**Selenium Remote Control (RC)**

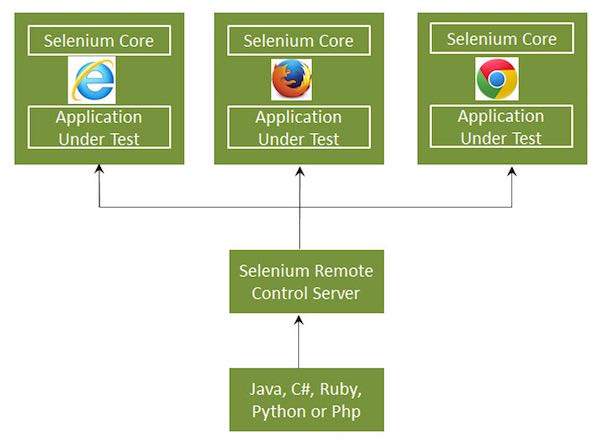
Selenium Remote Control (RC) was the main Selenium project that sustained for a long time before Selenium WebDriver (Selenium 2.0) came into existence. Now Selenium RC is hardly in use, as WebDriver offers more powerful features, however users can still continue to develop scripts using RC.

It allows us to write automated web application UI tests with the help of full power of programming languages such as Java, C#, Perl, Python and PHP to create more complex tests such as reading and writing files, querying a database, and emailing test results.

**Selenium RC Architecture**

Selenium RC works in such a way that the client libraries can communicate with the Selenium RC Server passing each Selenium command for execution. Then the server passes the Selenium command to the browser using Selenium-Core JavaScript commands.

The browser executes the Selenium command using its JavaScript interpreter.



**Selenium RC comes in two parts.**

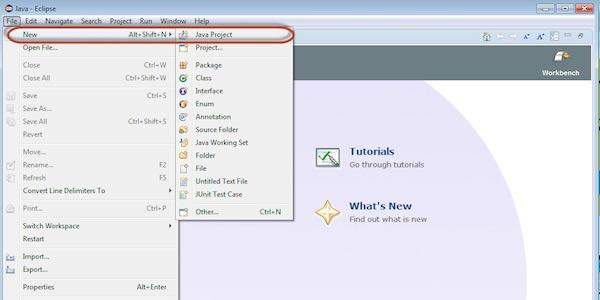
* The Selenium Server launches and kills browsers. In addition to that, it interprets and executes the Selenese commands. It also acts as an HTTP proxy by intercepting and verifying HTTP messages passed between the browser and the application under test.
* Client libraries that provide an interface between each one of the programming languages (Java, C#, Perl, Python and PHP) and the Selenium-RC Server.

**RC Scripting**

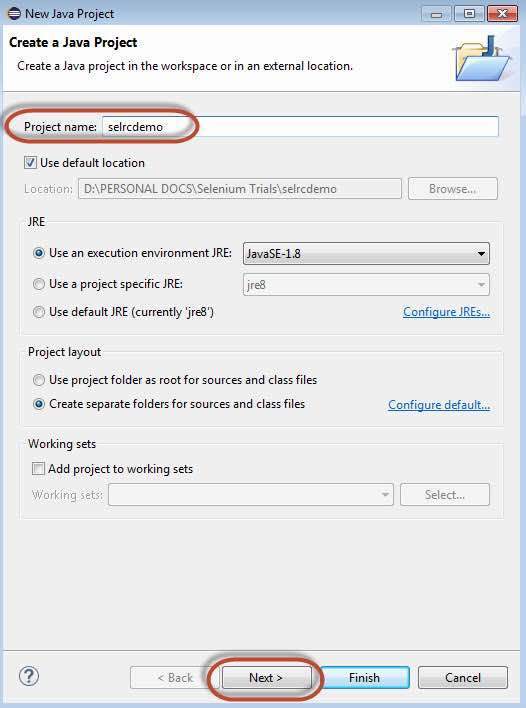
Now let us write a sample script using Selenium Remote Control. Let us use http://www.calculator.net/ for understanding Selenium RC. We will perform a Percent calculation using 'Percent Calculator' that is present under the 'Math Calculators' module.

