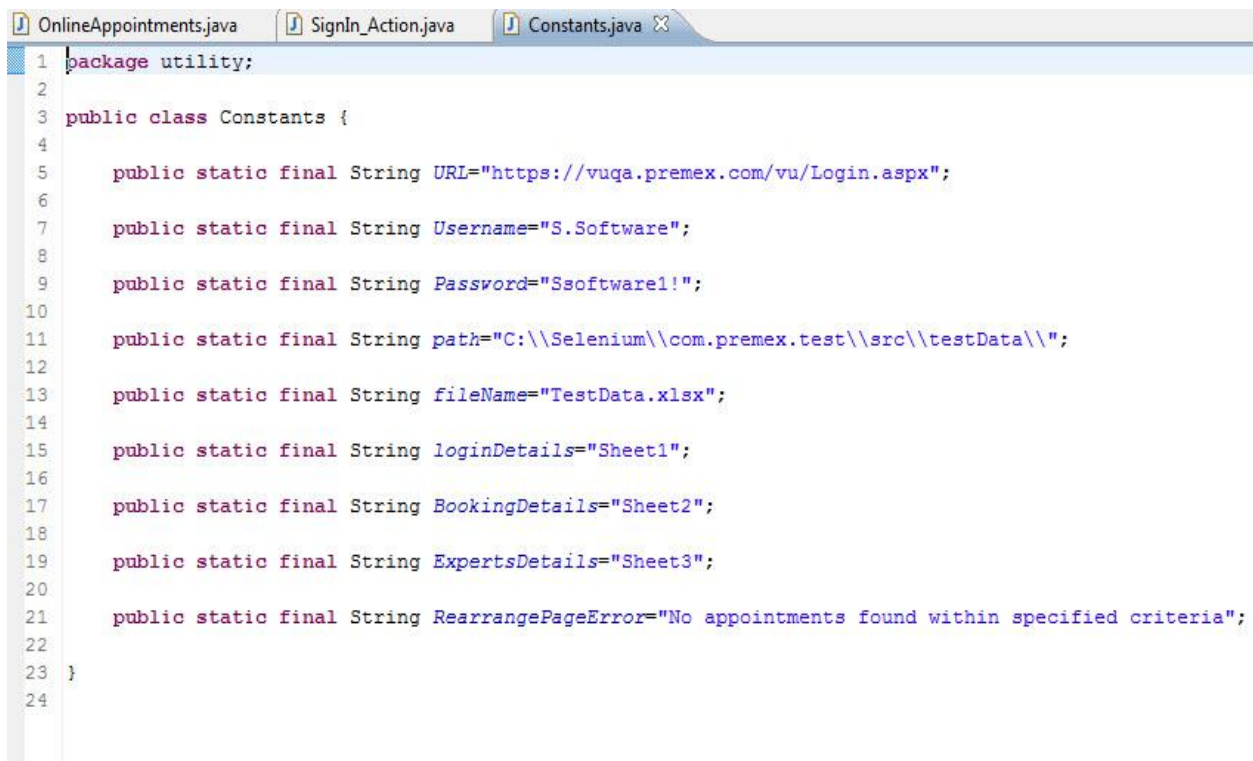


## Constant Variables

Test data can be of two types, fixed or variable. If it is fixed, we can easily hard code the test data in to our test scripts. But sometimes the fixed test data is also used in so many scripts and if it gets changed then it is a huge task to update all the effected test scripts for example the URL of your test application. It remains same but once you shifted to other environment, you need to change it in all of your test scripts. We can easily place the URL in Text file or Excel file outside our test scripts but Java gives us special feature of creating Constants variables.

## Steps

- Create a 'New Package' file and name it as “**utility**”, by right click on the Project and select **New > Package**.
- Create a 'New Class' file, by right click on the above created Package and select **New > Class** and name it as **Constant**.
- Assign keywords in this class to your fixed data for e.g. URL, Username and Password.



```
1 package utility;
2
3 public class Constants {
4
5     public static final String URL="https://vuqa.premex.com/vu/Login.aspx";
6
7     public static final String Username="S.Software";
8
9     public static final String Password="Ssoftware!";
10
11     public static final String path="C:\\Selenium\\com.premex.test\\src\\testData\\";
12
13     public static final String fileName="TestData.xlsx";
14
15     public static final String loginDetails="Sheet1";
16
17     public static final String BookingDetails="Sheet2";
18
19     public static final String ExpertsDetails="Sheet3";
20
21     public static final String RearrangePageError="No appointments found within specified criteria";
22
23 }
24
```

- Constants Variables are declared as **public static**, so that they can be called in any other methods **without instantiate** the class.
- Constant Variables are declared a **final**, so that they cannot be changed during the execution.

## **Data Driven Framework with Apache POI – Excel**

**Data-driven testing (DDT)** is a term used in the testing of computer software to describe testing done using a table of conditions directly as test inputs and verifiable outputs as well as the process where test environment settings and control are not hard-coded. In the simplest form the tester supplies the inputs from a row in the table and expects the outputs which occur in the same row.

### **Reading data from the Excel**

We need a way to open this Excel sheet and read data from it within our Selenium test script. For this purpose, I use the Apache POI library, which allows you to read, create and edit Microsoft Office-documents using Java.

### **Steps**

- Download JAR files of Apache POI and Add Jars to your project library. That's all about configuration of Apache POI with eclipse. Now you are ready to write your test.
- Create a 'New Package' file and name it as '**testData**', by right click on the Project and select **New > Package**. Place all of your test data in this folder (package) whether it is a sql file, excel file or anything.
- Place an **Excel** file in the above created package location and save it as **TestData.xlsx**.
- Fill the data in the excel like below image
- Login details



	A	B	C	D
1	Testcase name	username	password	Result
2	Login	S.Software	Ssoftware1!	
3				
4				
5				

