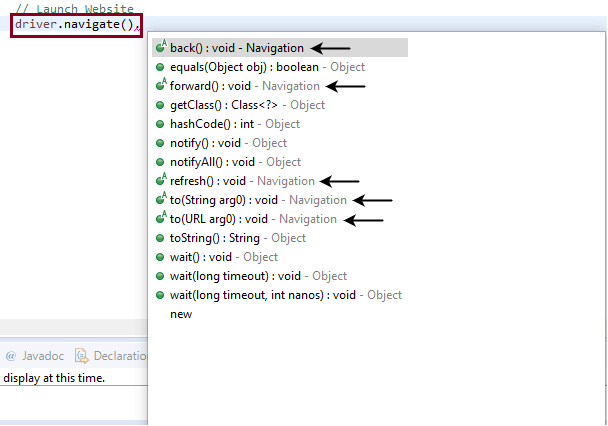
Selenium WebDriver - Navigation Commands

WebDriver provides some basic Browser Navigation Commands that allows the browser to move backwards or forwards in the browser's history.

Just like the browser methods provided by WebDriver, we can also access the navigation methods provided by WebDriver by typing driver.navigate() in the Eclipse panel.



Note: The methods having 'Navigation' as keyword are declared as Navigation commands.

Given are some of the most commonly used Browser Navigation commands for Selenium WebDriver.

1. Navigate To Command

**Method:**

1. to(String arg0) : **void**

In WebDriver, this method loads a new web page in the existing browser window. It accepts *String* as parameter and returns *void*.

The respective command to load/navigate a new web page can be written as:

1. driver.navigate().to("www.javatpoint.com");

Note: The get command (driver.get(URL);) which lies in the browser commands section does the same function as the navigate command

1. (driver.navigate().to("www.javatpoint.com");

2. Forward Command

**Method:**

1. to(String arg0) : **void**

In WebDriver, this method enables the web browser to click on the **forward** button in the existing browser window. It neither accepts anything nor returns anything.

The respective command that takes you forward by one page on the browser's history can be written as:

1. driver.navigate().forward();

3. Back Command

**Method:**

1. back() : **void**

In WebDriver, this method enables the web browser to click on the **back** button in the existing browser window. It neither accepts anything nor returns anything.

The respective command that takes you back by one page on the browser's history can be written as:

1. driver.navigate().back();

4. Refresh Command

**Method:**

1. refresh() : **void**

In WebDriver, this method refresh/reloads the current web page in the existing browser window. It neither accepts anything nor returns anything.

The respective command that takes you back by one page on the browser's history can be written as:

1. driver.navigate().refresh();

Let us consider a sample test script which will cover most of the Navigation Commands provided by WebDriver.

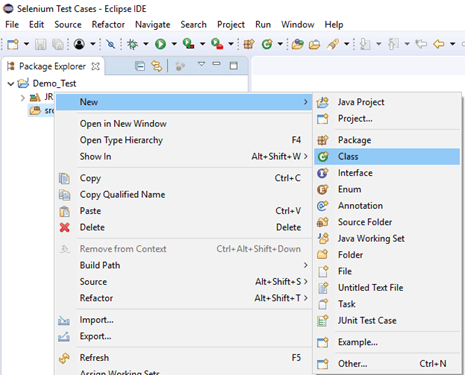
**In this sample test, we will automate the following test scenarios:**

* Invoke Firefox Browser
* Navigate to URL: <https://www.testandquiz.com/selenium/testing.html>
* Click on the "This is a link" link (This link will redirect you to the javaTpoint website)
* Come back to the Home page using the **back** command
* Again go back to the javaTpoint website using **forward** command
* Again come back to the Home page using **To** command
* Refresh the Browser using **Refresh** command
* Close the Browser

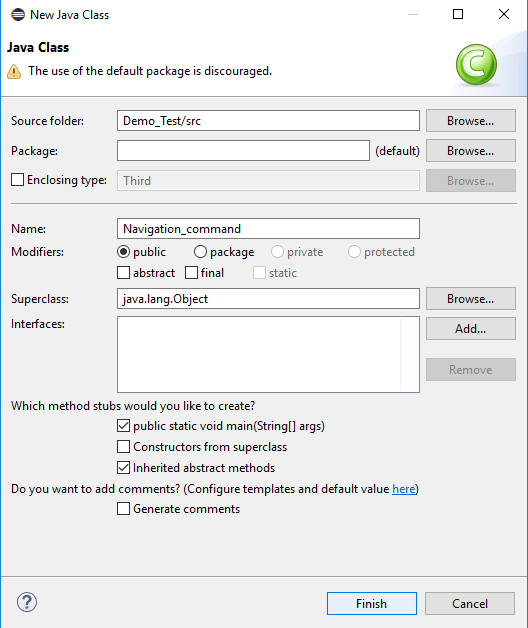
For our test purpose, we are using a dummy web page under the URL:

<https://www.testandquiz.com/selenium/testing.html> (You can also use this dummy web page for your Selenium Test Practices)

* **Step1.** Launch Eclipse IDE and open the existing test suite "Demo\_Test" which we have created in [WebDriver Installation](https://www.javatpoint.com/selenium-webdriver-installation" \t "_blank) section of WebDriver tutorial.
* **Step2.** Right click on the "src" folder and create a new Class File from **New >Class**.



