

Test Automation Using Selenium

Test Automation

- Test automation is the use of software
 - *To set test preconditions.*
 - *To control the execution of tests.*
 - *To compare the actual outcomes to predicted outcomes.*
 - *To report the Execution Status.*
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.

Why and When To Automate?

- Frequent regression testing
- Repeated test case Execution is required
- User Acceptance Tests
- Faster Feedback to the developers
- Reduce the Human Effort
- Test same application on multiple environments

Test Automation Tools

- Quick Test Professional By HP
- Rational Functional Tester By Rational (IBM Company)
- Silk Test By Borland
- Test Complete By Automated QA
- QA Run (Compuware)
- Watir (Open Source)
- Selenium (Open Source)
- Sahi (Open Source)

Selenium

- Selenium is a robust set of tools that supports rapid development of test automation for web-based applications.
- Selenium provides a rich set of testing functions specifically geared to the needs of testing of a web application.
- Selenium operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior.

Selenium Features

- Supports Cross Browser Testing. The Selenium tests can be run on multiple browsers.
- Allows scripting in several languages like Java, C#, PHP and Python.
- Assertion statements provide an efficient way of comparing expected and actual results.
- Inbuilt reporting mechanism.

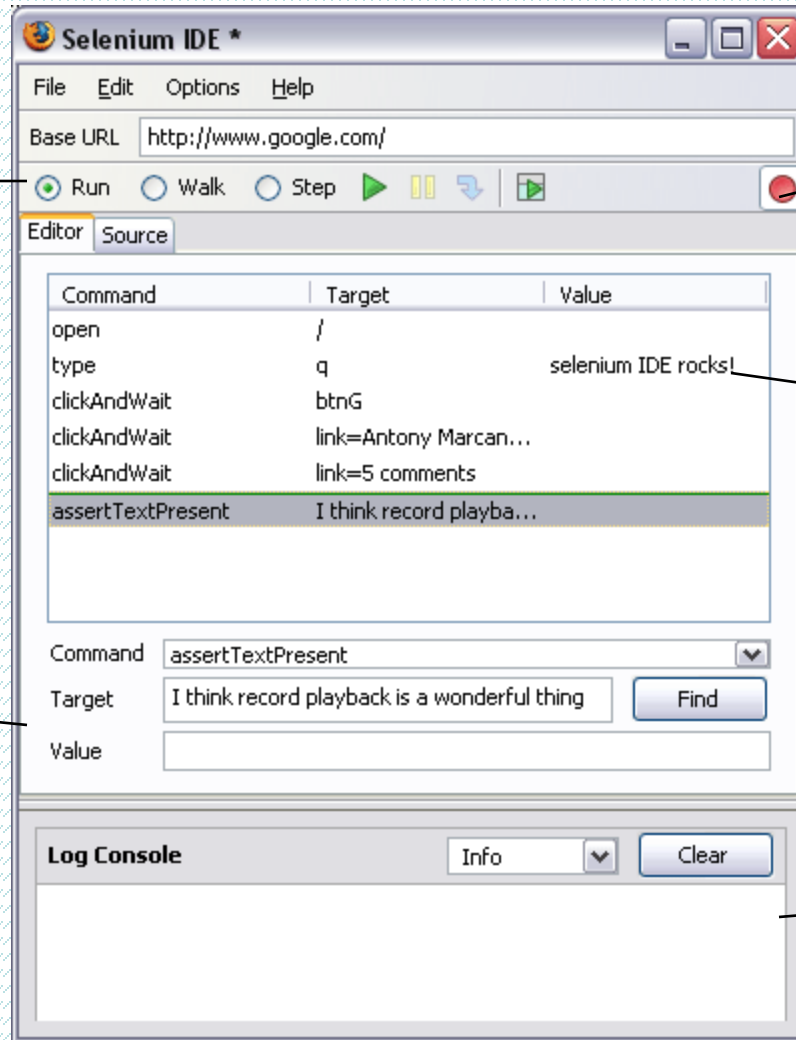
Selenium Components

- Selenium IDE
- Selenium Remote Control
- Selenium Web Driver
- Selenium Grid

Selenium IDE

- Selenium IDE is an integrated development environment for Selenium tests.
- It is implemented as a Firefox extension, and allows you to record, edit, and replay the test in firefox
- Selenium IDE allows you to save tests as HTML, Java, Ruby scripts, or any other format
- It allows you to automatically add assertions to all the pages.
- Allows you to add selenese commands as and when required

Selenium IDE - UI



Replay
Toolbar

Start and Stop
Recording

Selenese
Script
Editor

Accessor
Area

Selenium Log

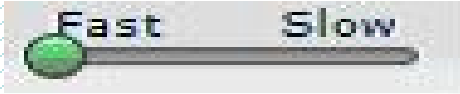
Recording a Selenium Test Case

- Open Firefox that has the IDE installed
- Open the base URL of the application to record.
- Keep the application in a common base state.
- Go To Tools → Selenium IDE and the IDE will be opened
- Now perform the operations on the application as you are testing the application.
- Once you are done with the recording click on the stop recording button and save the test case through the file menu. By default it will be saved as a selenese script (HTML format)

General Selenese Commands

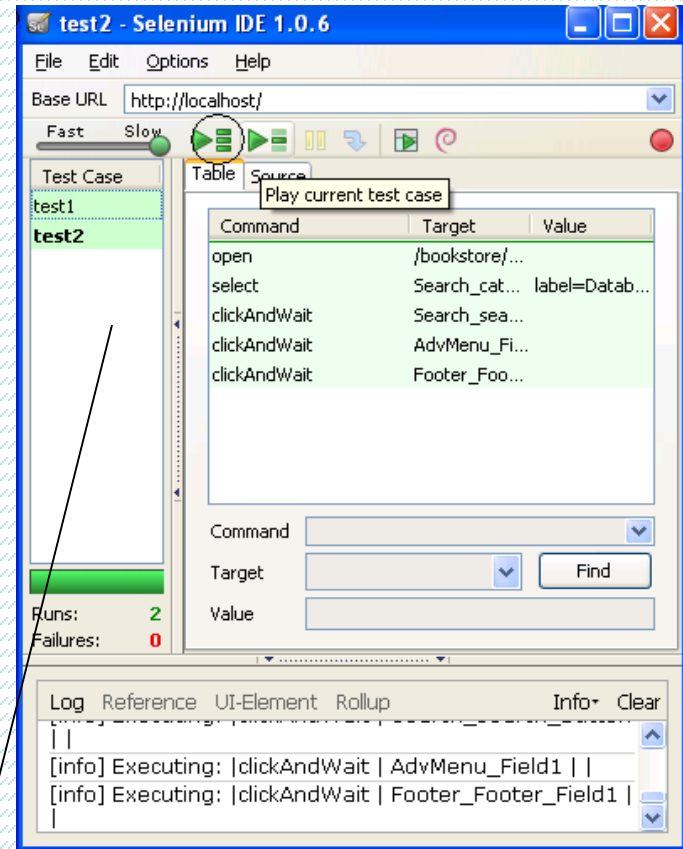
- clicking a link - *click* or *clickAndWait* commands
- entering values - *type* command
- selecting options from a drop-down listbox - *select* command
- clicking checkboxes or radio buttons - *click* command

Running Your First Selenium Script

- Make sure the application is in the common base state.
- Click on the run button. Here you can also control the speed of the execution using the  toolbar
- Once the test is run you can view the test log in the bottom of the IDE window

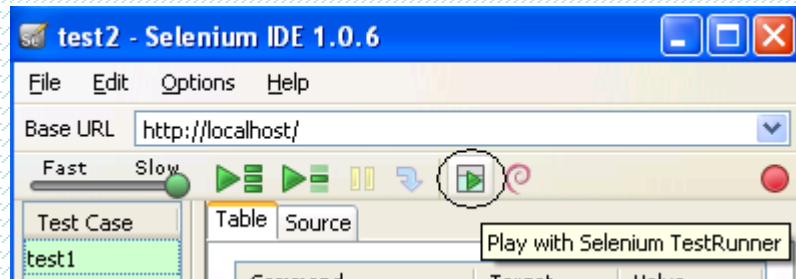
Creating a Test Suite

- In the Selenium IDE you can create any number of test cases and save them as test suite.
- To Run the test Suite click on the “Play entire test suite” button as shown below.