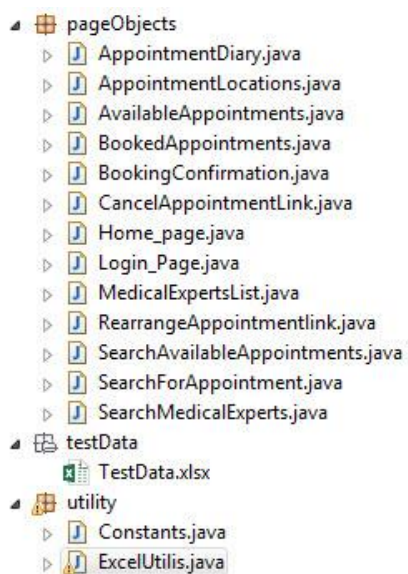
### ➤ Appointment Booking detail

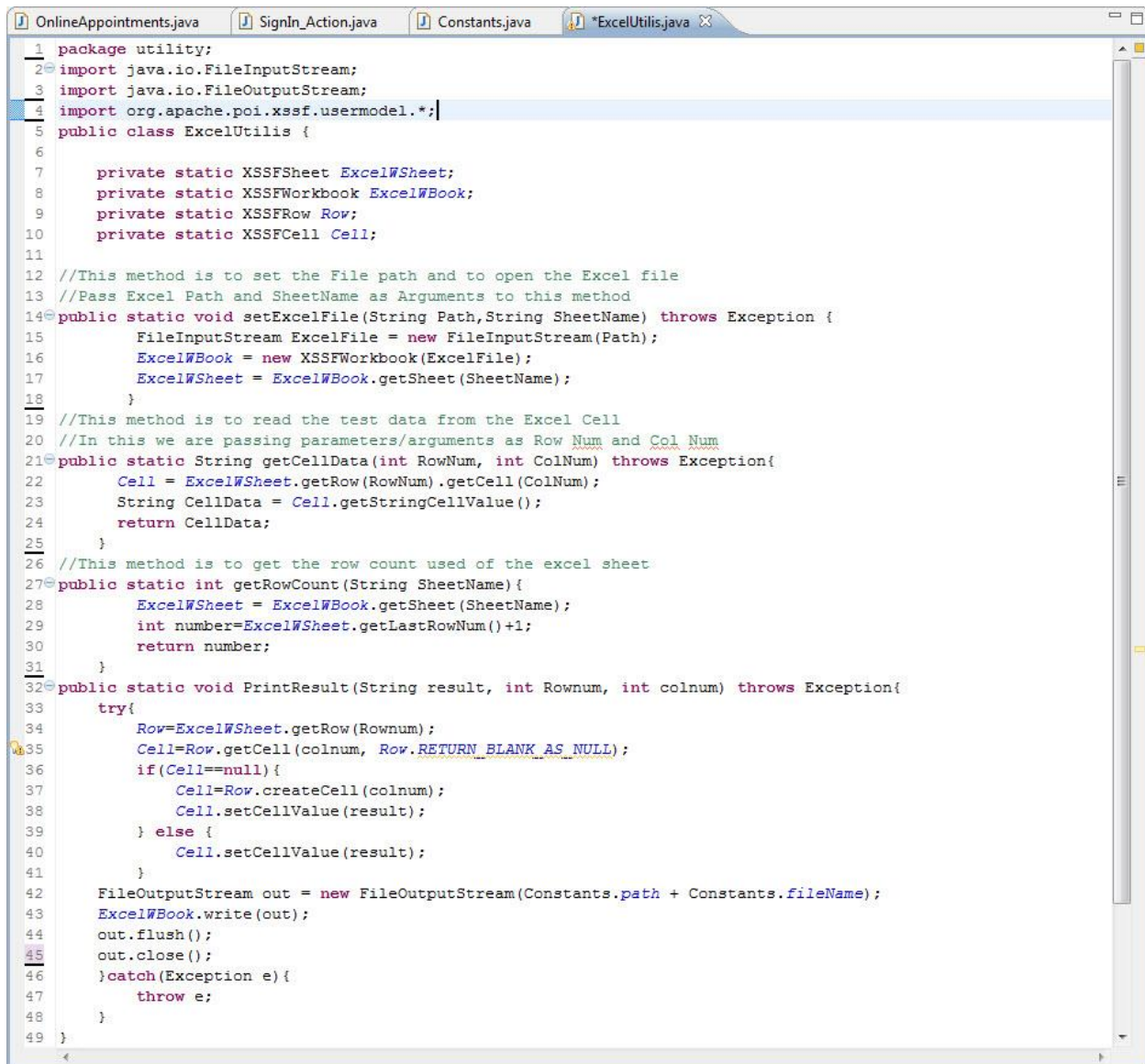
	A	B	C	D	E	F	G	H	I
1	Post code	Earliest Date	Latest Date	FirstName	LastName	CaseReference	Status	Booking Status	Current Satus
2	BL0 0AJ	28/09/2016	16/10/2016	Test	User	test Booking			
3									
4									
5									

### ➤ Appointment date detail

	A	B	C
1	Surname	Appointment Date	AppointmentEndDate
2	Smith	23/09/2016	30/09/2016
3			
4			
5			

- Create a 'New Class' file, by right click on the 'utility' Package and select **New > Class** and name it as '**ExcelUtils**'. First we will write basic read/write methods.
- Your Project explorer window will look like this now.





```
1 package utility;
2 import java.io.FileInputStream;
3 import java.io.FileOutputStream;
4 import org.apache.poi.xssf.usermodel.*;
5 public class ExcelUtilis {
6
7     private static XSSFSheet ExcelWSheet;
8     private static XSSFWorkbook ExcelWBook;
9     private static XSSFRow Row;
10    private static XSSFCell Cell;
11
12    //This method is to set the File path and to open the Excel file
13    //Pass Excel Path and SheetName as Arguments to this method
14    public static void setExcelFile(String Path,String SheetName) throws Exception {
15        FileInputStream ExcelFile = new FileInputStream(Path);
16        ExcelWBook = new XSSFWorkbook(ExcelFile);
17        ExcelWSheet = ExcelWBook.getSheet(SheetName);
18    }
19    //This method is to read the test data from the Excel Cell
20    //In this we are passing parameters/arguments as Row Num and Col Num
21    public static String getCellData(int RowNum, int ColNum) throws Exception{
22        Cell = ExcelWSheet.getRow(RowNum).getCell(ColNum);
23        String CellData = Cell.getStringCellValue();
24        return CellData;
25    }
26    //This method is to get the row count used of the excel sheet
27    public static int getRowCount(String SheetName){
28        ExcelWSheet = ExcelWBook.getSheet(SheetName);
29        int number=ExcelWSheet.getLastRowNum()+1;
30        return number;
31    }
32    public static void PrintResult(String result, int Rownum, int colnum) throws Exception{
33        try{
34            Row=ExcelWSheet.getRow(Rownum);
35            Cell=Row.getCell(colnum, Row.RETURN_BLANK_AS_NULL);
36            if(Cell==null){
37                Cell=Row.createCell(colnum);
38                Cell.setCellValue(result);
39            } else {
40                Cell.setCellValue(result);
41            }
42            FileOutputStream out = new FileOutputStream(Constants.path + Constants.fileName);
43            ExcelWBook.write(out);
44            out.flush();
45            out.close();
46        }catch(Exception e){
47            throw e;
48        }
49    }
```

## TestNG

- ✓ TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use.
- ✓ In simple words TestNG is a tool that help us to organize the tests and help us to produce the test reports.
- ✓ TestNG framework can be used for automation testing with Selenium (web application automation testing tool).

## TestNG Advantages

- ✓ Multiple built in **Annotations** which are easier to use and understand.
- ✓ Test method can be **dependent** to other method.
- ✓ Test cases can be **Grouped** and can be execute separately by groups.
- ✓ TestNG has built in HTML report and XML report generation facility. It has also built in logging facility.

## Steps

- ✓ 1) First step is to Install TestNG. It is easy to install TestNG, as it comes as a plugin for Eclipse IDE.
- ✓ 2) Create a 'New Class' by right click on the '**ExecutionEngine**' package then select **TestNG > Create a TestNG Class** and name it as **OnlineAppointments**.
- ✓ 3) Let's divide the test case in to three parts.
  - **@BeforeMethod**: Launch a new Firefox browser then open <https://vuqa.premex.com/vu/Login.aspx> and login with the Premex user credential.
  - **@Test**: Execute BookAppointment, SearchAppointment, RearrangeAppointment, CancelAppointment and ManageExpertsDiaries actions and Log out.
  - **@AfterMethod**: Close Firefox browser.

```
@BeforeTest
public void beforeTest() throws Exception {
    dr=new FirefoxDriver();
    dr.navigate().to(Constants.URL);
    dr.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    ExcelUtilis.setExcelFile(Constants.path + Constants.fileName, Constants.loginDetails);
    // Using login now
    SignIn_Action.Login_action(dr);
    ExcelUtilis.PrintResult("Pass", 1, 3);
}

@AfterTest
public void afterTest() {
    dr.close();
}
}
```

