SYNTAX
TECHNOLOGIES

Selenium

Class 1

Agenda

What is Selenium? History of the Selenium Project

First Selenium Webdriver script

WebDriver Commands

Selenium

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms

Selenium is not just a single tool but a suite of software, each catering to different testing needs of an organization.

Selenium WebDriver tool is used to automate web applications and verify that it works as expected.

Selenium WebDriver supports many browsers such as Firefox, Chrome, IE, and Safari.

Using the Selenium WebDriver, we can automate testing for web applications only. It does not qualify for window-based applications.

Selenium WebDriver supports different programming languages such as C#, Java, JavaScript, Phyton, PHP and Ruby for writing test scripts.

Selenium Webdriver is platform-independent since the same code can be used on different Operating Systems like Microsoft Windows, Mac OS and Linux.

When to automate?

- Frequent regression testing, whenever a new feature introduced to application we perform functional testing at the same time we also required to perform regression testing to ensure that new changes did not affect the existing functionality negatively.
- Repeated test case execution is required, we may need to test few functionalities again and again for set of data like login details
- Test same application on multiple environments.
- Faster feedback to the developers, when developers deployed something tester must be able to give the details of the deployment effect on basic functionalities of the application.
- Reduce the human effort, when we test a functionality manually it takes lot of time but automating reduces the effort

What type of tests we can automate with selenium

Selenium supports Functional Testing and Regression Testing of Web Applications

- Functional Testing which includes Smoke Testing and Regression Testing.
- Browser Compatibility Testing (Cross browser and Different versions of same browser)
- UI Testing
- Api Testing
- Database Testing using drivers like JDBC can also be performed as it is a part of Functional Testing

Selenium WebDriver Pros

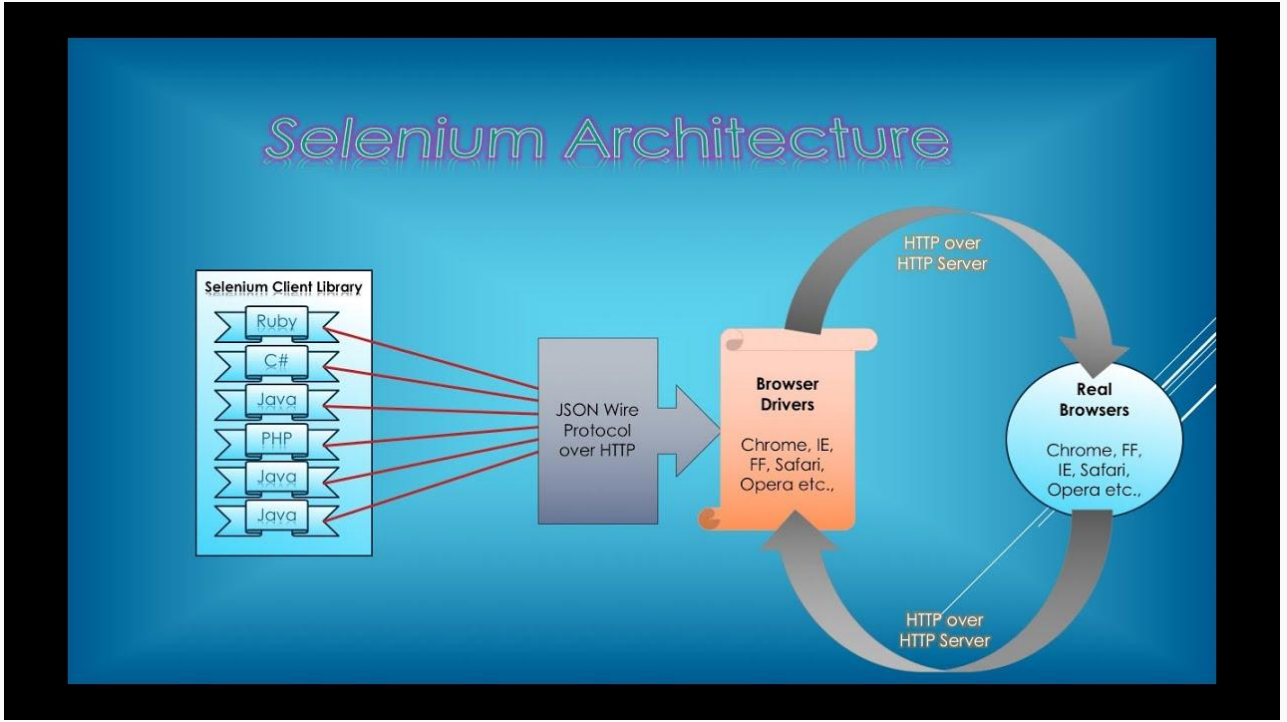
- Selenium webdriver is pure open source, freeware and portable tool
- Selenium webdriver supports many operating systems like Windows, Macintosh, Linux, Unix etc (platform independent)
- It supports all popular web browsers
- It supports parallel test execution
- It can be utilized for regression, user interface, functional and UAT testing.
- Supports to take screenshot of execution
- Selenium can be integrated with continuous integration tools like Jenkins or Bamboo
- Selenium Webdriver support third party tools like AutoIt, Sikuli, Apache POI, with help of the language

