**Selenium Tutorial – Selenium WebDriver**

Before I get started with Selenium WebDriver, let me show you the topics I will be covering in this blog which will help you write your first Selenium code for automation testing. In this blog, I have written Selenium code to test automated login to Facebook. The topics are:

* [Drawbacks of Selenium RC and the birth of WebDriver](https://www.edureka.co/blog/selenium-tutorial#DrawbacksOfSeleniumRC)
* [What is Selenium WebDriver?](https://www.edureka.co/blog/selenium-tutorial#WhatIsSeleniumWebDriver)
* [What are browser elements?](https://www.edureka.co/blog/selenium-tutorial#WhatAreBrowserElements)
* [Locating browser elements on a web page](https://www.edureka.co/blog/selenium-tutorial#LocatingBrowserElements)
* [Operations on browser elements](https://www.edureka.co/blog/selenium-tutorial#OperationsOnBrowserElements)

**Drawbacks Of Selenium RC And The Birth Of WebDriver**

Let’s first discuss the limitations of Selenium RC because that was the reason for the eventual development of WebDriver. You might be surprised when I say that Selenium RC became an instant hit when it was launched. That was because it overcame the **same origin policy** issue which was a major problem while testing web apps with Selenium Core. But do you know what the same origin policy issue was?

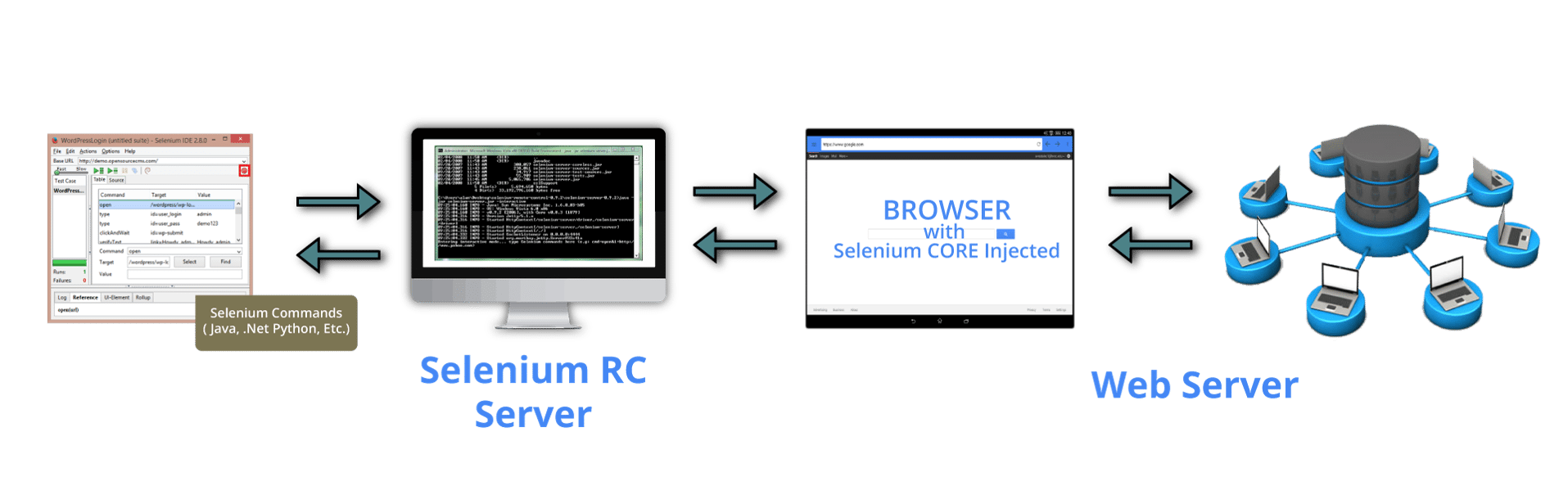
Same origin policy are rules which enforce web application security model. According to the same origin policy, the web browser will allow JavaScript codes to access elements on the web page, if and only if both the JavaScript and web page being tested are hosted from the same domain. Selenium Core being a JavaScript-based testing tool, was handicapped for the same reason that it could not test every web page.