## ➤ BookAppointment method.

```
public void BookAppointment() throws Exception {
    Home_page.BookAppointment(dr).click();
    ExcelUtilis.setExcelFile(Constants.path + Constants.fileName, Constants.BookingDetails);
    String postcode=ExcelUtilis.getCellData(1, 0);
    String EarlierDate=ExcelUtilis.getCellData(1, 1);
    String latestDate=ExcelUtilis.getCellData(1, 2);
    String FirstName=ExcelUtilis.getCellData(1, 3);
    String lastName=ExcelUtilis.getCellData(1, 4);
    String caseReference=ExcelUtilis.getCellData(1, 5);
    SearchAvailableAppointments.FromPostCode(dr).sendKeys(postcode);
    Select sel= new Select(SearchAvailableAppointments.ExpertType(dr));
    sel.selectByIndex(1);
    Select max= new Select(SearchAvailableAppointments.MaxTravelDistance(dr));
    max.selectByIndex(5);
    SearchAvailableAppointments.EarliestDate(dr).clear();
    SearchAvailableAppointments.EarliestDate(dr).sendKeys(EarlierDate);
    SearchAvailableAppointments.latestDate(dr).clear();
    SearchAvailableAppointments.latestDate(dr).sendKeys(latestDate);
    SearchAvailableAppointments.SearchButton(dr).click();
    AppointmentLocations.ViewAppointment1(dr).click();
    AvailableAppointments.BookAppointment1(dr).click();
    Select title = new Select(BookingConfirmation.Title(dr));
    title.selectByIndex(1);
    BookingConfirmation.FirstName(dr).sendKeys(FirstName);
    BookingConfirmation.SurName(dr).sendKeys(lastName);
    BookingConfirmation.caseReference(dr).sendKeys(caseReference);
    BookingConfirmation.ConfirmBooking(dr).click();
    TimeUnit.SECONDS.sleep(3);
    String ps=dr.getPageSource();
    if(ps.contains("Warning")){
        BookingConfirmation.ConfirmBooking(dr).click();
    }
    BookingReference =BookingConfirmation.BookingReference(dr).getText();
    System.out.println("Booking done successfully");
    System.out.println("Booking Reference ID is:" +BookingReference);
    System.out.println("*****");
    ExcelUtilis.PrintResult("Booked", 1, 6);
    BookingConfirmation.HomeButton(dr).click();
}
}
```

## ➤ SearchAppointment method.

```
@Test(enabled=true, priority=1)
public static void SearchAppointment() throws Exception{
    Home_page.SearchAppointments(dr).click();
    SearchForAppointment.BookingReferenceNo(dr).sendKeys(BookingReference);
    SearchForAppointment.SearchButton(dr).click();
    String ReferenceID=SearchForAppointment.BookingID(dr).getText();
    if(ReferenceID.equalsIgnoreCase(BookingReference)){
        System.out.println("Searching done successfully");
        System.out.println("Searched Booking Reference ID is:" +BookingReference);
        System.out.println("*****");
        SearchForAppointment.Home(dr).click();
    } else {
        System.out.println("Try again... !");
    }
}
}
```



## ➤ RearrangeAppointment method.

```
@Test(enabled=true, priority=2)
public static void RearrangeAppointment() throws Exception{
    ExcelUtilis.setExcelFile(Constants.path + Constants.fileName, Constants.BookingDetails);
    String EarlierDate=ExcelUtilis.getCellData(1, 1);
    Home_page.RearrangeAppointment(dr).click();
    BookedAppointments.BookingReferenceNo(dr).sendKeys(BookingReference);
    BookedAppointments.Search(dr).click();
    RearrangeAppointmentlink.RearrangeAppointment(dr).click();
    String postcode=ExcelUtilis.getCellData(1, 0);
    SearchAvailableAppointments.FromPostCode(dr).sendKeys(postcode);
    SearchAvailableAppointments.SearchButton(dr).click();
    String PS=dr.getPageSource();
    if(PS.contains(Constants.RearrangePageError)){
        AppointmentLocations.SearchAgain(dr).click();
        SearchAvailableAppointments.EarliestDate(dr).clear();
        SearchAvailableAppointments.EarliestDate(dr).sendKeys(EarlierDate);
        SearchAvailableAppointments.SearchButton(dr).click();
    }
    AppointmentLocations.ViewAppointment1(dr).click();
    AvailableAppointments.BookAppointment2(dr).click();
    BookingConfirmation.ConfirmBooking(dr).click();
    BookingReference =BookingConfirmation.BookingReference(dr).getText();
    System.out.println("Appointment date rearranged done successfully");
    System.out.println("Rearranged Booking Reference ID is:" +BookingReference);
    System.out.println("*****");
    ExcelUtilis.PrintResult("Rearranged", 1, 7);
    BookingConfirmation.HomeButton(dr).click();
}
}
```

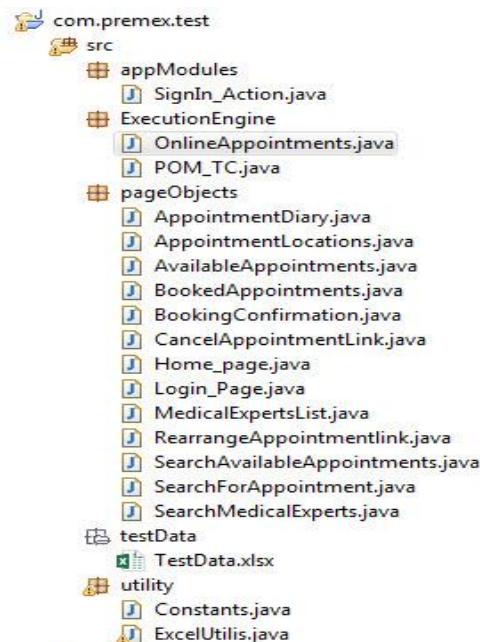
## ➤ CancelAppointment method.

```
@Test(enabled=true, priority=3)
public static void CancelAppointment() throws Exception{
    Home_page.CancelAppointment(dr).click();
    BookedAppointments.BookingReferenceNo(dr).sendKeys(BookingReference);
    BookedAppointments.Search(dr).click();
    String PS=dr.getPageSource();
    if(PS.contains(BookingReference)){
        CancelAppointmentLink.CancelAppointment(dr).click();
        CancelAppointmentLink.ConfirmCancel(dr).click();
        BookingReference =BookingConfirmation.BookingReference(dr).getText();
        System.out.println("Cancel Appointment done successfully");
        System.out.println("Cancelled Booking Reference ID is:" +BookingReference);
        System.out.println("*****");
        ExcelUtilis.PrintResult("Booking Cancelled", 1, 8);
        TimeUnit.SECONDS.sleep(2);
        CancelAppointmentLink.HomeButton(dr).click();
    }
}
}
```