Give your Class name as "Navigation\_command" and click on "Finish" button.



**Step3.** Let's get to the coding ground.

* To invoke Firefox browser, we need to download Gecko driver and set the system property for Gecko driver.

Here is the sample code to set system property for Gecko driver:

1. // System Property for Gecko Driver System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" )

After that we have to initialize Gecko Driver using Desired Capabilities Class.

Here is the sample code to initialize gecko driver using DesiredCapabilities class.

1. // Initialize Gecko Driver using Desired Capabilities Class
2. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
3. capabilities.setCapability("marionette",**true**);
4. WebDriver driver= **new** FirefoxDriver(capabilities);

Combining both of the above code blocks, we will get the code snippet to launch Firefox browser.

1. // System Property for Gecko Driver
2. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
4. // Initialize Gecko Driver using Desired Capabilities Class
5. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
6. capabilities.setCapability("marionette",**true**);
7. WebDriver driver= **new** FirefoxDriver(capabilities);

* After that we need to write the code which will automate our second test scenario (get the desired URL)

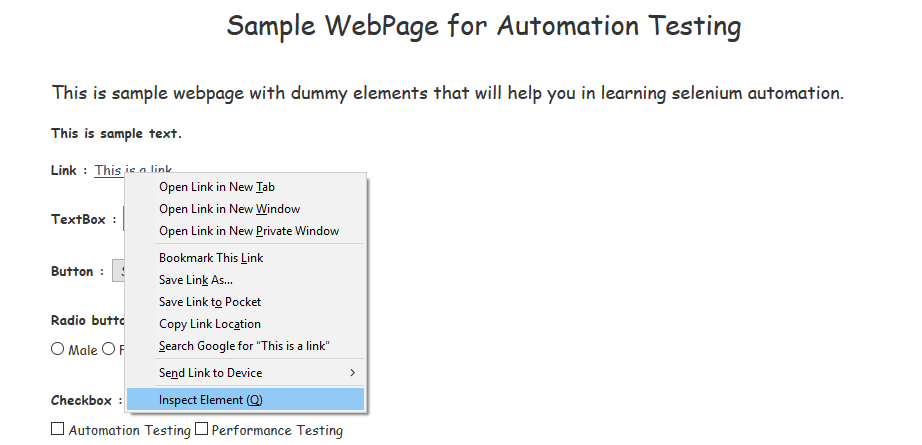
Here is the sample code to navigate to the desired URL:

1. //Navigate to the desired URL
2. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");

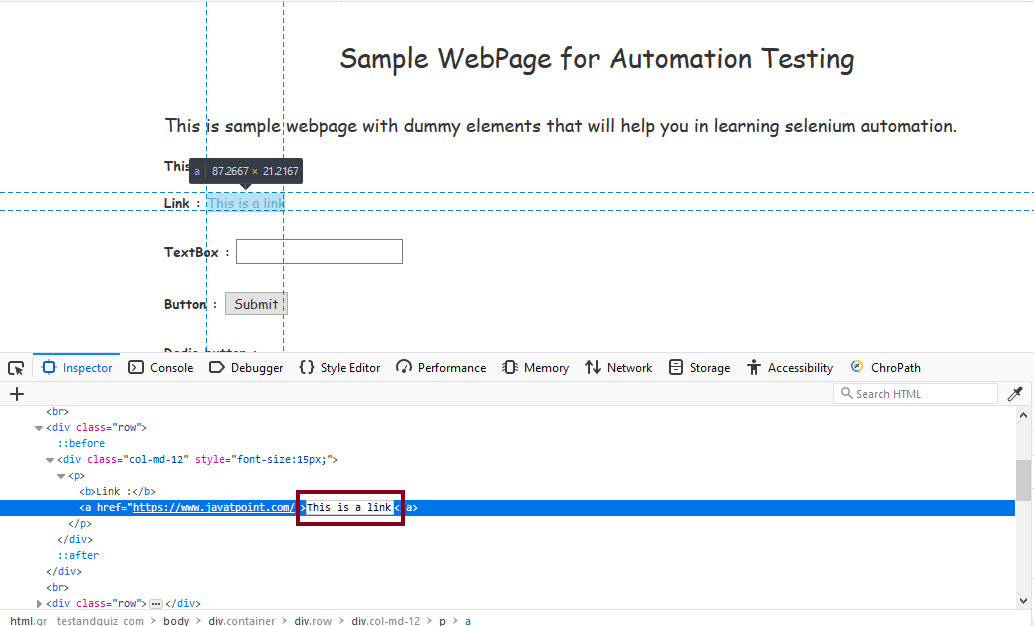
* To automate our third test scenario, first we have to uniquely identify the "This is a link" link on the dummy test page.

The method for finding a unique identification element involves inspection of HTML codes.

1. Open URL: <https://www.testandquiz.com/selenium/testing.html> in your firefox browser.
2. Right click on the "This is a link" link text and select Inspect Element.