But when Selenium RC came into the picture, it rid testers of the same origin policy issue. But, how did RC do that? RC did that by using another component called **Selenium RC server**. So, RC is a tool which is a combination of two components: **Selenium RC server** and **Selenium RC client**.

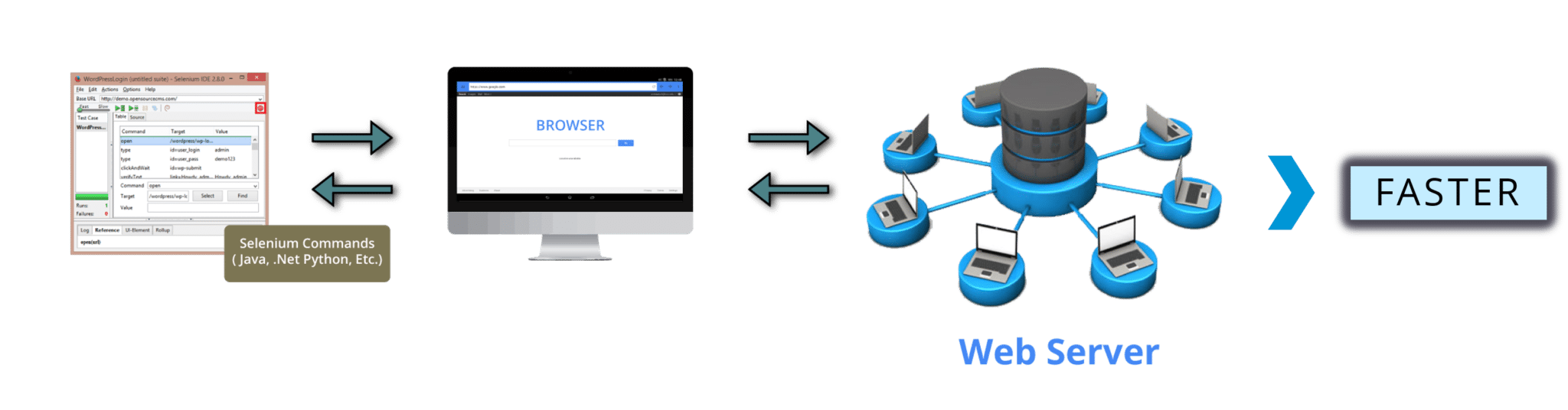
Selenium RC server is an HTTP proxy server, designed to “trick” the browser into believing that Selenium Core and the web application being tested are from the same domain. Hence, there is no stopping the JavaScript code from accessing and testing any web site.

Even though Selenium RC was a major hit, it had its own share of problems. The major one being time taken for executing tests. Since the Selenium RC server is the middle man in the communication between the browser and your Selenium commands, test executions are very time consuming. Besides the time factor, RC’s architecture is also slightly complicated.

This architecture involves first injecting Selenium Core into the web browser. Then Selenium Core will receive the instructions from RC server and convert it into a JavaScript command. This JavaScript code is responsible for accessing and testing the web  elements. If you look at the image below, you will get an idea of how RC works.



To overcome these problems, Selenium WebDriver was developed. WebDriver is faster because it interacts directly with the browser and there is no involvement of an external proxy server. The architecture is also simpler as the browser is controlled from the OS level. The below image will help you understand how WebDriver works.



Another benefit with WebDriver is that it supports testing on the HTML Unit driver which is a headless driver. When we say a headless driver, it refers to the fact that the browser has no GUI. RC on the other hand does not support HTML Unit driver. These are some of the reasons why WebDriver scores over RC.

**What Is Selenium WebDriver?**

In this part of Selenium tutorial blog, I will dig deep into Selenium WebDriver. There is a good chance that you will be aware of the details I have covered in the below paragraph, but I will be revising it anyway.