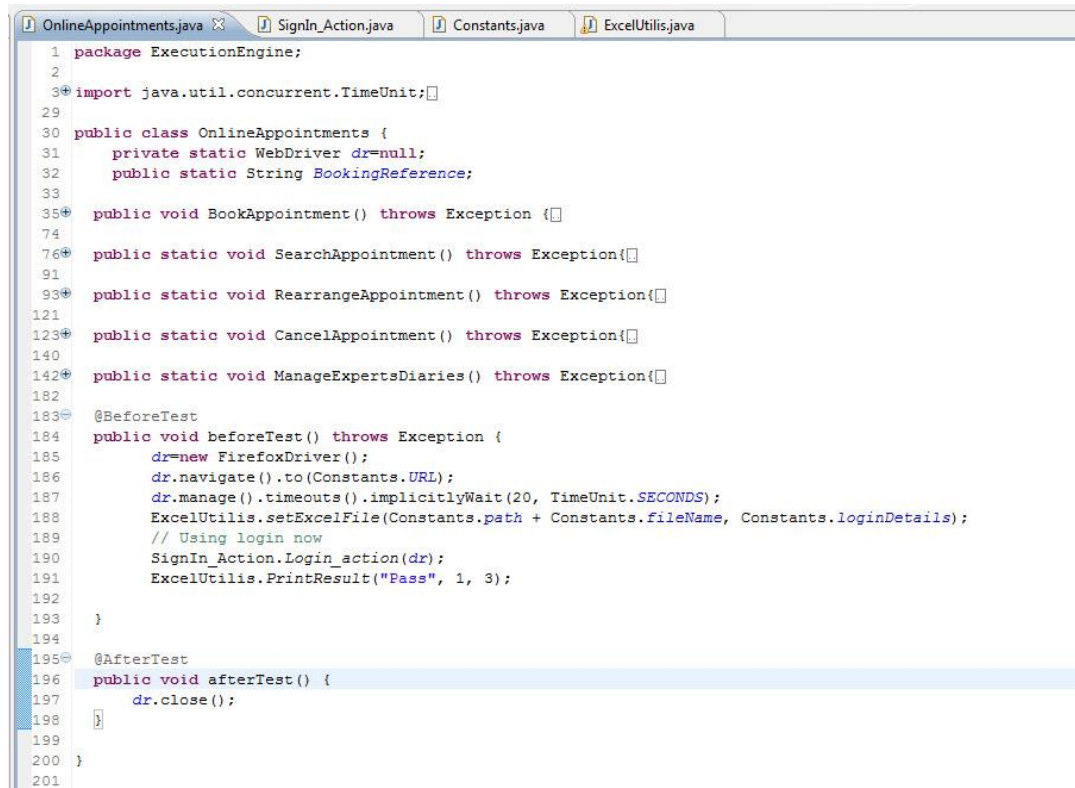
## ➤ ManageExpertsDiaries method.

```
@Test(enabled=true, priority=4)
public static void ManageExpertsDiaries() throws Exception{
    Home_page.manageExpertDiaries(dr).click();
    dr.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);
    ExcelUtilis.setExcelFile(Constants.path + Constants.fileName, Constants.ExpertsDetails);
    String Surname=ExcelUtilis.getCellData(1, 0);
    String AppointmentDate=ExcelUtilis.getCellData(1, 1);
    String AppointmentEndDate=ExcelUtilis.getCellData(1, 2);
    SearchMedicalExperts.Surname(dr).sendKeys(Surname);
    SearchMedicalExperts.Search(dr).click();
    MedicalExpertsList.manageDiary(dr).click();
    String ExpertName=AppointmentDiary.ExpertName(dr).getText();
    String ExpertLocation=AppointmentDiary.ExpertLocation(dr).getText();
    System.out.println("Expert name is :"+ExpertName);
    System.out.println("Expert Location is :"+ExpertLocation);
    System.out.println("*****");
    AppointmentDiary.AddAvailability(dr).click();
    AppointmentDiary.Appointmentdate(dr).clear();
    AppointmentDiary.Appointmentdate(dr).sendKeys(AppointmentDate);
    Select ST =new Select(AppointmentDiary.StartTimeHour(dr));
    ST.selectByIndex(4);
    Select SM =new Select(AppointmentDiary.StartTimeMinutes(dr));
    SM.selectByIndex(6);
    Select Duration =new Select(AppointmentDiary.Duration(dr));
    Duration.selectByValue("45");
    AppointmentDiary.OkButton(dr).click();
    TimeUnit.SECONDS.sleep(2);
    System.out.println("Appointment added successfully");
    System.out.println("*****");
    AppointmentDiary.DeleteAvailability(dr).click();
    AppointmentDiary.StartDate(dr).clear();
    AppointmentDiary.StartDate(dr).sendKeys(AppointmentDate);
    AppointmentDiary.EndDate(dr).clear();
    AppointmentDiary.EndDate(dr).sendKeys(AppointmentEndDate);
    AppointmentDiary.SelectAllCheckBox(dr).click();
    AppointmentDiary.Delete(dr).click();
    System.out.println("Appointment Deleted successfully");
    System.out.println("*****");
    AppointmentDiary.HomeButton(dr).click();
}
```

## ➤ Your Project explorer window will look like this now.

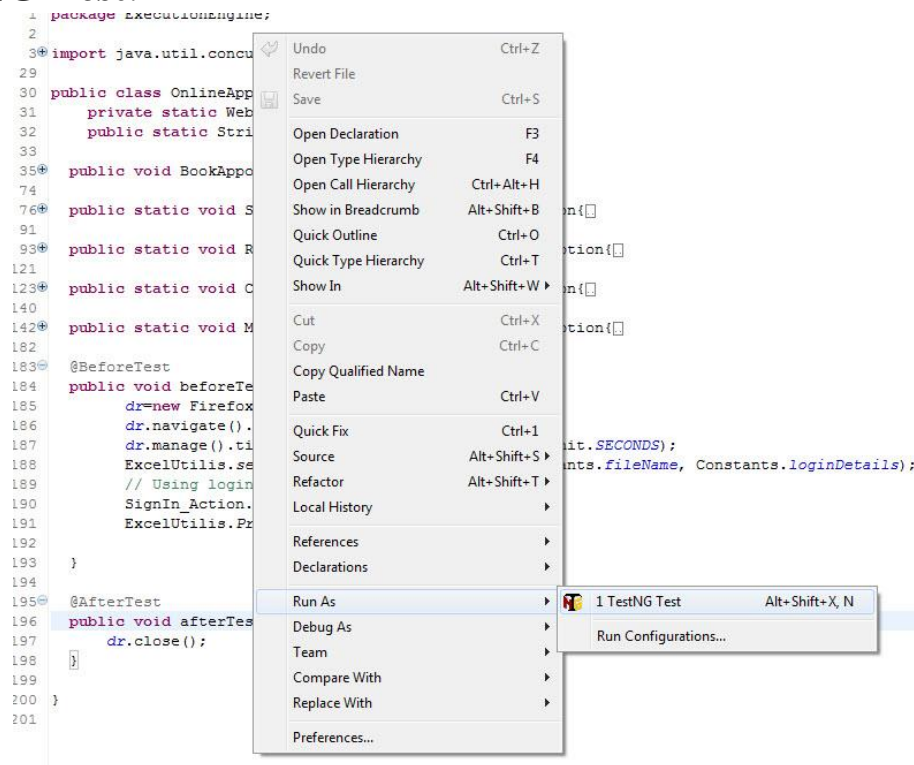


➤ **OnlineAppointments class file** will look like this now.



```
1 package ExecutionEngine;
2
3 import java.util.concurrent.TimeUnit;
4
29 public class OnlineAppointments {
30     private static WebDriver dr=null;
31     public static String BookingReference;
32
33     public void BookAppointment() throws Exception {}
34
35     public static void SearchAppointment() throws Exception{}
36
37     public static void RearrangeAppointment() throws Exception{}
38
39     public static void CancelAppointment() throws Exception{}
40
41     public static void ManageExpertsDiaries() throws Exception{}
42
43     @BeforeTest
44     public void beforeTest() throws Exception {
45         dr=new FirefoxDriver();
46         dr.navigate().to(Constants.URL);
47         dr.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
48         ExcelUtilis.setExcelFile(Constants.path + Constants.fileName, Constants.loginDetails);
49         // Using login now
50         SignIn_Action.Login_action(dr);
51         ExcelUtilis.PrintResult("Pass", 1, 3);
52     }
53
54     @AfterTest
55     public void afterTest() {
56         dr.close();
57     }
58 }
```