**Step 1**: Start Selenium Remote Control (with the help of command prompt).

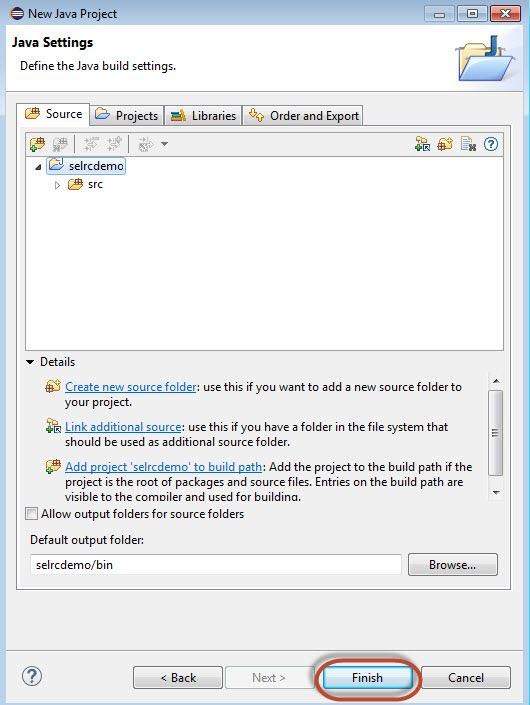
**Step 2**: After launching Selenium RC, open Eclipse and create a "New Project" as shown below.



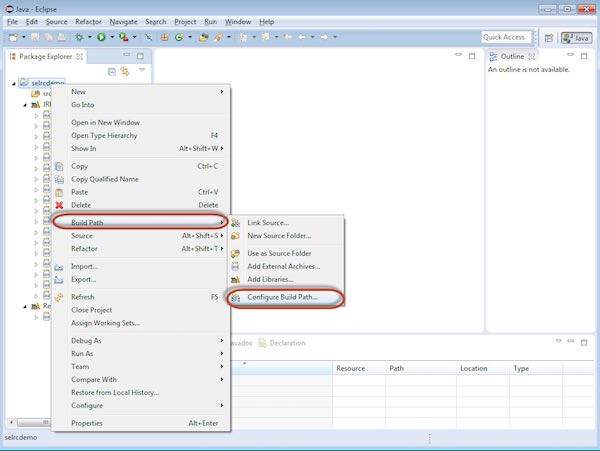
**Step 3**: Enter the project name and click 'Next' button.



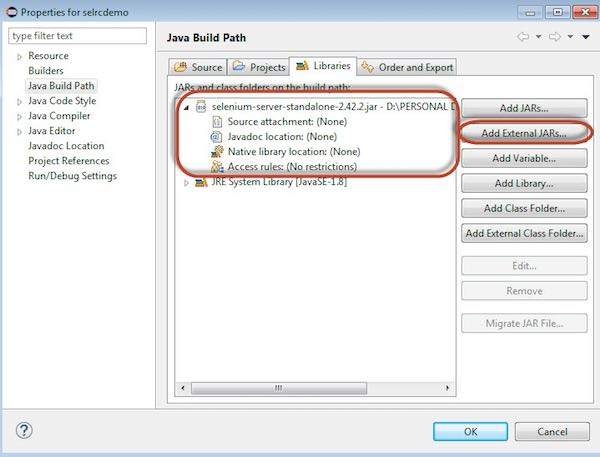
**Step 4**: Verify the Source, Projects, Libraries, and Output folder and click 'Finish'.



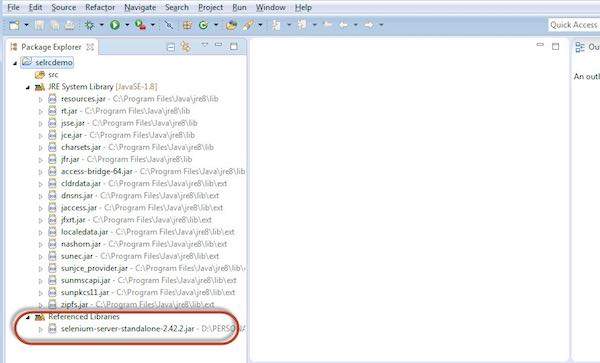
**Step 5**: Right click on 'project' container and choose 'Configure Build Path'.



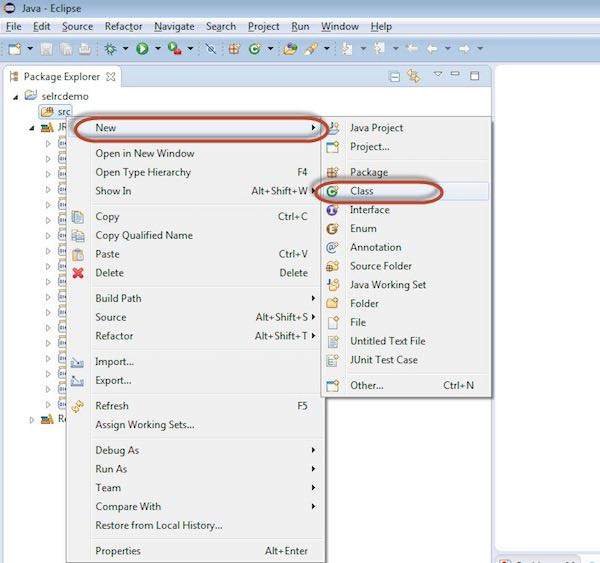
**Step 6**: Properties for 'selrcdemo' opens up. Navigate to 'Libraries' tab and select 'Add External JARs'. Choose the Selenium RC jar file that we have downloaded and it would appear as shown below.