Selenium WebDriver is a web-based automation testing framework which can test web pages initiated on various web browsers and various operating systems. In fact, you also have the freedom to write test scripts in different programming languages like: Java, Perl, Python, Ruby, C#, PHP and JavaScript. Do note that Mozilla Firefox is Selenium WebDriver’s default browser.

WebDriver was introduced as part of Selenium v2.0. Selenium v1 consisted of only IDE, RC and Grid. But the major breakthrough in the Selenium project was when WebDriver was developed and introduced as a replacement in Selenium v2. However, with the release of Selenium v3, RC has been deprecated and moved to legacy package. You can still download and work with RC but, don’t expect any support for it.

In a nutshell, the advantages WebDriver has over RC are:

* Support for more programming languages, operating system and web browsers
* Overcoming the limitations of Selenium 1 like file upload, download, pop-ups & dialog barrier
* Simpler commands when compared to RC, and a better API
* Support for Batch testing, Cross browser testing & Data driven testing

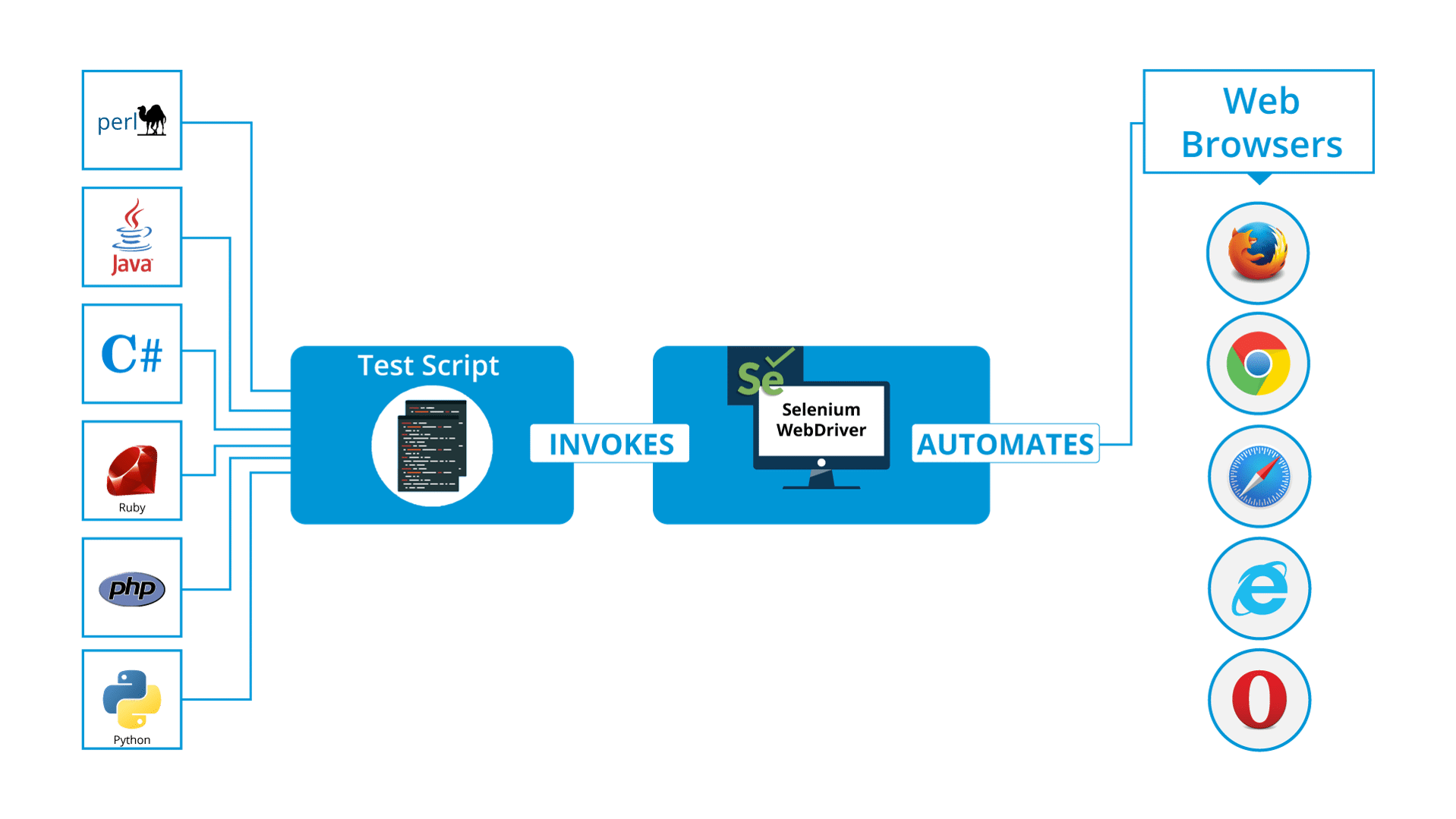
But the drawback when compared to RC is that, the test reports cannot be generated. RC generates detailed reports.