✓ Run the test by right click on the test case script and select **Run As > TestNG Test**.



```
1 package ExecutionEngine;
2
3 import java.util.concu
4
29 public class OnlineApp
30     private static Web
31     public static Stri
32
33     public void BookAppo
34
35     public static void S
36
37     public static void R
38
39     public static void C
40
41     public static void M
42
43     @BeforeTest
44     public void beforeTe
45         dr=new Firefox
46         dr.navigate().t
47         dr.manage().ti
48         ExcelUtilis.se
49         // Using login
50         SignIn_Action.Pr
51         ExcelUtilis.Pr
52     }
53
54     @AfterTest
55     public void afterTes
56         dr.close();
57     }
58 }
```



- ✓ Give it few minutes to complete the execution, once it is finished the results will look like this in the **Console** window.

```
Results of running class OnlineAppointments Console
<terminated> OnlineAppointments [TestNG] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Sep 7, 2016 11:34:16 AM)
[TestNG] Running:
  C:\Users\sakthivel\AppData\Local\Temp\testng-eclipse-1118845769\testng-customsuite.xml

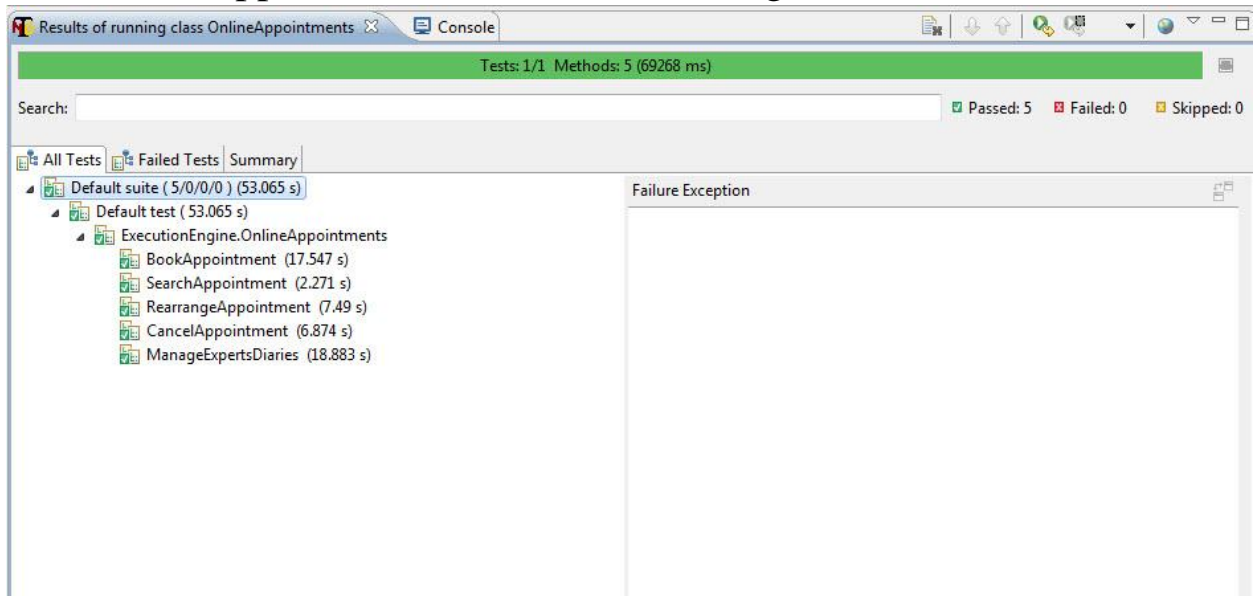
*****
Login done successfully
*****
Booking done successfully
Booking Reference ID is:A50137470
*****
Searching done successfully
Searched Booking Reference ID is:A50137470
*****
Appointment date rearranged done successfully
Rearranged Booking Reference ID is:A50137481
*****
Cancel Appointment done successfully
Cancelled Booking Reference ID is:A50137481
*****
Expert name is :Dr Alexander P Smith
Expert Location is :PREMEX HOUSE, BL6 6SX
*****
Appointment added successfully
*****
Appointment Deleted successfully
*****
PASSED: BookAppointment
PASSED: SearchAppointment
PASSED: RearrangeAppointment
PASSED: CancelAppointment
PASSED: ManageExpertsDiaries

=====
  Default test
  Tests run: 5, Failures: 0, Skips: 0
=====

=====
Default suite
Total tests run: 5, Failures: 0, Skips: 0
=====
```



- ✓ Click on the **Results of TestNG** tab. It will display the total passed, failed and skipped test with time taken during the execution.



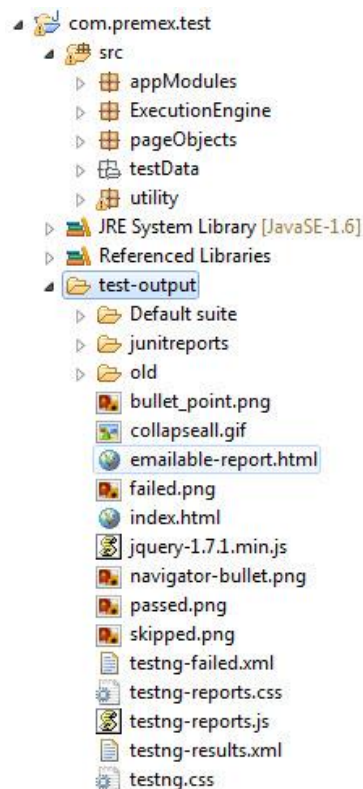
- ✓ It displayed 'Passed: 5'. This means test is successful and Passed.
- ✓ There are 3 sub tabs. "All Tests", "Failed Tests" and "Summary". Just click "Summary" to see what is there.

The screenshot shows the 'Results of running class OnlineAppointments' window. At the top, a green status bar indicates 'Tests: 1/1 Methods: 5 (69268 ms)'. Below this is a search bar and summary statistics: 'Passed: 5', 'Failed: 0', and 'Skipped: 0'. The 'Tests' tab is selected, showing a table with the following data:

Test name	Time (seconds)	Class count	Method count
Default test	53.065	1	5

Below the 'Tests' tab is the 'Excluded methods' section, which is currently empty.

- ✓ TestNG also produce HTML reports. To access those reports go to your **Project** folder and open **test-output** folder.
- ✓ Open '**emailable-report.html**', as this is a html report open it with browser.



Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	5	0	0	69,141		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
ExecutionEngine.OnlineAppointments	BookAppointments	1473228273314	17547
	CancelAppointments	1473228300628	6874
	ManualCancelAppointments	1473228307503	18883
	RetrieveAppointments	1473228293136	7490
	SearchAppointments	1473228290863	2271

- ✓ TestNG also produce '**index.html**' report and it resides in the same **test-output** folder.
- ✓ This reports gives the link to all the different component of the TestNG reports like **Groups & Reporter Output**.
- ✓ On clicking these will display detailed descriptions of execution. In the advance chapter of TestNG we will go through each of the TestNG topics.

The screenshot displays a web browser window showing a TestNG report. The browser's address bar indicates the file path: `file:///C:/Selenium/com.premex.test/test-output/index.html#SearchAppointment`. The report is titled "Test results" and shows "1 suite".