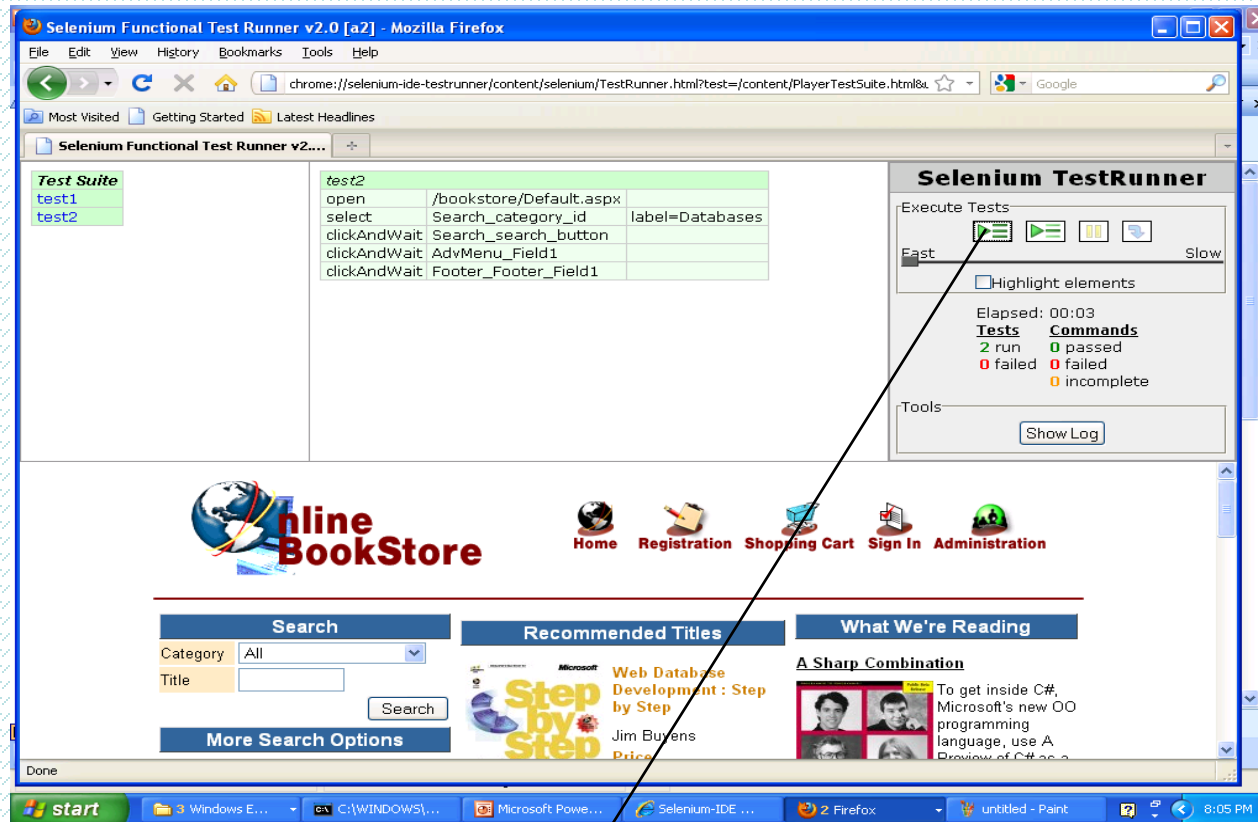
Playing The test Suite with Test Runner

- Test Runner allows you to run the test case in a browser loaded with the Selenium-Core TestRunner.
- Test runner is invoked by clicking the below Shown button in the IDE



On Clicking the Test Runner Button you will the window as seen in the next slide

Test Runner



Click this button to run all the tests

Running Options

Run a Test Case

Click the Run button to run the currently displayed test case.

Run a Test Suite

Click the Run All button to run all the test cases in the currently loaded test suite.

Stop and Start

The Pause button can be used to stop the test case while it is running. The icon of this button then changes to indicate the Resume button. To continue click Resume.

Stop in the Middle

You can set a breakpoint in the test case to cause it to stop on a particular command. This is useful for debugging your test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.

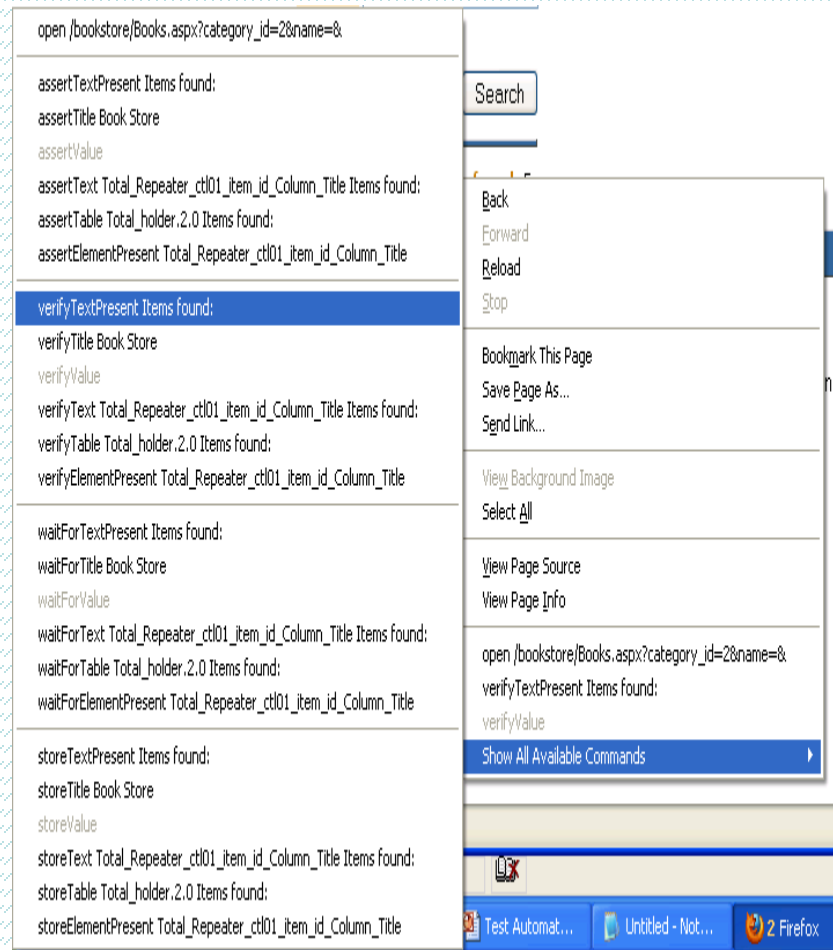
Adding Assertions to the Script

- Selenese allows multiple ways of checking for UI elements.
- Verifications and assertions are used to check if
 - an element is present somewhere on the page?
 - specific text is somewhere on the page?
 - specific text is at a specific location on the page?
- Verifications and assertions are not one and the same.
- If an assertion fails, the script will be aborted but if a verification fails the script will continue.