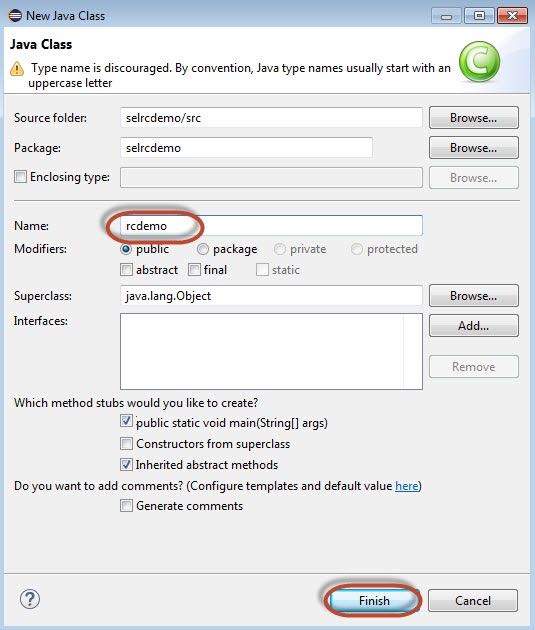
**Step 7** : The referenced Libraries are shown as displayed below.



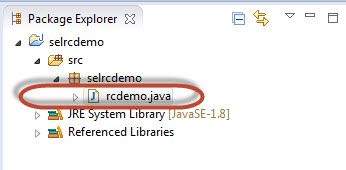
**Step 8** : Create a new class file by performing a right click on 'src' folder and select 'New' >> 'class'.



**Step 9**: Enter a name of the class file and enable 'public static void main' as shown below.



**Step 10**: The Created Class is created under the folder structure as shown below.



**Step 11**: Now it is time for coding. The following code has comments embedded in it to make the readers understand what has been put forth.

package selrcdemo;

import com.thoughtworks.selenium.DefaultSelenium;

import com.thoughtworks.selenium.Selenium;

public class rcdemo {

public static void main(String[] args) throws InterruptedException {

// Instatiate the RC Server

Selenium selenium = new DefaultSelenium("localhost", 4444 , "Firefox", "http://www.calculator.net");

selenium.start(); // Start

selenium.open("/"); // Open the URL

selenium.windowMaximize();

// Click on Link Math Calculator

selenium.click("xpath=.//\*[@id='menu']/div[3]/a");

Thread.sleep(2500); // Wait for page load

// Click on Link Percent Calculator

selenium.click("xpath=.//\*[@id='menu']/div[4]/div[3]/a");

Thread.sleep(4000); // Wait for page load

// Focus on text Box

selenium.focus("name=cpar1");

// enter a value in Text box 1

selenium.type("css=input[name=\"cpar1\"]", "10");

// enter a value in Text box 2

selenium.focus("name=cpar2");

selenium.type("css=input[name=\"cpar2\"]", "50");

// Click Calculate button

selenium.click("xpath=.//\*[@id='content']/table/tbody/tr/td[2]/input");

// verify if the result is 5

String result = selenium.getText (".//\*[@id='content']/p[2]");

if (result == "5"){

System.out.println ("Pass");

}

else{

System.out.println ("Fail");

}

}

}

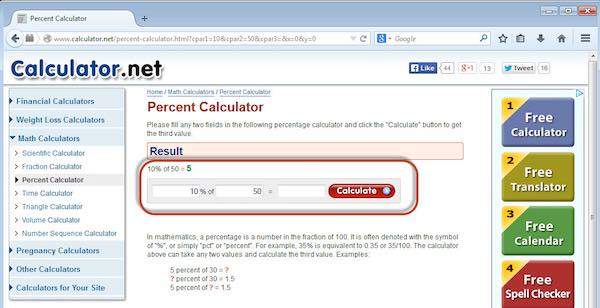
**Step 11**: Now, let us execute the script by clicking 'Run' Button.



**Step 12**: The script would start executing and user would be able to see the command history under 'Command History' Tab.



**Step 13**: The final state of the application is shown as below. The percentage is calculated and it displayed the result on screen as shown below.



**Step 14**: The output of the test is printed on the Eclipse console as shown below as we have printed the output to the console. In real time the output is written to an HTML file or in a simple Text file.