**[Selenium Certification Training](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)*
* *[Assignments](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/testing-with-selenium-webdriver" \t "_blank)

The below image depicts how WebDriver works:



You must have heard the term “browser elements” a number of times. The next part of this Selenium tutorial will be about what are these elements and how testing happens on these web elements.

**What Are Browser Elements?**

Elements are the different components that are present on web pages. The most common elements we notice while browsing are:

* Text boxes
* CTA Buttons
* Images
* Hyperlinks
* Radio buttons/ Check boxes
* Text area/ Error messages
* Drop down box/ List box/ Combo box
* Web Table/ HTML Table
* Frame

Testing these elements essentially means we have to check whether they are working fine and responding the way we want it to. For example, if we are testing text boxes, what would you test it for?

1. Whether we are able to send text or numbers to the text box
2. Can we retrieve text that has been passed to the text box, etc.

If we are testing an image, we might want to:

1. Download the image
2. Upload the image
3. Click on the image link
4. Retrieve the image title, etc.

Similarly, operations can be performed on each of the elements mentioned earlier. But only after the elements are located on the web page, we can perform operations and start testing them right? So, the next topic, I will be covering in this Selenium tutorial blog is element locator techniques.

**Locating Browser Elements Present On The Web Page**

Every element on a web page will have attributes (properties). Elements can have more than one attribute and most of these attributes will be unique for different elements. For example, consider a page having two elements: an image and a text box. Both these elements have a ‘Name’ attribute and an ‘ID’ attribute. These attribute values need to be unique for each element. In other words, two elements cannot have the same attribute value. Elements can have the same value for ‘Class Name’.

In the example considered, the image and text box can neither have the same ‘ID’ value nor the same ‘Name’ value. However, there are some attributes that can be common for a group of elements on the page. I will tell you which are those attributes later, but before that let me list down the 8 attributes using which we can locate elements. Those attributes are ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, CSS and XPath.

Since the elements are located using these attributes, we refer to them as ‘Locators’. The locators are:

* **By.id**  
  Syntax: driver.findElement(By.id(“xxx”));
* **By.name**  
  Syntax: driver.findElement(By.name(“xxx”));
* **By.className**  
  Syntax: driver.findElement(By.className(“xxx”));
* **By.tagName**  
  Syntax: driver.findElement(By.tagName(“xxx”));
* **By.linkText**  
  Syntax: driver.findElement(By.linkText(“xxx”));
* **By.partialLinkText**  
  Syntax: driver.findElement(By.partialLinkText(“xxx”));
* **By.css**  
  Syntax: driver.findElement(By.css(“xxx”));
* **By.xpath**  
  Syntax: driver.findElement(By.xpath(“xxx”));

By looking at the syntax above, you might have realized locators are called inside methods. So, before going any further, you need to learn all the other methods, browser commands and functions that can be used to perform operations on the elements.

**Operations On Browser Elements**

From this section of the blog onwards, you will be having a lot of fun because there will be less theory and more codes. So be prepared, and keep your Eclipse IDE open with the required Selenium packages installed.