Verification Commands

verifyTextPresent

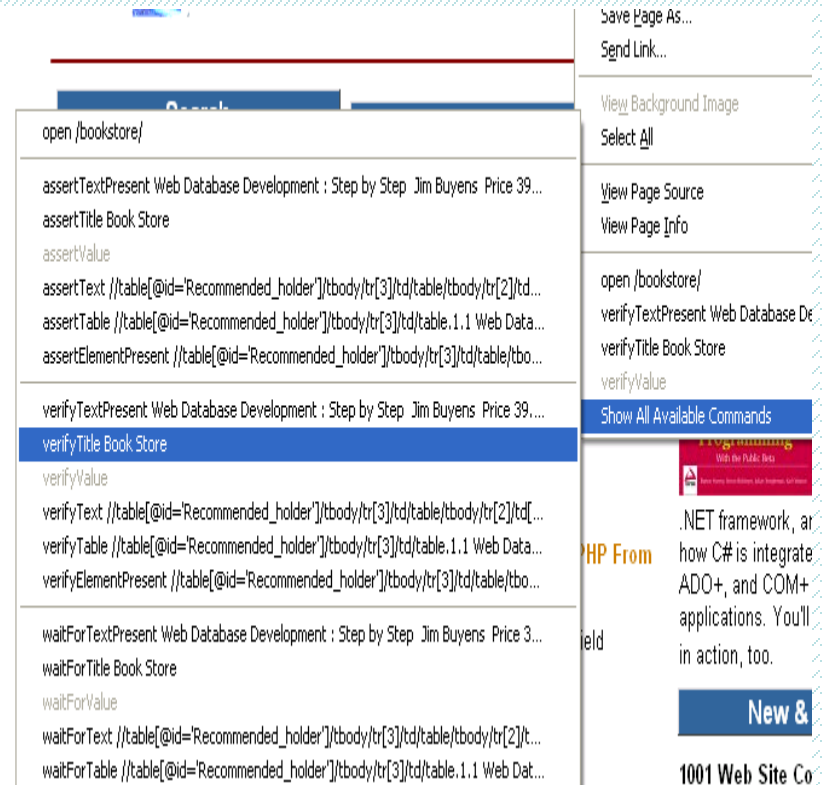
- This command is used to check if a particular text is present in a page or not.
- To add this command , While recording the test steps right click on the text item that you want verify. Once right clicked you can find an option “Show all commands”. On Clicking it you will find an option “verifyTextPresent”, select it



Verification Commands

verifyTitle

- This command is used to check if the page title is correct or not.
- To add this command , While recording the test steps right click any where on the page that you want verify. Once right clicked you can find an option “Show all commands”. On Clicking it you will find an option “verifyTitle”, select it



Verification Commands

verifyElementPresent

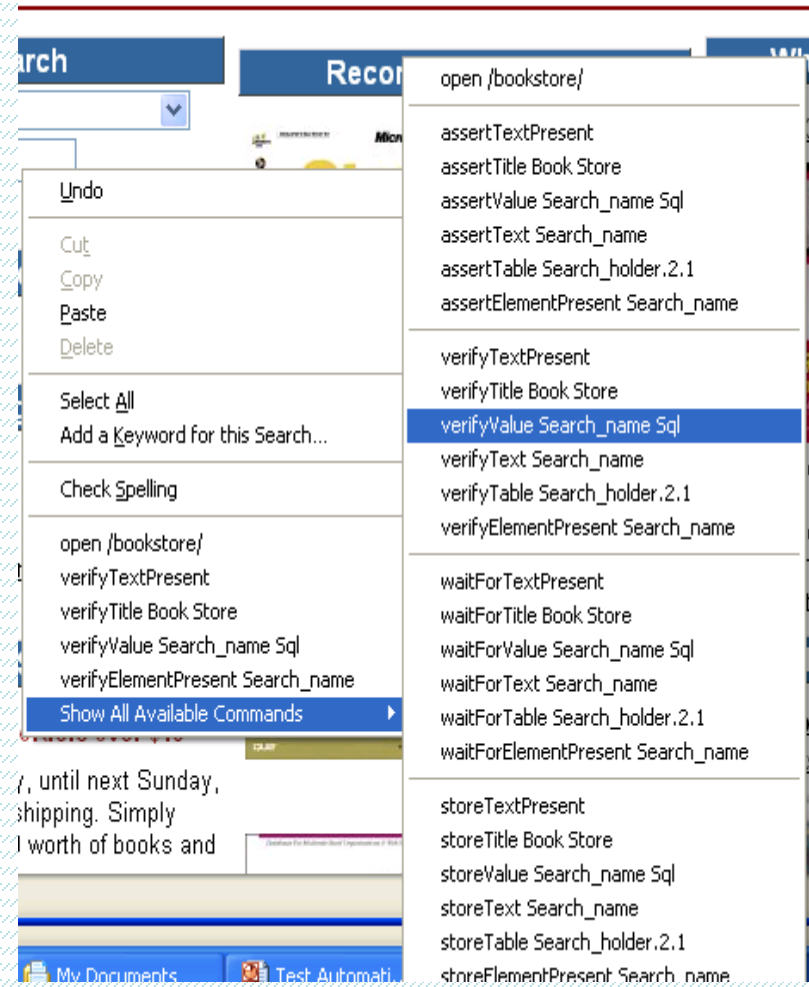
- This command is used to verify if a page element is present in the page or not.
- To add this command , While recording the test steps right click any element on the page that you want verify. Once right clicked you can find an option “Show all commands”. On Clicking it you will find an option “verifyElementPresent”, select it



Verification Commands

verifyValue

- This method is used to check if edit box has particular value or if the check box is on. Basically this method returns the value of present in the object.
- To add this command , While recording the test steps right click any element on the page that you want verify the value of. Once right clicked you can find an option “Show all commands”. On Clicking it you will find an option “verifyValue”, select it.



Assertions

- Assertions are same as Verifications. The only difference is, if the assertions fail the script will abort. But the script will continue run in case a verification point fails.
- The steps for inserting the assertions is same as that of verification point.
- While recording Right Click → Show all commands → select an assertion.

Assertion Statements

- **assertTextPresent**

This will assert if the text is present in the page.

- **assertText**

This will assert if a particular element is having the particular text.

- **assertTitle**

This will assert if the page is having a proper title.

- **assertValue**

This will assert if a Text box or check box has a particular value

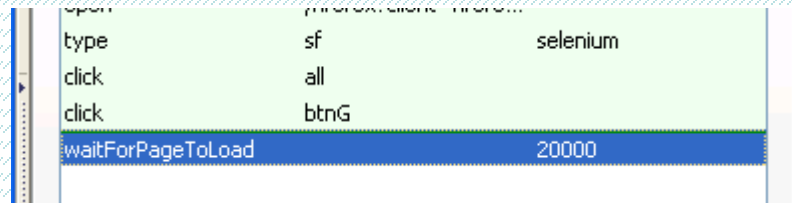
- **assertElementPresent**

This will assert if a particular UI Element is present in the page.

Selenium WaitFor Commands

waitForPageToLoad

- This command will make the script to wait till the page loads.
- Syntax is `waitForPageToLoad(timeout)`; Time out is the maximum time the script will wait for the page to load.

A screenshot of a Selenium IDE command table. The table has three columns: Command, Target, and Value. The first four rows are: 'open' with target 'firefox:chrome://test/' and value 'chrome://test/'; 'type' with target 'sf' and value 'selenium'; 'click' with target 'all' and value ''; 'click' with target 'btnG' and value ''. The fifth row, 'waitForPageToLoad', is highlighted in blue and has a value of '20000'.

open	firefox:chrome://test/	chrome://test/
type	sf	selenium
click	all	
click	btnG	
waitForPageToLoad		20000

Other waitFor Commands

waitForAlert

This command will wait for the alert message to appear

waitForTable

This command will wait for the Web table to completely load in the page

waitForTitle

This command will for the page Title to appear on the browser.

Other waitFor commands

Selenium has several other wait command like `waitForText`, `waitForPopup` and so on. These commands are generically called Synchronization commands

Store Commands

- Store commands are used to fetch the values from the application and store it in a variable. These variables can be used later for validation purposes.
- The Store command can be used to retrieve the page title, text from the page and other attributes from the application.

Echo Command

- Echo command is used to print the value in to the selenium IDS log.
- When printing a variable use `${var}`
- There are some limitations for this methods this has to be used with caution

The screenshot displays the Selenium IDE interface. At the top, a table lists the commands in the test suite:

Command	Target	Value
open	/search?client=firefo...	
storeTextPresent	Selenium web applica... var	
echo	\${var}	

Below the table, the configuration for the selected 'echo' command is shown:

- Command: `echo` (selected from a dropdown)
- Target: `${var}` (entered in a text field)
- Value: (empty text field)
- A 'Find' button is located next to the Target field.