Selenium WebDriver Cons

- Since it is open source tool, No reliable Technical support (Official Users Group, Chat room in seleniumhq.org)
- It doesn't support Desktop Applications/Windows based applications
- No Other tool integration for Test Management
- Image verification is not possible
- Additional Tools Required for Generating Reports.
- New features may not work properly(with new browser driver executable like gecko, chrome functionalities like dragging mouse movement).
- Good tests require good coders.

Selenium Webdriver Architecture

Before starting the automation using any automation tool, it is very important to know how that tool works and how it is architecture.



Selenium webdriver architecture divided into 3 parts

1) Language Level Bindings :

Bindings are language level bindings and with which we can implement the Selenium webdriver code. These the languages will interact with the Selenium Webdriver and work on various browsers and other devices. There's Java, Java, Python, Ruby, there's also some other bindings and new bindings can be added very easily.

2) Selenium Webdriver API:
Now these bindings communicate with Selenium Webdriver API and this API send the commands taken from language level bindings interpret it and sent it to Respective driver. In basic term it contains set of common library which allow to send command to respective drivers.

3) Drivers:

We have various internet browser specific drivers such as IE driver, a Firefox, Chrome, and other Drivers such as HTML.

How all blocks work together?

So what's happening here we are going to write test in Java and we are going to be using common **Selenium API** and that **Java binding** is going to be sending command across this common WebDriver API.

Now on the other end is going to be listening a driver, It's going to interpret those commands and it's going to execute them on the actual browser and then it's going to return the result backup using the WebDriver API to our code where we can look at that result.

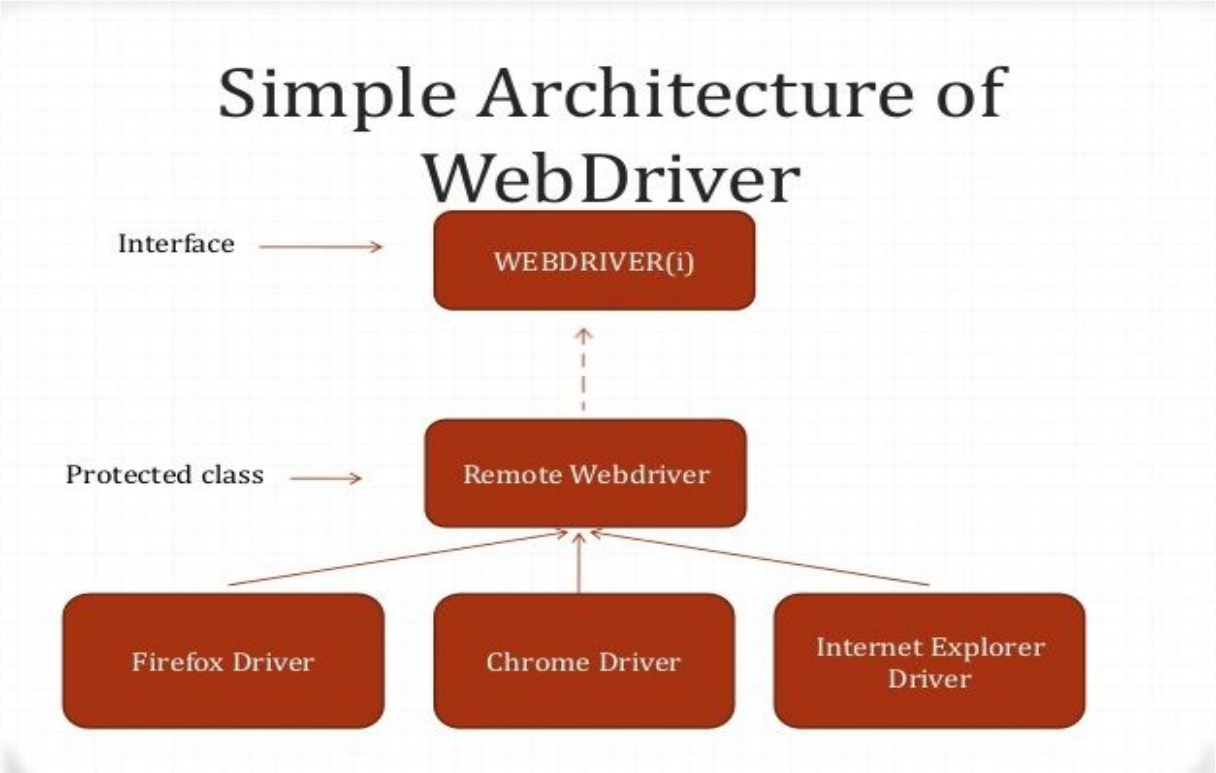
WebDriver Cannot Readily Support New Browsers