The left sidebar, under "All suites", highlights the "Default suite". It provides summary information:

- Info:** C:\Users\sakthivel\AppData\Local\Temp\testng-eclipse-1118845769\testng-customsuite.xml, 1 test, 0 groups.
- Results:** 5 methods, 5 passed. Passed methods (hide):
  - BookAppointment
  - CancelAppointment
  - ManageExpertsDiaries
  - RearrangeAppointment
  - SearchAppointment

The main content area shows a table of test methods, all of which passed:

Test Method
BookAppointment
CancelAppointment
ManageExpertsDiaries
RearrangeAppointment
SearchAppointment

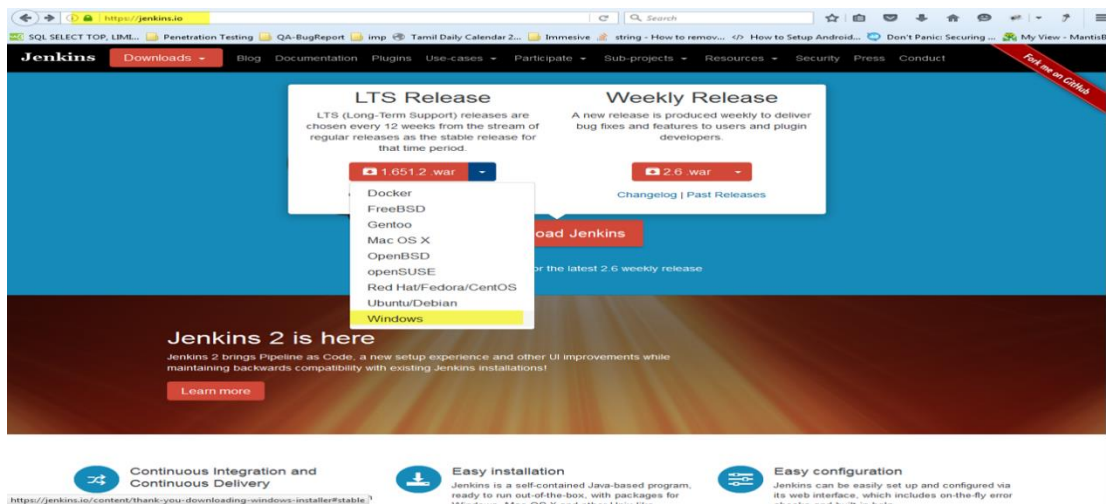
# Jenkins integration with Selenium

## WebDriver

- ✓ Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on.
- ✓ It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.
- ✓ How to integrate Jenkins with selenium?
  - 1- Download Jenkins
  - 2- Configure Jenkins for Running Build
  - 3- Execute Selenium build using Jenkins
  - 4- Schedule Jobs in Jenkins to run periodically

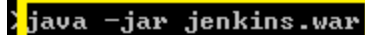
## Download Jenkins

- ✓ Step 1- Open your web browser and then Navigate to Below URL <http://jenkins-ci.org> this is the official website of Jenkins
- ✓ Step 2- Now download Jenkins.war /Jenkins.zip file and save into desktop or any other location depends on your choice

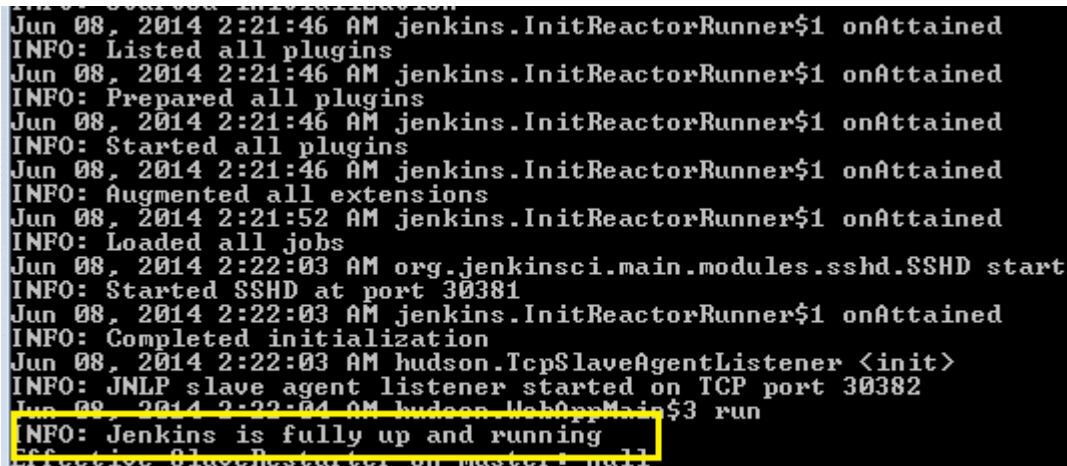


## Configure Jenkins for Selenium

- ✓ Step 1- Go to location where Jenkins.war is available.
- ✓ Step 2- Open Command prompt known as CMD and navigate till project home directory and Start Jenkins server
- ✓ Start- cmd > (location) > java -jar jenkins.war