At the bottom, the 'Log' tab is active, showing the execution history:

- [info] Executing: |storeTextPresent | Selenium web application testing system | var |
- [info] Executing: |echo | \${var} | |
- [info] echo: true

Limitations of Selenium IDE

- Can run the test only on Firefox
- No Programming login (like loops, conditional statements) can be applied
- Selenium IDE can execute scripts created in Selenese only.
- It is difficult to use Selenium IDE for checking complex test cases involving dynamic contents

Selenium RC

- A solution to cross browser testing.
- A server, written in Java and so available on all the platforms.
- Acts as a proxy for web requests from them.
- Client libraries for many popular languages.
- Bundles Selenium Core and automatically loads into the browser

Installing Selenium RC

Software Required

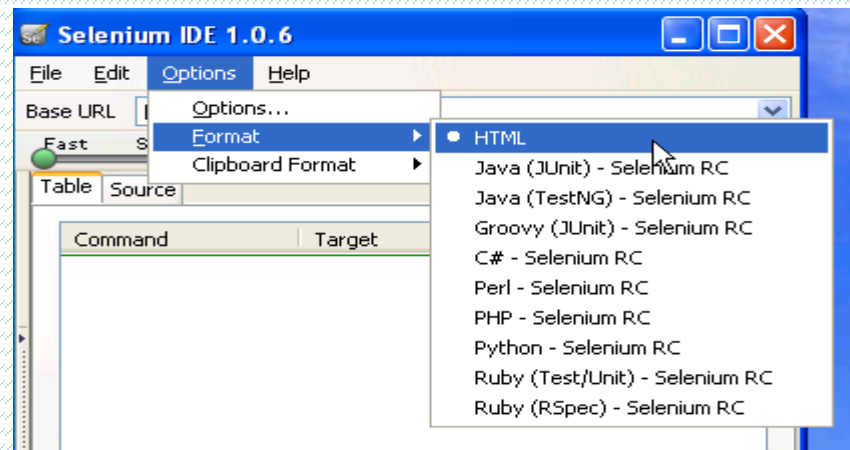
- JDK 1.6 , selenium-remote-control-1.0.3 (this can be downloaded from <http://seleniumhq.org/download/>)

Installation Procedure

- Selenium RC is simply a jar file and to run it we need java installed. (JDK 1.6 is preferred)
- Once the Java is installed just unzip the selenium-remote-control-1.0.3. zip which was downloaded from the selenium site to a directory.

Selenium Test Automation Process

- First Generate the Script using selenium IDE in the firefox IDE
- Once the Scripts are recorded add assertions where ever required
- Now format the Selenese test into the language of your choice. Please refer to the Image

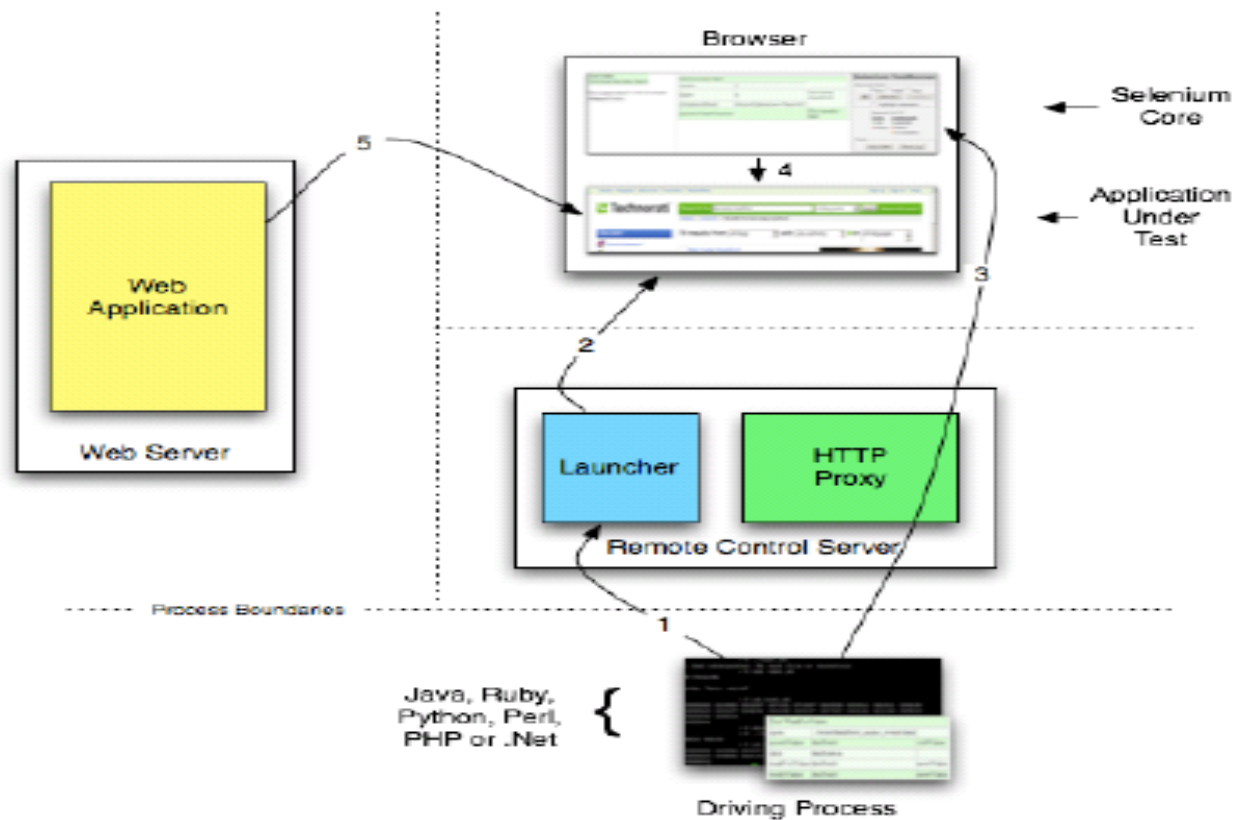


Selenium Test Automation Process

- Once the Selenese script is converted into your preferred language you can you can run them using Selenium Server.
- For running the script you also need the client driver for that particular language.
- To enhance the script we will require IDE like netbeans or Eclipse IDE
- To Integrate the script and run them as a suite we will require build integration tools like Maven or Ant.

How Selenium Works

How it works



Selenium Test Case Development Using Java


Required Software

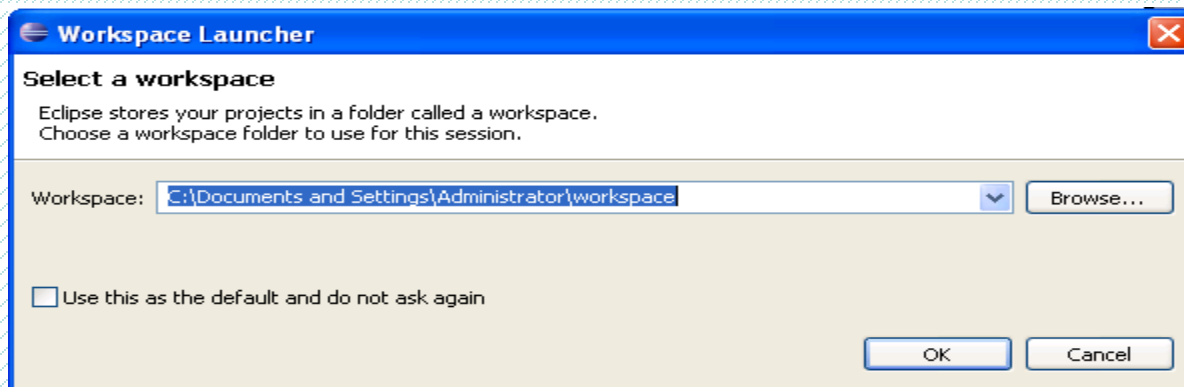
- Selenium Server jar , Selenium Java Client Driver jar , JDK 1.6 +, Eclipse (or any other IDE), Junit jar and testng jar

Setting Up an Eclipse Project for Selenium Automation

- Eclipse is an open source community whose projects are focused on building an extensible development platform for building Java applications and frameworks. Eclipse is one of the best Java IDE and as a matter of fact Eclipse is much more than a Java IDE.
- We can configure a selenium project in eclipse and even run the scripts from eclipse.
- Using eclipse its easy to enhance the recorded script. We can add power to the recorded script by parameterizing the test inputs and even validate the back values.
- Eclipse also allows us to write reusable code for efficient test automation.