To start testing a web page, we need to first open a browser, then navigate to the web page by providing the URL right? Check out the below piece of code, where I have replicated the same. Firefox browser will first be initiated and then it will navigate to Facebook’s login page.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package seleniumWebDriver;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;    public class WebDriverClass  {  public static void main(String[] args)  {  System.setProperty("webdriver.gecko.driver", "files/geckodriver.exe");  WebDriver driver = new FirefoxDriver();  driver.get("<https://www.facebook.com/>");  driver.getTitle();  driver.quit();  }  } |

**import org.openqa.selenium.WebDriver;** is a library package which contains the required class to initiate the browser loaded with a specific driver.

**import org.openqa.selenium.firefox.FirefoxDriver;** is a library package which contains the FirefoxDriver class needed to start FirefoxDriver as the browser initiated by the WebDriver class.

**System.setProperty(“webdriver.gecko.driver”, “files/geckodriver.exe”); –**This command notifies the run-time engine that the Gecko driver is present in the specified path. After Firefox 35, we need to download the Gecko driver for working with WebDriver. In case you want to test on chrome, then you have to download ChromeDriver, which is a .exe file and specify it’s path in this line of code. We have to do it similarly in case of other browsers also.

**WebDriver driver = new FirefoxDriver(); –**This command is used to initiate a new Firefox driver object.

**driver.get(“https://www.gmail.com/”); –**This method is used to open the specified URL.

**driver.getTitle();** – This command gets the title of the tab that is currently open in the browser.

**Software Testing Training**

**[SELENIUM CERTIFICATION TRAINING](https://www.edureka.co/testing-with-selenium-webdriver" \o "Selenium Certification Training " \t "_blank)**

**[Selenium Certification Training](https://www.edureka.co/testing-with-selenium-webdriver" \o "Selenium Certification Training " \t "_blank)**

*[Reviews](https://www.edureka.co/testing-with-selenium-webdriver" \o "Selenium Certification Training " \t "_blank)*

**[5](https://www.edureka.co/testing-with-selenium-webdriver" \o "Selenium Certification Training " \t "_blank)**[(20100)](https://www.edureka.co/testing-with-selenium-webdriver" \o "Selenium Certification Training " \t "_blank)

**[MANUAL TESTING CERTIFICATION TRAINING](https://www.edureka.co/software-testing" \o "Manual Testing Certification Training" \t "_blank)**

**[Manual Testing Certification Training](https://www.edureka.co/software-testing" \o "Manual Testing Certification Training" \t "_blank)**

*[Reviews](https://www.edureka.co/software-testing" \o "Manual Testing Certification Training" \t "_blank)*

**[5](https://www.edureka.co/software-testing" \o "Manual Testing Certification Training" \t "_blank)**[(3707)](https://www.edureka.co/software-testing" \o "Manual Testing Certification Training" \t "_blank)

**[MOBILE APP TESTING USING APPIUM](https://www.edureka.co/appium-training-mobile-automation-testing" \o "Mobile App Testing Using Appium" \t "_blank)**

**[Mobile App Testing Using Appium](https://www.edureka.co/appium-training-mobile-automation-testing" \o "Mobile App Testing Using Appium" \t "_blank)**

*[Reviews](https://www.edureka.co/appium-training-mobile-automation-testing" \o "Mobile App Testing Using Appium" \t "_blank)*

**[5](https://www.edureka.co/appium-training-mobile-automation-testing" \o "Mobile App Testing Using Appium" \t "_blank)**[(1315)](https://www.edureka.co/appium-training-mobile-automation-testing" \o "Mobile App Testing Using Appium" \t "_blank)

**[PERFORMANCE TESTING USING JMETER](https://www.edureka.co/jmeter-training-performance-testing" \o "Performance Testing Using JMeter" \t "_blank)**

**[Performance Testing Using JMeter](https://www.edureka.co/jmeter-training-performance-testing" \o "Performance Testing Using JMeter" \t "_blank)**