It will launch a window containing all the specific codes involved in the development of the "This is a link" link. Select the name of the link text from inspector text box.



The Java Syntax for uniquely identifying a web element through its Link Text is written as:

1. driver.findElement(By.linkText (<linktext>)

Therefore, for locating the Link Text on the sample web page we will use the value of its Link Text:

1. driver.findElement(By.linkText (<"This is a Link">))

Now, we need to write the code which will click on the Link Text.

Here is the sample code to click on the Link Text.

1. // Click on the Link Text using click() command  driver.findElement(By.linkText("This is a Link")).click();

On click, the link will redirect the browser window to the official web page of javaTpoint website.

* To automate our fourth test scenario, we have to revert the action performed by our third test scenario. To do that, we will use the **Back** command to undo the action performed on click of the link text.

Here is the sample code to return to the home page after being directed to the javaTpoint website.

1. // Go back to Home Page
2. driver.navigate().back();

* Now, the next test scenario requires us to again go to the action performed by our third test scenario i.e., the window will again directed to the javaTpoint website.

Here is the sample code to go forward again to the official web page of javaTpoint website.

1. // Go forward to Registration page
2. driver.navigate().forward();

* Now, to automate our sixth test scenario, we will require to again navigate to the home page of dummy website by using the **To command**.

Here is the sample code to go back to the home page.

1. // Go back to Home page
2. driver.navigate().to(appUrl);

* To refresh the browser window, use the **Refresh command** as:

1. // Refresh browser
2. driver.navigate().refresh();

* Finally, the given code snippet will terminate the process and close the browser.

1. driver.close();

Combining all of the above code blocks together, we will get the required source code to execute our test script "Navigation\_command".

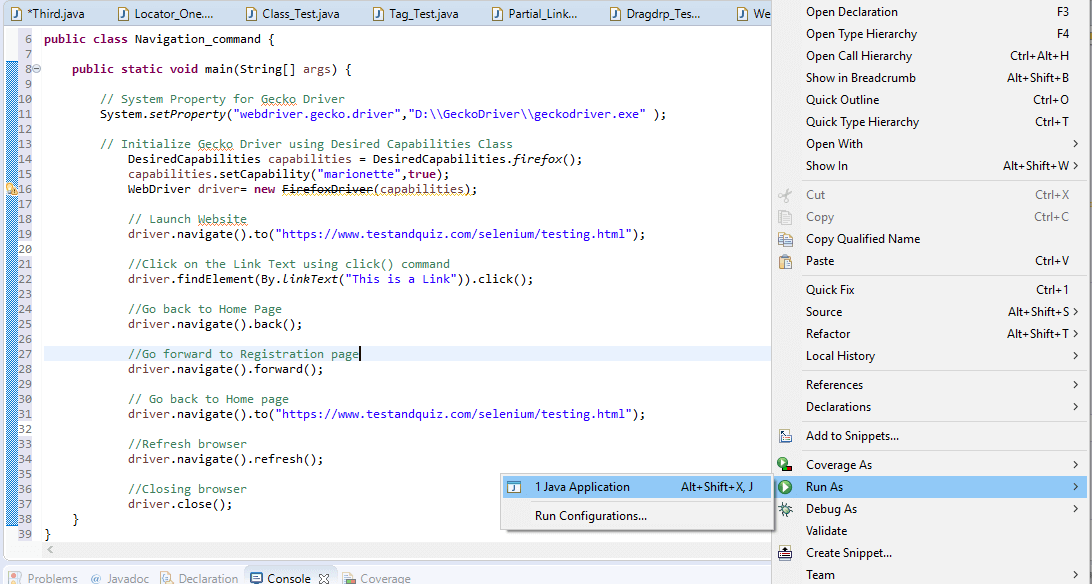
The final test script will appear something like this:

(We have embedded comment in each section to explain the steps clearly)

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.firefox.FirefoxDriver;
4. **import** org.openqa.selenium.remote.DesiredCapabilities;
6. **public** **class** Navigation\_command {
8. **public** **static** **void** main(String[] args) {
10. // System Property for Gecko Driver
11. System.setProperty("webdriver.gecko.driver","D:\\GeckoDriver\\geckodriver.exe" );
13. // Initialize Gecko Driver using Desired Capabilities Class
14. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
15. capabilities.setCapability("marionette",**true**);
16. WebDriver driver= **new** FirefoxDriver(capabilities);
18. // Launch Website
19. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
21. //Click on the Link Text using click() command
22. driver.findElement(By.linkText("This is a Link")).click();
24. //Go back to Home Page
25. driver.navigate().back();
27. //Go forward to Registration page
28. driver.navigate().forward();
30. // Go back to Home page
31. driver.navigate().to("https://www.testandquiz.com/selenium/testing.html");
33. //Refresh browser
34. driver.navigate().refresh();
36. //Closing browser
37. driver.close();
38. }
39. }

To run the test script on Eclipse window, right click on the screen and click

**Run as → Java application**



Upon execution the test script will launch the Firefox browser and automate all the test scenarios.

Next Topic[WebDriver - WebElement Commands](https://www.javatpoint.com/selenium-webdriver-webelement-commands)