Installing Eclipse

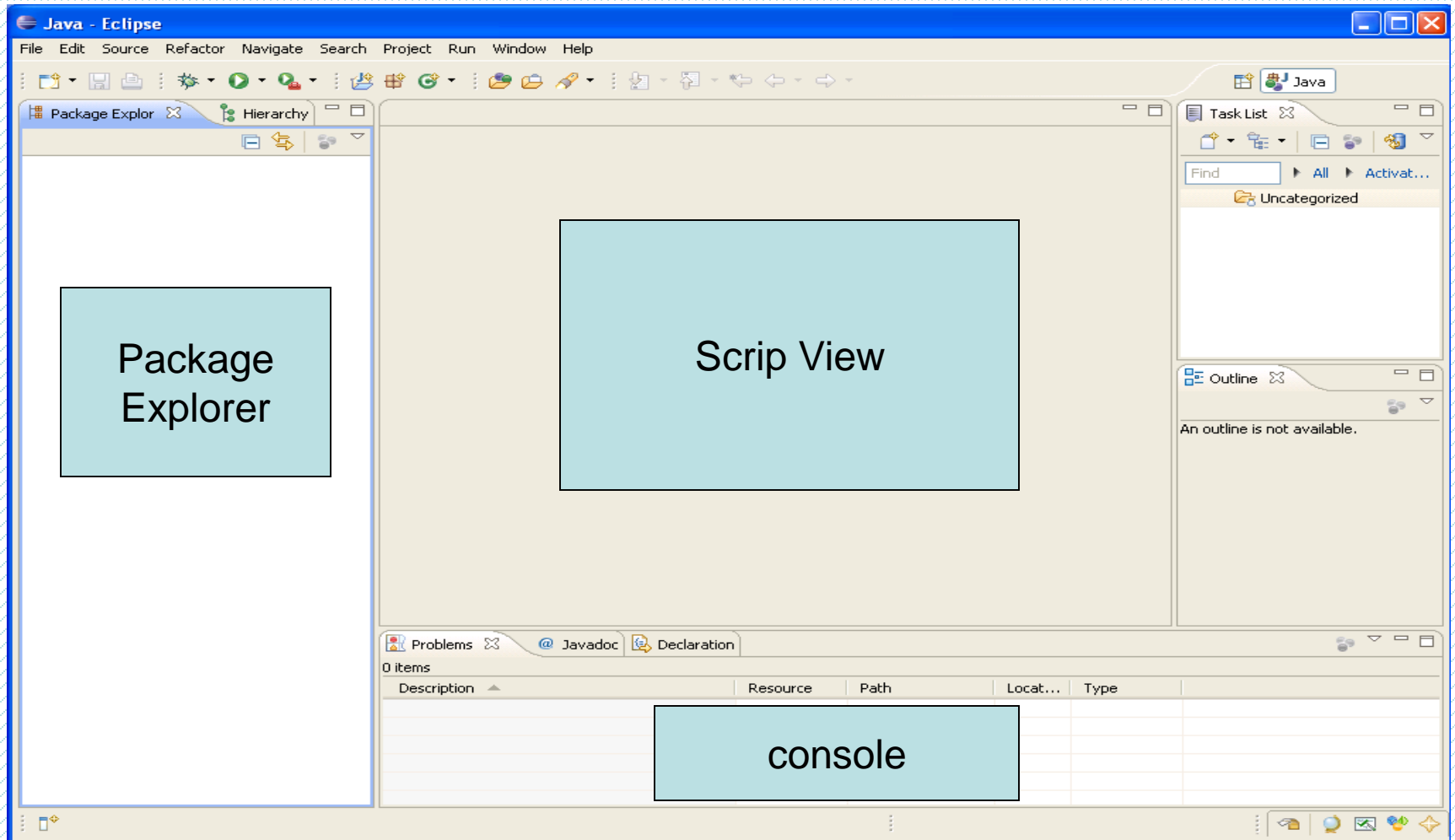
- Download the “Eclipse IDE for Java Developers” from the <http://www.eclipse.org/downloads/> page.
- Unzip the downloaded zip file from the above site into a directory.
- Once the unzipping is over open the folder and double click on the  icon and it will open a dialog box as shown below.



Setting up the workspace

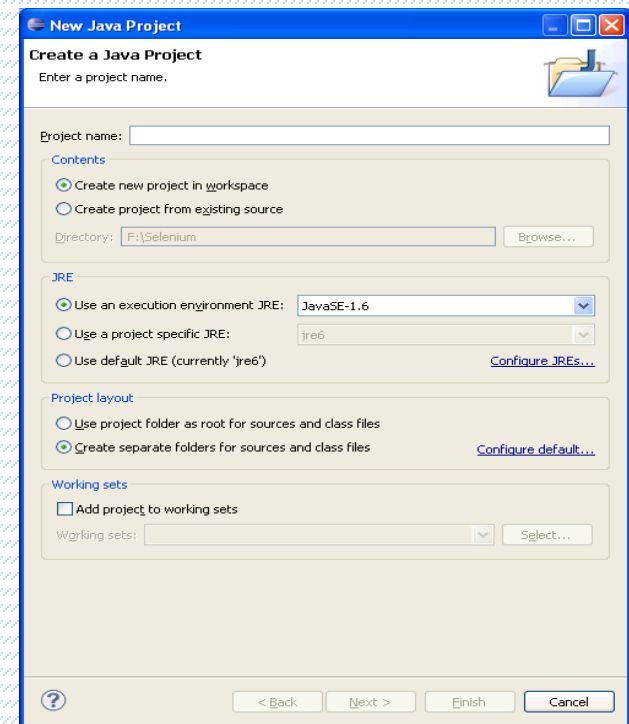
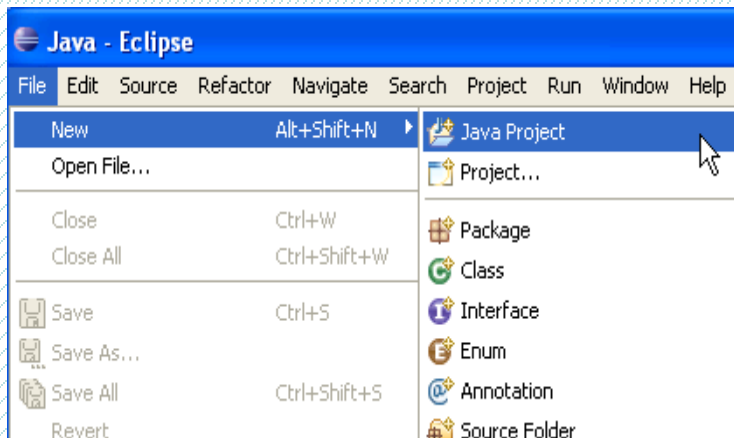
- Create a folder say (selenium) in any one of the directory and change the workspace location to the directory created by you. Then click ok button.
- On doing so you will see the eclipse welcome screen if you are doing it for the first time. On the welcome screen click on the workbench icon to open the project explorer.
- When you to try to open the eclipse from next time it will directly show the project explorer as shown in the next slide.