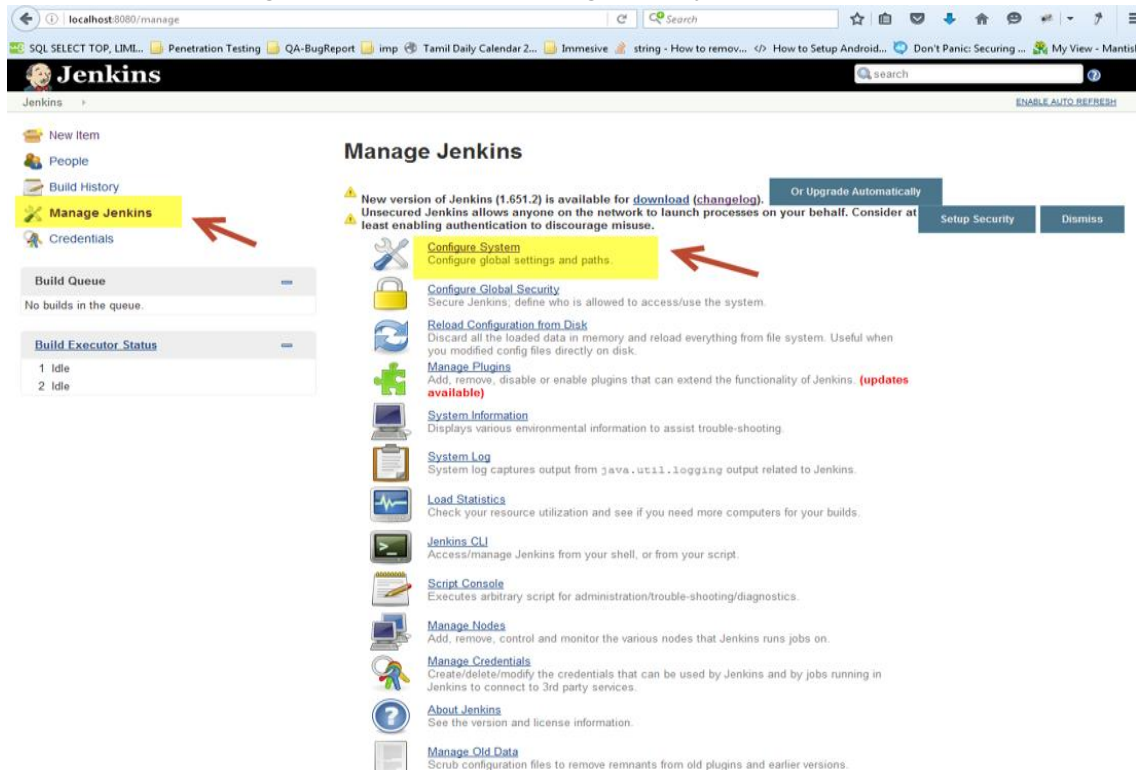
```
java -jar jenkins.war
```



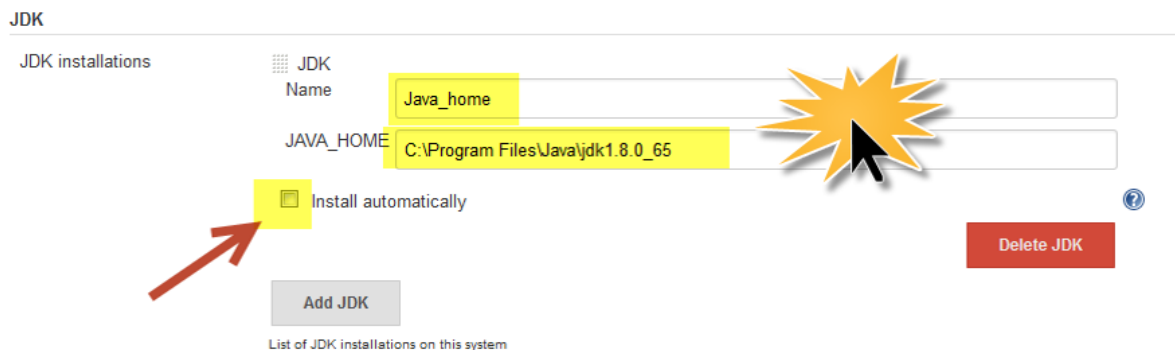
```
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Prepared all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Augmented all extensions
Jun 08, 2014 2:21:52 AM jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Jun 08, 2014 2:22:03 AM org.jenkinsci.main.modules.sshd.SSHD start
INFO: Started SSHD at port 30381
Jun 08, 2014 2:22:03 AM jenkins.InitReactorRunner$1 onAttained
INFO: Completed initialization
Jun 08, 2014 2:22:03 AM hudson.TcpSlaveAgentListener <init>
INFO: JNLP slave agent listener started on TCP port 30382
Jun 08, 2014 2:22:04 AM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
Effective SlaveRestart on master: null
```

- ✓ Once Jenkins server is up and running, you will get above success message.
- ✓ Step 3- By default Jenkins runs on 8080 port. Open any browser and type the url <http://localhost:8080>
- ✓ Now Jenkins is up and running so now we have to configure Jenkins so that we can execute our test case via Jenkins.

- ✓ Step 4- Once Jenkins is running so we are almost done but before moving to create build we need to configure Jenkins so that Jenkins can identify other tools as well like Java.
- ✓ Click on > Manage Jenkins > Configure System.



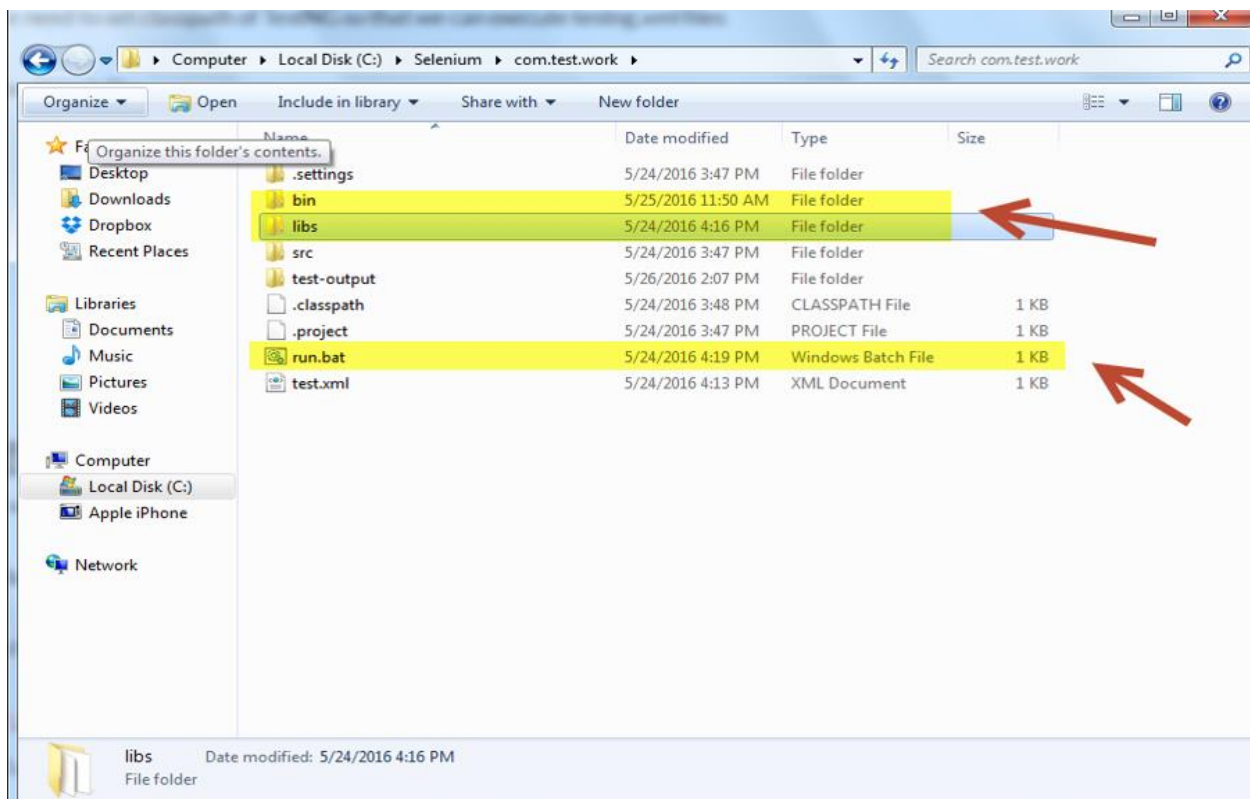
- Navigate to JDK section and Click on Add JDK button
- Uncheck Install automatically check box so Jenkins will only take java which we have mention above.
- Give the name as JAVA\_HOME and Specify the JDK path
- Once done click on save and apply. Congrats, your Jenkins is configured now.