*[Reviews](https://www.edureka.co/jmeter-training-performance-testing" \o "Performance Testing Using JMeter" \t "_blank)*

**[5](https://www.edureka.co/jmeter-training-performance-testing" \o "Performance Testing Using JMeter" \t "_blank)**[(1313)](https://www.edureka.co/jmeter-training-performance-testing" \o "Performance Testing Using JMeter" \t "_blank)

**[CONTINUOUS TESTING IN DEVOPS](https://www.edureka.co/continuous-testing-devops-training" \o "Continuous Testing in DevOps" \t "_blank)**

**[Continuous Testing in DevOps](https://www.edureka.co/continuous-testing-devops-training" \o "Continuous Testing in DevOps" \t "_blank)**

*[Reviews](https://www.edureka.co/continuous-testing-devops-training" \o "Continuous Testing in DevOps" \t "_blank)*

**[5](https://www.edureka.co/continuous-testing-devops-training" \o "Continuous Testing in DevOps" \t "_blank)**[(1150)](https://www.edureka.co/continuous-testing-devops-training" \o "Continuous Testing in DevOps" \t "_blank)

**[SOFTWARE TESTING FUNDAMENTALS COURSE](https://www.edureka.co/software-testing-fundamentals-training" \o "Software Testing Fundamentals Course" \t "_blank)**

**[Software Testing Fundamentals Course](https://www.edureka.co/software-testing-fundamentals-training" \o "Software Testing Fundamentals Course" \t "_blank)**

*[Reviews](https://www.edureka.co/software-testing-fundamentals-training" \o "Software Testing Fundamentals Course" \t "_blank)*

**[5](https://www.edureka.co/software-testing-fundamentals-training" \o "Software Testing Fundamentals Course" \t "_blank)**[(214)](https://www.edureka.co/software-testing-fundamentals-training" \o "Software Testing Fundamentals Course" \t "_blank)

**[AUTOMATION TESTING USING TESTCOMPLETE 11.0](https://www.edureka.co/automation-testing-testcomplete" \o "Automation Testing using TestComplete 11.0" \t "_blank)**

**[Automation Testing using TestComplete 11.0](https://www.edureka.co/automation-testing-testcomplete" \o "Automation Testing using TestComplete 11.0" \t "_blank)**

*[Reviews](https://www.edureka.co/automation-testing-testcomplete" \o "Automation Testing using TestComplete 11.0" \t "_blank)*

**[5](https://www.edureka.co/automation-testing-testcomplete" \o "Automation Testing using TestComplete 11.0" \t "_blank)**[(1431)](https://www.edureka.co/automation-testing-testcomplete" \o "Automation Testing using TestComplete 11.0" \t "_blank)

**[RUBY WITH CUCUMBER CERTIFICATION TRAINING](https://www.edureka.co/ruby-with-cucumber-sp" \o "Ruby with Cucumber Certification Training" \t "_blank)**

**[Ruby with Cucumber Certification Training](https://www.edureka.co/ruby-with-cucumber-sp" \o "Ruby with Cucumber Certification Training" \t "_blank)**

*[Reviews](https://www.edureka.co/ruby-with-cucumber-sp" \o "Ruby with Cucumber Certification Training" \t "_blank)*

**[5](https://www.edureka.co/ruby-with-cucumber-sp" \o "Ruby with Cucumber Certification Training" \t "_blank)**[(922)](https://www.edureka.co/ruby-with-cucumber-sp" \o "Ruby with Cucumber Certification Training" \t "_blank)

Next

**driver.quit(); –**This command closes the browser driver.

But, what if you want to navigate to a different URL and then do testing? In that case you can use the navigate.to() command as shown in the below code snippet. If you then want to come back to the previous page, then you can do that by using navigate.back() command. Similarly for refreshing the current page, you can use navigate.refresh() command.

|  |  |
| --- | --- |
| 1  2  3 | driver.navigate().to(“https://www.gmail.co/testing-with-selenium-webdriver”);  driver.navigate().refresh();  driver.navigate().back(); |

If you want to maximize the size of browser window, then you can do that by using the code in the snippet below.

|  |  |
| --- | --- |
| 1 | driver.manage().window().maximize(); |

In case you want to set a custom size for the browser window, then you can set your own dimensions as shown in the below code snippet.