Eclipse IDE



Create a Project In eclipse

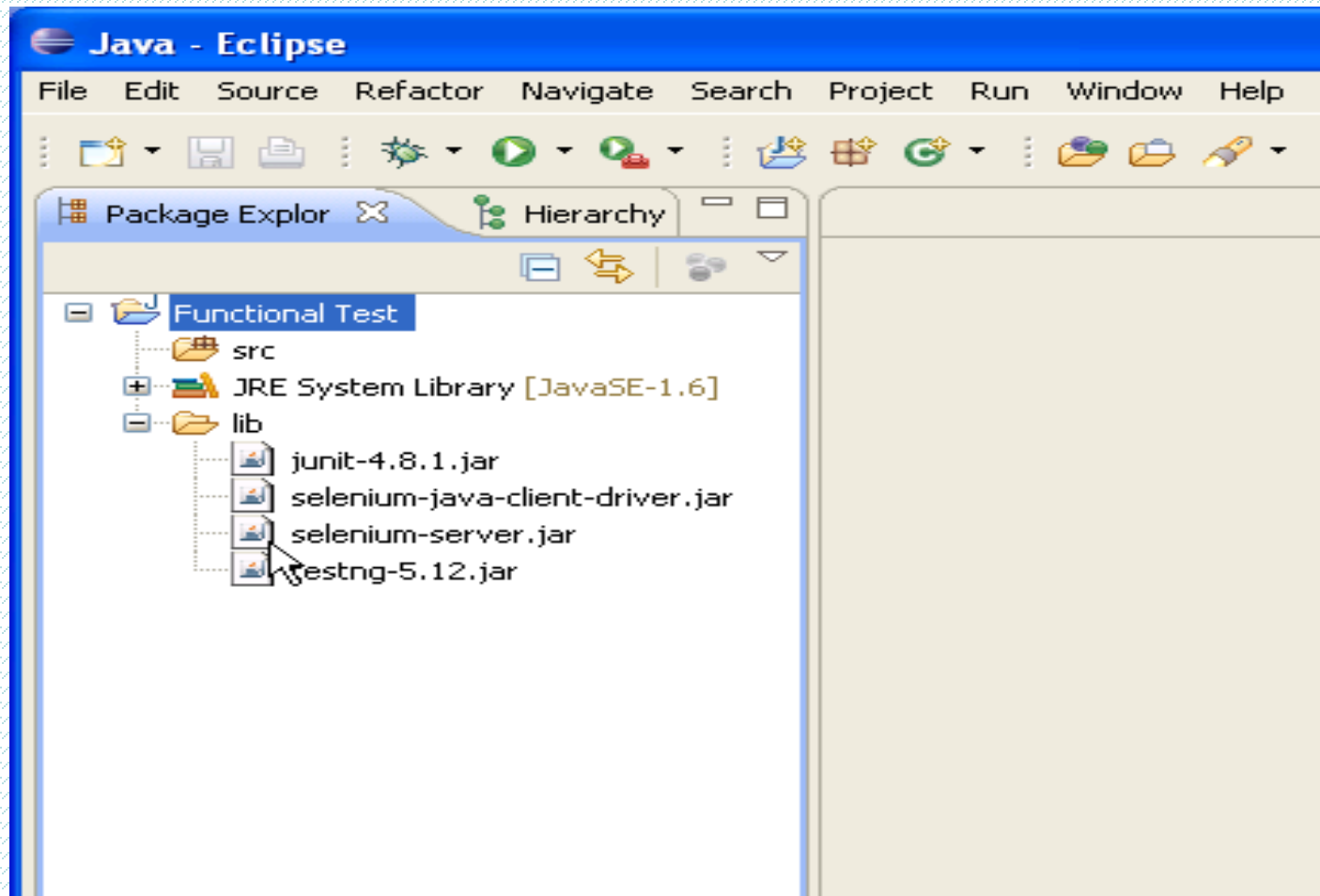
- File → New → Java Project
- Specify the project name and click finish



Adding the required Jars

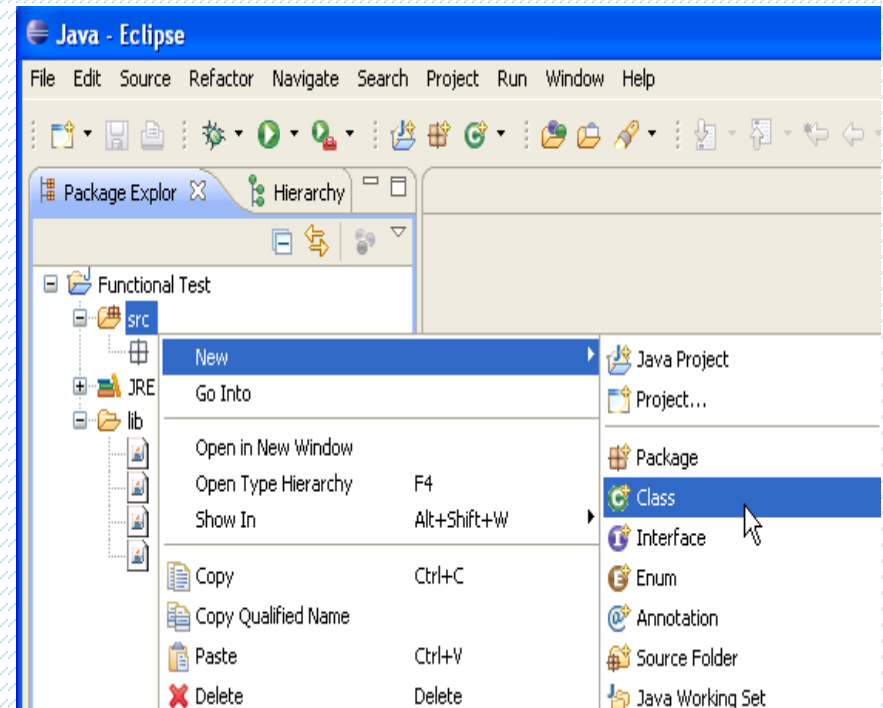
- Open the folder contain the selenium project that you have just created. Inside that folder create another folder with name “lib”
- Inside the lib folder place the following jar files.
 - ❖ junit-4.8.1.jar
 - ❖ selenium-java-client-driver.jar
 - ❖ selenium-server.jar
 - ❖ testng-5.12.jars
- ❖ After placing the jar file in lib come back to the eclipse and click on the project explorer strip and press F5. You should see the all the jar files under the lib folder in the project explorer as shown in the Image in next slide.

Selenium Libraries



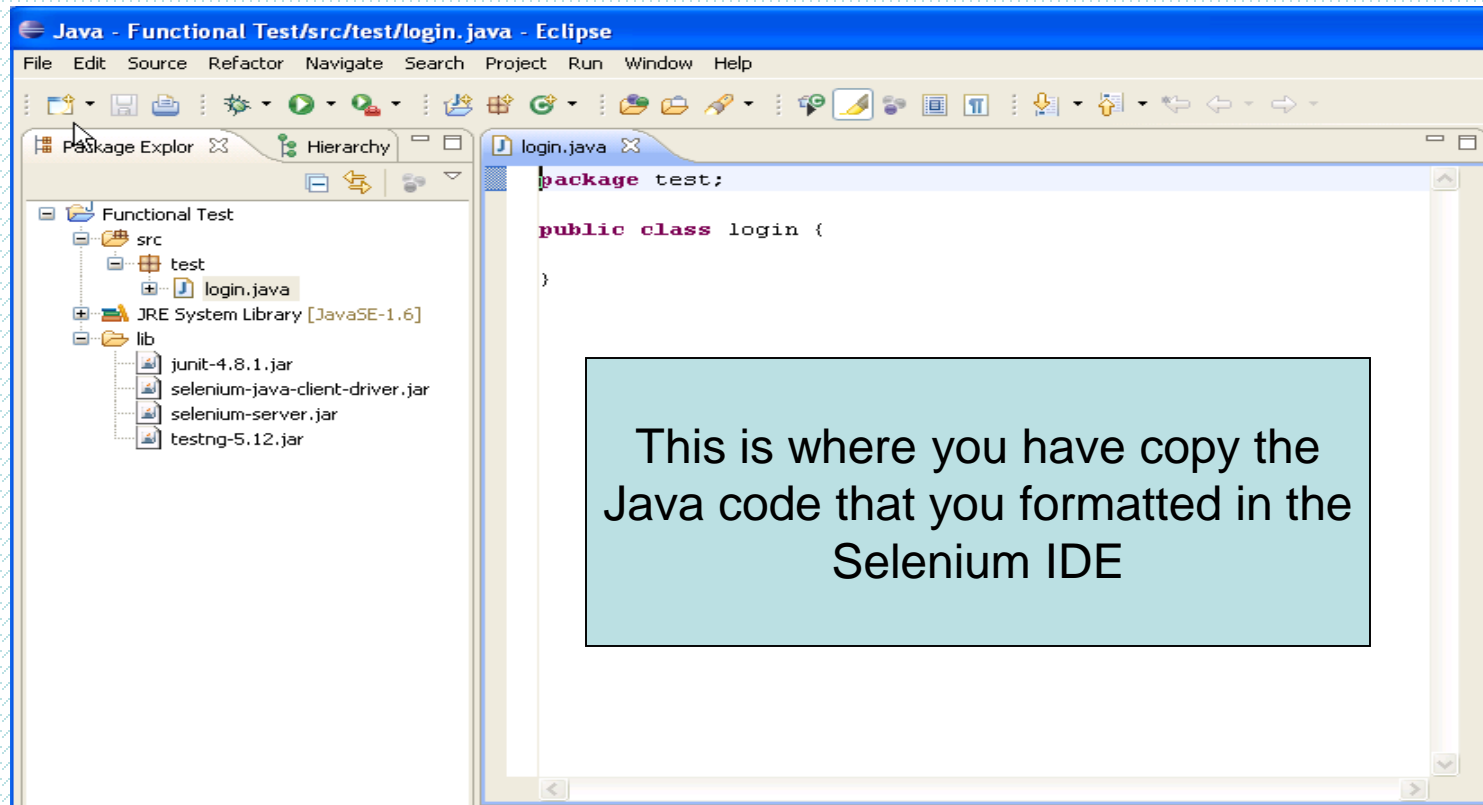
Creating a package and adding a class file