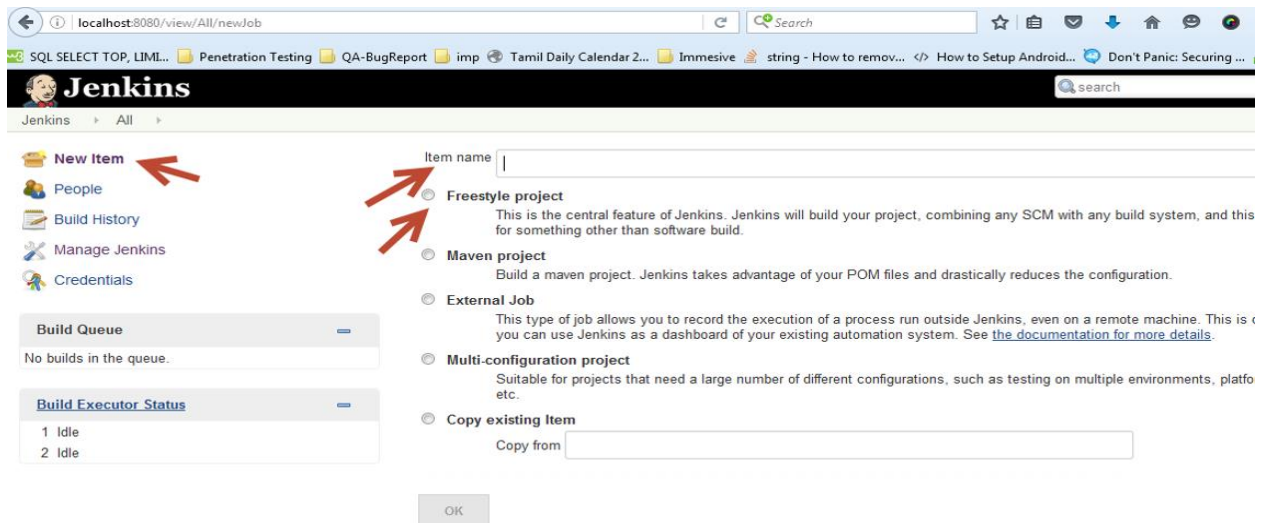


## Execute Selenium build using Jenkins

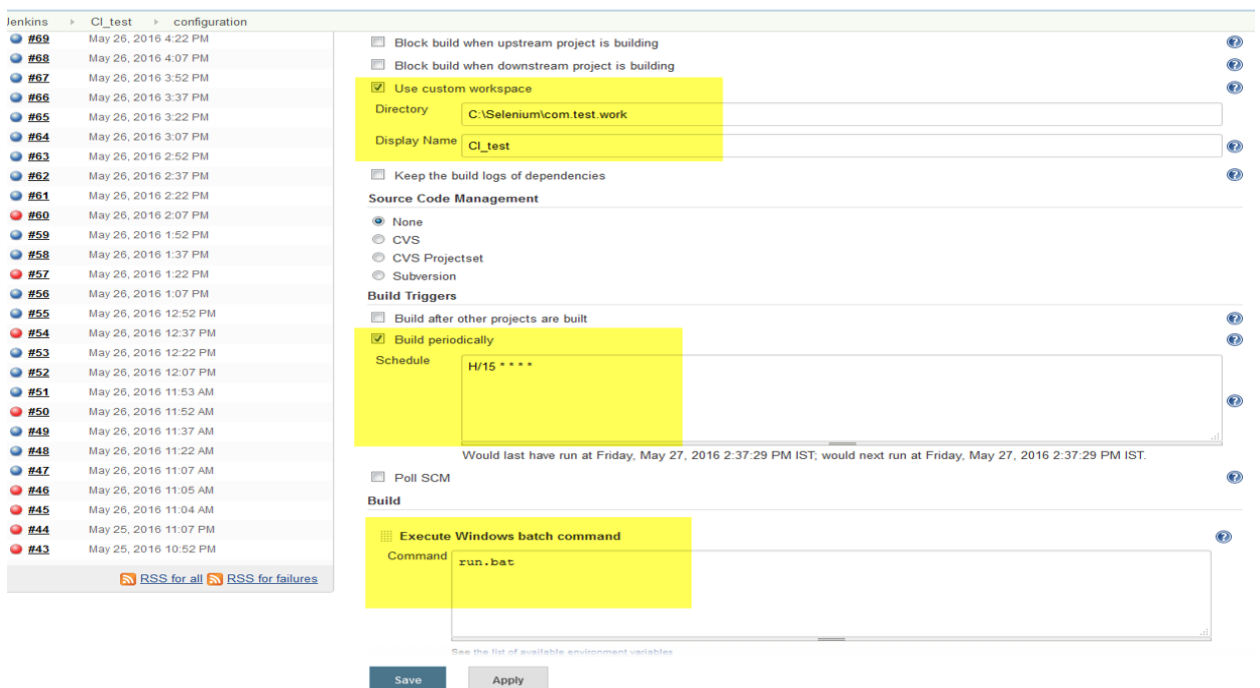
- Step 1- Create a batch file first then we will add the same batch file to Jenkins.
  - ✓ To create batch file we need to set classpath of TestNG so that we can execute testng.xml files
  - ✓ Create a folder libs and add all the selenium jar files.
  - ✓ Open notepad and type the below command and save as .bat file –
    - `java -cp bin;libs/* org.testng.TestNG test.xml`



- **Step 2-** Create a job in Jenkins which will execute our build. Open Jenkins on browser (type <http://localhost:8080>)
- Click on new item.
- Give the Job-Name, select Build a free-style software project and Click on OK button.



- Navigate to Advanced Project Options > Check the use custom workspace > in directory we will specify the project home directory
- Important part now specify the Add Build step > Click on Execute Windows batch command
- In the section please specify the batch file which we created and click on Apply and save.





- Step 3- you can finally run the Build > Click on Build now option
- Step 4- Check Build history and Console output and verify the output

Jenkins

Jenkins
> CI\_test
>

↑ Back to Dashboard
🔍 Status
📄 Changes
📁 Workspace
🏗 Build Now
🚫 Delete Project
⚙ Configure

Jenkins
> Premex\_CI
> #45

```

*****
Login done successfully
*****
[Invoker 191382150] Invoking ExecutionEngine.OnlineAppointments.BookAppointment
Booking done successfully
Booking Reference ID is:A50137769
*****
[Invoker 191382150] Invoking ExecutionEngine.OnlineAppointments.SearchAppointment
Searching done successfully
Searched Booking Reference ID is:A50137769
*****
[Invoker 191382150] Invoking ExecutionEngine.OnlineAppointments.RearrangeAppointment
Appointment date rearranged done successfully
Rearranged Booking Reference ID is:A50137770
*****
[Invoker 191382150] Invoking ExecutionEngine.OnlineAppointments.CancelAppointment
Cancel Appointment done successfully
Cancelled Booking Reference ID is:A50137770
*****
[Invoker 191382150] Invoking ExecutionEngine.OnlineAppointments.ManageExpertsDiaries
Expert name is :Dr Alexander P Smith
Expert Location is :PREMEX HOUSE, BL6 6SX
*****
Appointment added successfully
*****
Appointment Deleted successfully
*****
[Invoker 191382150] Keeping method OnlineAppointments.afterTest()[pri:0, instance:ExecutionEngine.OnlineA
for class null
[Invoker 191382150] Invoking @AfterTest OnlineAppointments.afterTest()[pri:0,
instance:ExecutionEngine.OnlineAppointments@1134affc]
===== Invoked methods
OnlineAppointments.beforeTest()[pri:0, instance:ExecutionEngine.OnlineAppointments@1134affc] 288665596
OnlineAppointments.BookAppointment()[pri:0, instance:ExecutionEngine.OnlineAppointments@1134affc] 288
OnlineAppointments.SearchAppointment()[pri:1, instance:ExecutionEngine.OnlineAppointments@1134affc] 2
OnlineAppointments.RearrangeAppointment()[pri:2, instance:ExecutionEngine.OnlineAppointments@1134affc
OnlineAppointments.CancelAppointment()[pri:3, instance:ExecutionEngine.OnlineAppointments@1134affc] 2
OnlineAppointments.ManageExpertsDiaries()[pri:4, instance:ExecutionEngine.OnlineAppointments@1134affc
OnlineAppointments.afterTest()[pri:0, instance:ExecutionEngine.OnlineAppointments@1134affc] 288665596
=====
Creating C:\Selenium\jenkins\premix\test-output\Suite\Test.html
Creating C:\Selenium\jenkins\premix\test-output\Suite\Test.xml
PASSED: BookAppointment
PASSED: SearchAppointment
PASSED: RearrangeAppointment
PASSED: CancelAppointment
PASSED: ManageExpertsDiaries

=====
Test
Tests run: 5, Failures: 0, Skips: 0
=====

```