- Remember that WebDriver operates on the OS level. Also, remember that different browsers communicate with the OS in different ways.

- If a new browser comes out, it may have a different process of communicating with the OS as compared to other browsers. So, **we have to give the WebDriver team quite some time to figure that new process out** before they can implement it on the next WebDriver release.

- However, it is up to the WebDriver's team of developers to decide if they should support the new browser or not.

Selenium Webdriver Architecture



First Selenium WebDriver Script

As selenium webdriver supports only web based application, opening a browser for operation is must.

We cannot access already opened browser in selenium

To open Chrome Browser
WebDriver driver=new ChromeDriver();

To open Firefox Browser
WebDriver driver=new FirefoxDriver();

To open IE Browser
WebDriver driver=new InternetExplorerDriver();

First Selenium WebDriver Script

WebDriver - webdriver is the interface which is inherited from SearchContext

new - new is the keyword in java which creates an object (address space)in the heap area of CPU

FirefoxDriver() - **FirefoxDriver()** is a constructor of FirefoxDriver class which implements all the methods in the present in the webdriver interface and this opens the firefox Browser .

driver - driver is reference variable ,which refers the address space created on heap.

Steps

1. **Set System Property**

   System.setProperty(key, value)

1. **Instantiate objects and variables**

   WebDriver driver=new ChromeDriver();

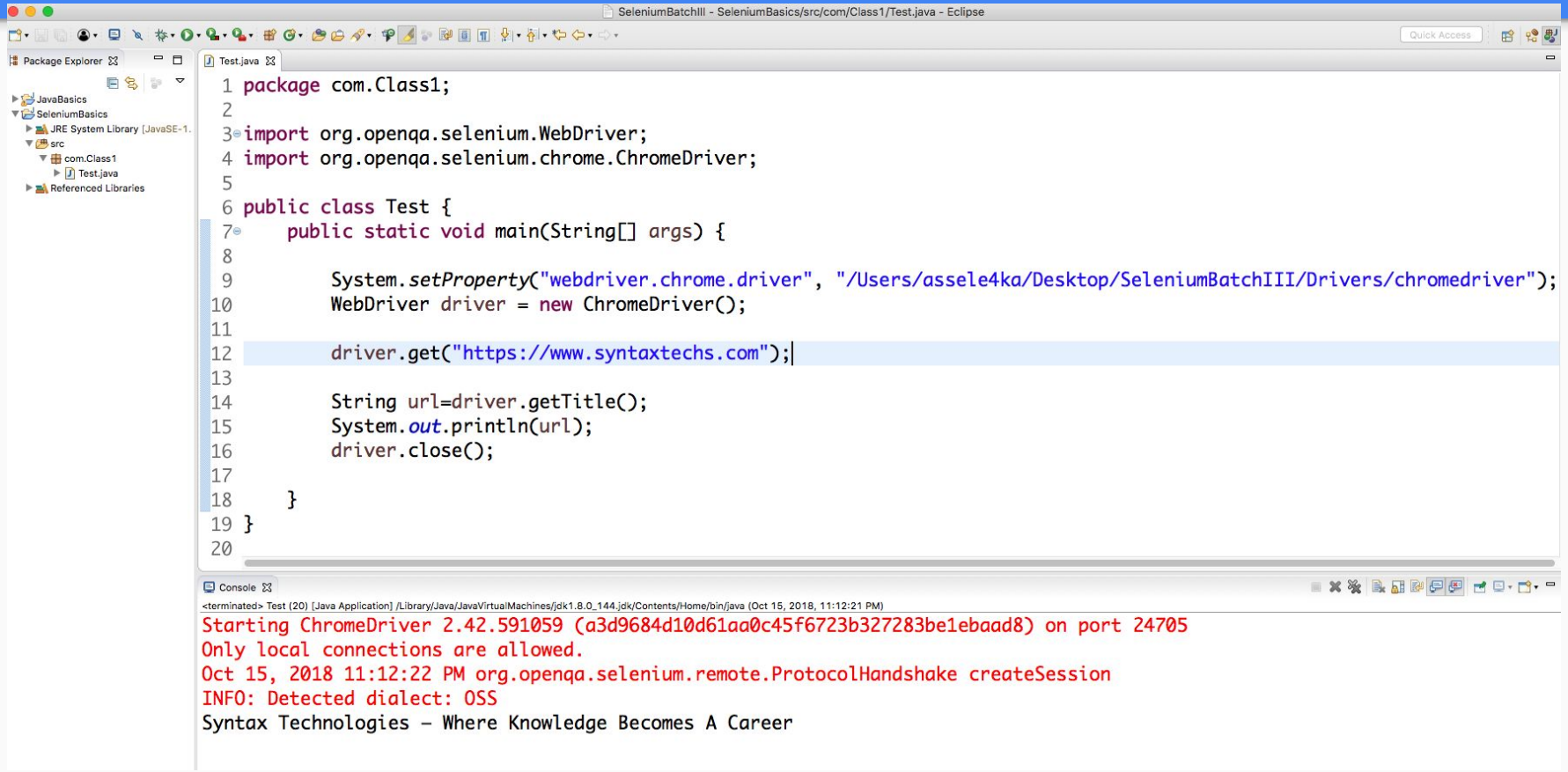
1. **Launch a Browser Session**
   WebDriver's get() method is used to launch a new browser session and directs it to the URL that you specify as its parameter.