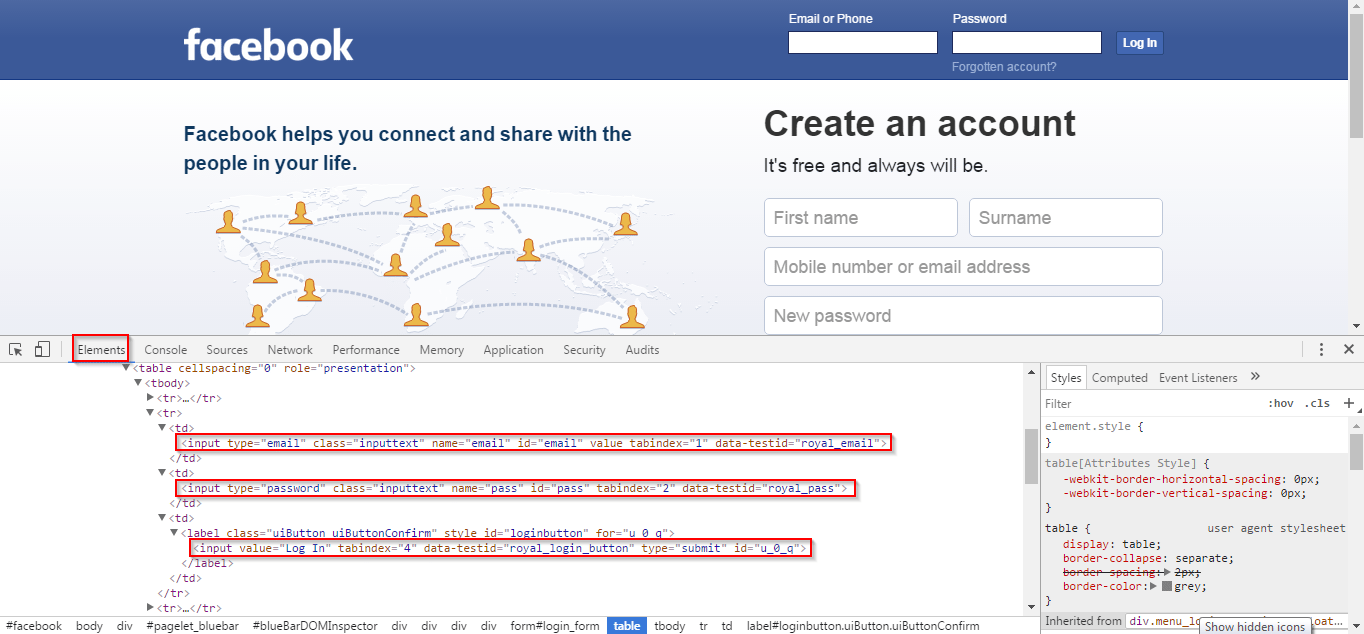
|  |  |
| --- | --- |
| 1  2 | Dimension d = new Dimension(420,600);  driver.manage().window().setSize(d); |

Now that you know most of the basics, let’s go to the next topic in this Selenium tutorial blog. Let’s try to find an element on the web page and then perform any operation that is possible.

I’m pretty sure, you all have Facebook accounts. So, let me show you how to log into Facebook by passing the credentials from the code itself.

There are two text fields in the Facebook login page, one for **Email/Phone** and another for **Password**. We have to locate these two elements, pass the credentials to those elements and then find the third element: **Login** button which needs to be clicked on.

Look at the screenshot below. It is the screenshot of **Facebook’s**loginpage.



If you Inspect (Ctlr + Shift + i) this page, then you will get the same window in your browser. Then, under **Elements**, list of all the elements present on the page and their attributes will be displayed. There are three portions highlighted in the above screenshot. The first highlighted element is an email text field, the second is the password text field and the third is the Login button.