- In the eclipse → Package Explorer → right click on the src (source) folder. In that select New → Package and give a name to the package. See the Picture1
- Now right click on the created package and click on new → class and provide a class name. You will see a java script template as seen in the next slide



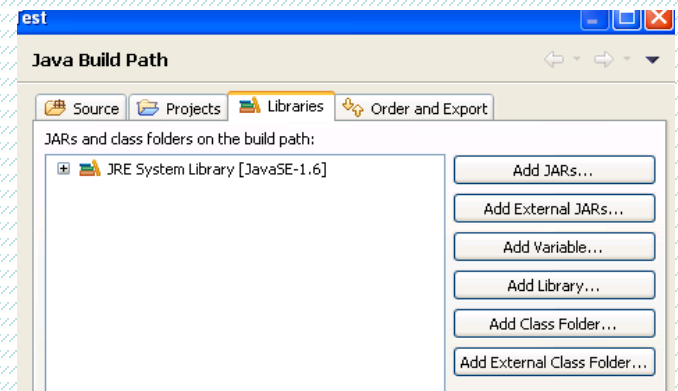
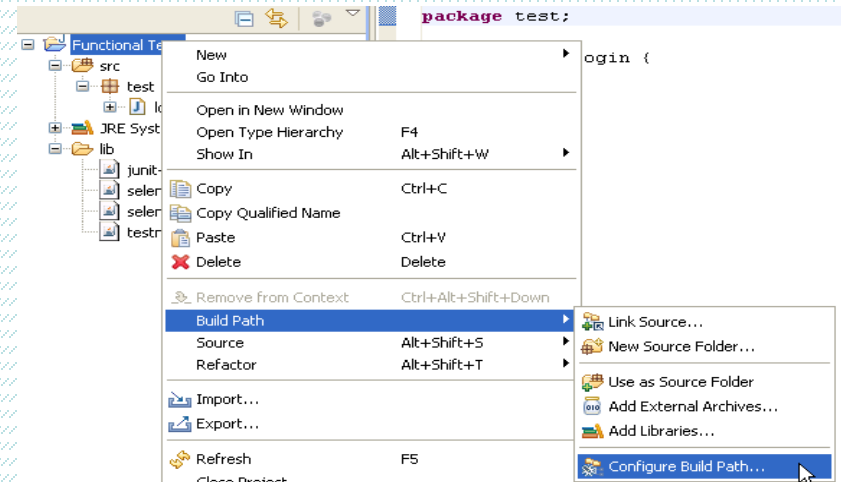
Eclipse IDE with Java Script Template

- Your IDE should look like this



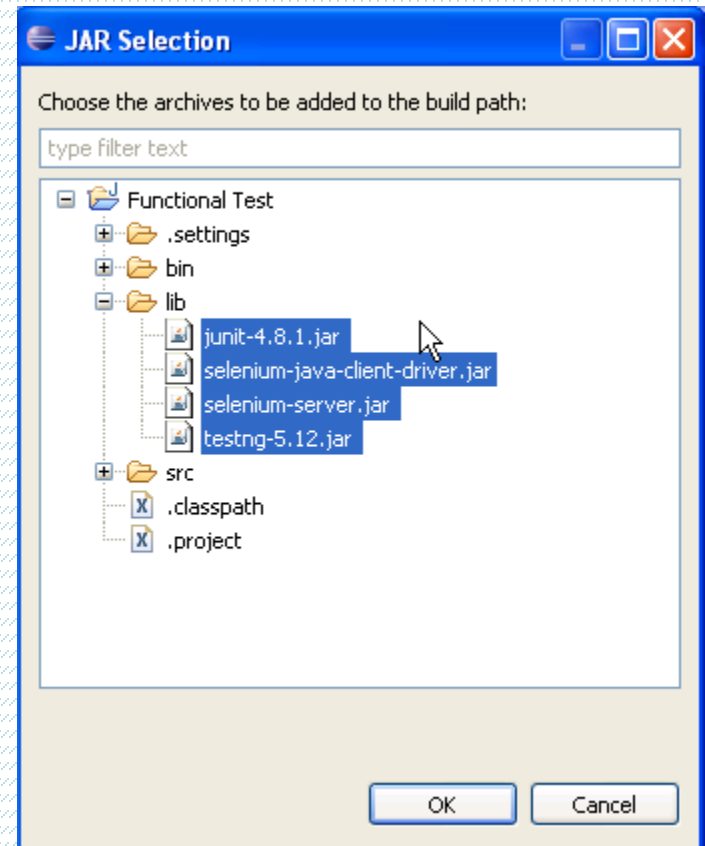
Adding the Jar files to the class path

- Right Click on the Project
- Select Build Path
→ Configure Build path
- On the next window click on the libraries tab and click on the add jars tab



Adding the Jar files to the class path

- As you click on the add jars button a window showing all the jar files will be opened. Select all the jar files and click on ok button
- By doing this step we are almost ready to write the script and run it.

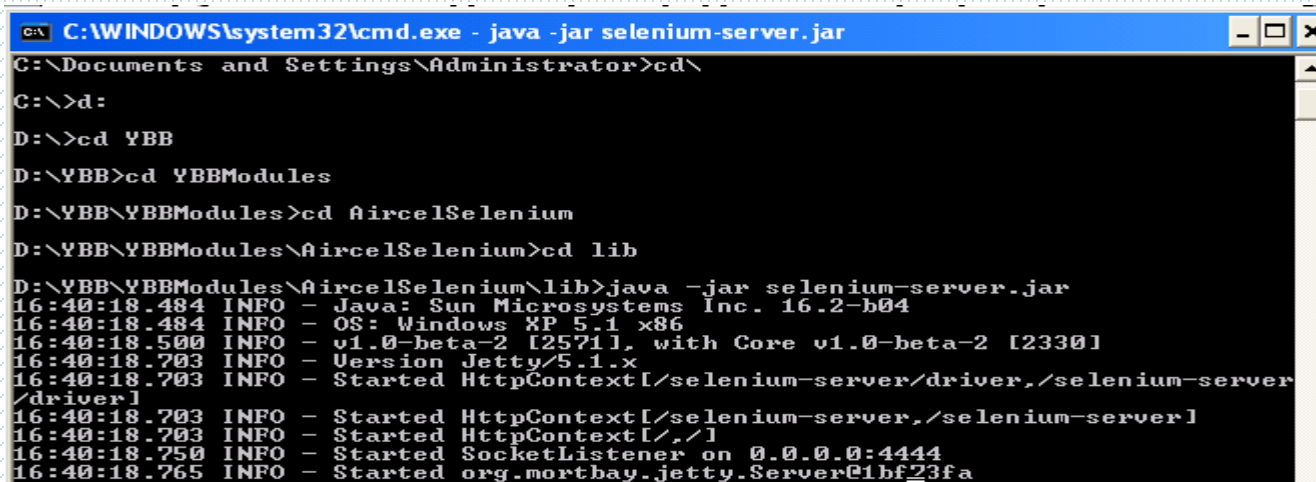


Creating the test script inside the class file

- Copy the code that was formatted in the selenium IDE and paste it in the Eclipse IDE script template.
- Make sure the class name that you created and the class name in the script are same
- Make sure that you have add the methods setUp and tearDown