1. **Get the Actual Page Title**
   The WebDriver class has the getTitle() method that is always used to obtain the page title of the currently loaded page.

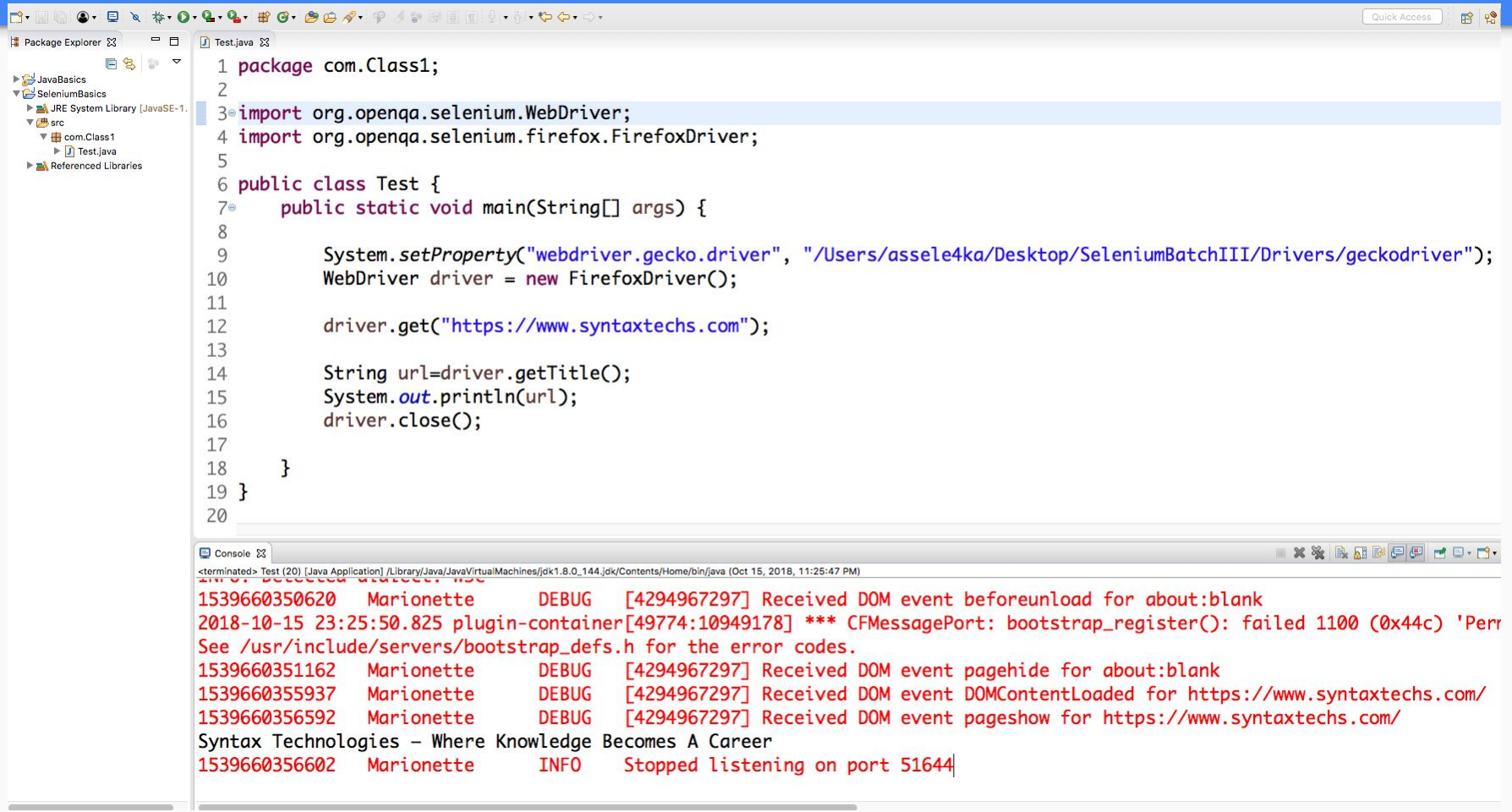
1. **Terminate a Browser Session**

The "close()" method is used to close the browser window.

Chrome Demo


SeleniumBatchIII - SeleniumBasics/src/com/Class1/Test.java - Eclipse
Package Explorer
JavaBasics
SeleniumBasics
JRE System Library [JavaSE-1.
src
com.Class1
Test.java
Referenced Libraries
Test.java
1 package com.Class1;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.chrome.ChromeDriver;
5
6 public class Test {
7 public static void main(String[] args) {
8
9 System.setProperty("webdriver.chrome.driver", "/Users/assele4ka/Desktop/SeleniumBatchIII/Drivers/chromedriver");
10 WebDriver driver = new ChromeDriver();
11
12 driver.get("https://www.syntaxtechs.com");
13
14 String url=driver.getTitle();
15 System.out.println(url);
16 driver.close();
17
18 }
19 }
20
Console
<terminated> Test (20) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Oct 15, 2018, 11:12:21 PM)
Starting ChromeDriver 2.42.591059 (a3d9684d10d61aa0c45f6723b327283be1ebaad8) on port 24705
Only local connections are allowed.
Oct 15, 2018 11:12:22 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Syntax Technologies – Where Knowledge Becomes A Career

Firefox Demo



WebDriver Browser Commands

➔ **get(String url);**

- Command launches a new browser and opens the specified URL in the browser instance
- The command takes a single string type parameter that is usually a URL of application under test