If you can recall, I mentioned earlier that these elements can be located using element locator techniques. Let’s use it to locate these elements and send the field values.  
This is the syntax for finding the element: **driver.findElement(By.id(“xxx”));**For sending it values, we can use the method **sendKeys(“*credentials*“);**For clicking on a button, we have to use the method **click();**

So, let’s get started with finding the element and performing an operation on it. The code for it is in the below snippet.

|  |  |
| --- | --- |
| 1  2  3 | driver.findElement(By.name("email")).sendKeys("xxx@gmail.com");  driver.findElement(By.name("pass")).sendKeys("xxxxxx");  driver.findElement(By.id("u\_0\_q")).click(); |

In line #1, we are identifying the *Email* element by its unique ‘Name’ attribute and sending it the EmailID.   
In line #2, we are identifying the *Password* element by its unique ‘Name’ attribute and sending it the password.  
In line #3, we are locating the *Login button* element by its unique ID and clicking on that button.

Adding just these lines of code might not be enough. That is because of the dynamics of the page, it might not respond immediately and by the time the page loads, WebDriver will get terminated and throw a timeout exception error. This issue might not happen in Facebook’s page because it is fast, but will most likely happen in any other E-Commerce site and other dynamic web sites.

To overcome this problem, we need to use an advanced technique. We need to request our WebDriver to wait after the page is accessed and after it loads completely, we need to locate the elements and then perform actions.

In case you want your WebDriver to wait until all the elements load in a web page and then close the browser, then we can achieve that by using **driver.wait()** method or **Threads.sleep() method**. However, if you are writing more advanced code, then you should use **Implicit waits** or **Explicit waits**. In the next blog of this Selenium tutorial series, I will explain the concept of wait conditions. But for our case, the below commands are enough.

|  |  |
| --- | --- |
| 1  2  3 | driver.wait(5000);  // or use this:-  Thread.sleep(5000); |

But, while working with wait conditions, remember to import this library:  
**import java.util.concurrent.TimeUnit;**We do it because, the class for wait and its related methods will be present in this library.

The entire code I explained, is present in the below code snippet.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package seleniumWebDriver;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import java.util.concurrent.TimeUnit;    public class WebDriverClass  {  public static void main(String[] args)  {  System.setProperty("webdriver.gecko.driver", "files/geckodriver.exe");  WebDriver driver = new FirefoxDriver();  driver.get("<https://www.facebook.com/>");  driver.manage().window().maximize();  driver.getTitle();  driver.navigate().to(“https://www.gmail.co/testing-with-selenium-webdriver”);    driver.navigate().back();  driver.navigate().refresh();  driver.wait(5000);  // or use  // Thread.sleep(5000);    driver.findElement(By.name("email")).sendKeys("xxx@gmail.com");  driver.findElement(By.name("pass")).sendKeys("xxxxxx");  driver.findElement(By.id("u\_0\_q")).click();    driver.quit();  }  } |

When you replace the credentials with your actual email and password and execute this code, then Facebook will open in a new window, enter your credentials and login to your account.

**Voila**! You have successfully logged in, which means your complete code executed completely.

I have used the **ID** and **Name**attributes for locating elements. You can in fact use any other locator for finding the elements. **XPath** is the most useful and important of the locator techniques. But, as long as you can find even one of the attributes and use them for locating elements, you should be good. You can see the below video delivered by an industry expert, where she has shown all the above mentioned features of Selenium WebDriver with hands-on.

This Selenium WebDriver tutorial video talks about the drawbacks of Selenium RC and what was the need for Selenium WebDriver. It goes into the details of the advantages that WebDriver has over RC and how it replaced RC for automation testing.

My next blog in this Selenium tutorial series is about how /blog/selenium-tutorial can be used along with Selenium WebDriver. I urge you to read it because it talks about how the limitations of WebDriver (test case management and report generation) can be overcome by using TestNG.