Running the test through Eclipse

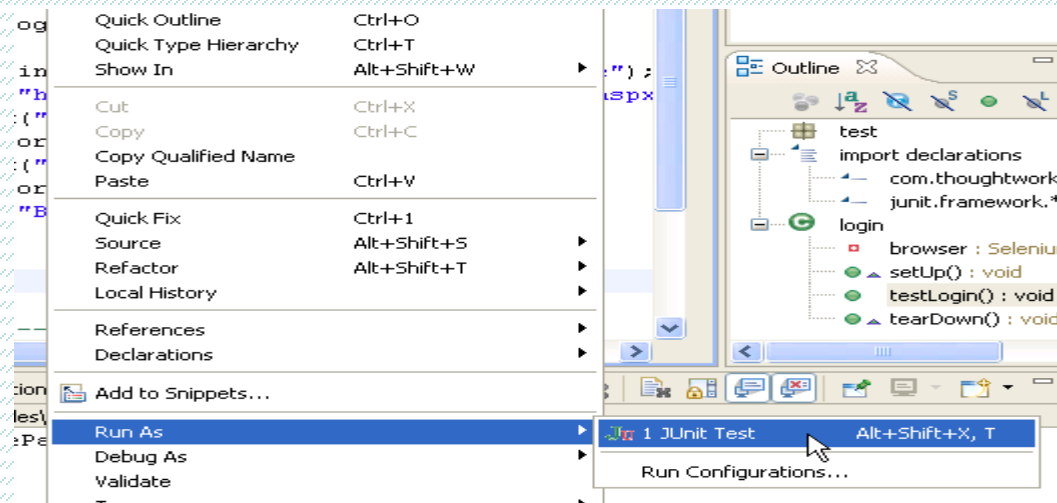
- Start the Selenium Server
- For this open the command prompt and go to the folder where the selenium server is present.
- Run the command “java -jar selenium-server.jar”
- The Selenium server will start running and you can see the message in the command prompt.



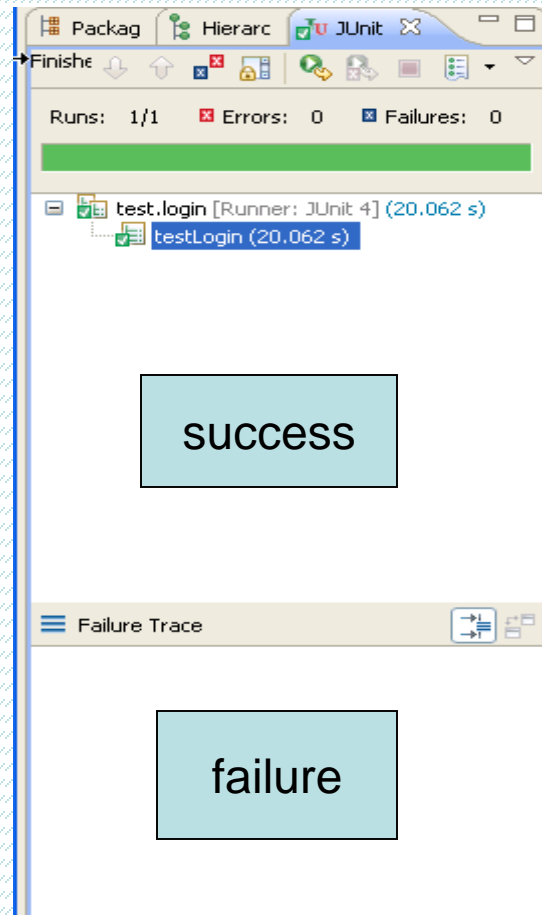
```
C:\WINDOWS\system32\cmd.exe - java -jar selenium-server.jar
C:\Documents and Settings\Administrator>cd\
C::\>d:
D::\>cd YBB
D:\YBB>cd YBBModules
D:\YBB\YBBModules>cd AircelSelenium
D:\YBB\YBBModules\AircelSelenium>cd lib
D:\YBB\YBBModules\AircelSelenium\lib>java -jar selenium-server.jar
16:40:18.484 INFO - Java: Sun Microsystems Inc. 16.2-b04
16:40:18.484 INFO - OS: Windows XP 5.1 x86
16:40:18.500 INFO - v1.0-beta-2 [25711, with Core v1.0-beta-2 [23301
16:40:18.703 INFO - Version Jetty/5.1.x
16:40:18.703 INFO - Started HttpContext[/selenium-server/driver,/selenium-server
/driver/
16:40:18.703 INFO - Started HttpContext[/selenium-server,/selenium-server/]
16:40:18.703 INFO - Started HttpContext[/,/]
16:40:18.750 INFO - Started SocketListener on 0.0.0.0:4444
16:40:18.765 INFO - Started org.mortbay.jetty.Server@1bf23fa
```

Running the test through Eclipse

- Once the server is up go to the eclipse and right click on the script.
- In that select Run As → JUnit Test. See the image below. After this you will get will see that the script has run successfully in the window that's shown in the next slide.



Selenium Results Strip in the Eclipse IDE



Enhancing the Selenium commands

The generated selenium command can be enhanced by writing java commands. Using java methods we can do parameterization and data base validation. More over java methods can be used for to perform some complex validations and testing activities.

The best way to use java methods is used to is have a Java IDE like eclipse.

Some Java Basics

Java Data Types

Int, float, String, char, Boolean and double

Declarations

```
String s = "Selenium";  
Int i = 20;  
float cur = 3.35;
```

Operators

* , / , % , + , - are the mathematical operators
* , / , % , have a higher precedence than + or -

Relational Operators

==	Equal (careful)
!=	Not equal
>=	Greater than or equal
<=	Less than or equal
>	Greater than
<	Less than

Programming Elements

If Condition

```
if (name != "selenium")
{
    System.out.print("Tool Changed");
}
Else
{
    System.out.print("Tool is ok");
}
```

Loop n times

```
for ( i = 0; i < n; n++ )
{
    // this code body will execute n times
    // I from 0 to n-1
}
```

Use string functions

```
String s = browser.getBodyText();  
int pos = s.indexOf("books");  
System.out.print(pos);
```

```
String s = browser.getBodyText();  
int len = s.length();  
System.out.print(len);
```

```
String s = browser.getBodyText();  
boolean flg = s.endsWith("pass");  
System.out.print(flg);
```

Date Functions

Required Packages

- **import** java.util.Date;
- **import** java.text.SimpleDateFormat;

Sample Code

```
public void sampled()  
{  
    Date currentDatetime = new Date(System.currentTimeMillis());  
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm");  
    String myDate = formatter.format(currentDatetime);  
    System.out.print(myDate);  
}
```

Example 1 – Get the values from the list box and check if the value “HTML” is present in it

Implementation Steps

- 1) First get the values from the list box
- 2) To get the values from the list box or any other HTML element we need to know its Xpath. (use Xpath checker)
- 3) The values taken from the list box are stored in a variable.
- 4) Use the indexOf method to find if “HTML” is present in the extracted variable.

Screen Shot of the Example Script

```
1
System.out.println ("Running the Test testHomePage");
browser.open("http://localhost/bookstore/Default.aspx");
String listContent= browser.getText("//table[@id='Search_holder']/tbody/tr[2]/td[2]");
int var = listContent.indexOf("HTML");
if(var>0)
    System.out.println("Pass");
else
    System.out.println("Pass");
browser.click("Search_search_button");
browser.waitForPageToLoad("30000");
browser.click("//a[@id='Header_Menu_Home']/img");
browser.waitForPageToLoad("30000");
assertEquals("Book Store",browser.getTitle());
```



Xpath Locator

Selenium-Grid

Selenium-Grid allows the Selenium-RC solution to scale for test suites or test suites to be run in multiple environments.

- With Selenium-Grid multiple instances of Selenium-RC are running on various operating system and browser configurations, each of these when launching register with a hub. When tests are sent to the hub they are then redirected to an available Selenium-RC, which will launch the browser and run the test.
- This allows for running tests in parallel, with the entire test suite theoretically taking only as long to run as the longest individual test.

Best practices

- Use Selenium to verify workflow and session
- Don't put Selenium tests in your main development build – run them overnight
- Have dedicated machines that run tests
- **DON'T THINK OF THIS AS A REPLACEMENT FOR EXPLORATORY TESTING!!!**