```
driver.get("https://google.com");
//Or can be written as
String URL = "https://google.com";
driver.get(URL);
```

➔ **getTitle();**

- Command is used to retrieve the title of the webpage the user is currently working on.
- A null string is returned if the webpage has no title
- Command doesn't require any parameter and returns a trimmed string value

```
String title = driver.getTitle();
```

WebDriver Browser Commands

➔ **close();**

- This method **Close** only the current window the WebDriver is currently controlling.
- Accepts nothing as a parameter and returns nothing.
- Quit the browser if it's the last window currently open.

driver.close();

➔ **quit();**

- This method **Closes** all windows opened by the WebDriver.
- Accepts nothing as a parameter and returns nothing.
- Close every associated window.

driver.quit();

WebDriver Navigation Commands

➔ **navigate().to(String url);**

- This method **Loads** a new web page in the current browser window. It accepts a String parameter and returns nothing.
- It does exactly the same thing as the **driver.get(String url)** method.
- Able to redirect from the current web page to the expected web page.

```
driver.navigate().to("https://www.google.com");
```

➔ **navigate().refresh();**

- This method ***Refresh*** the current page. It neither accepted nor returns anything.
- Perform the same function as pressing F5 in the browser.

```
driver.navigate().refresh();
```

WebDriver Navigation Commands

➔ **navigate().forward();**

- This method does the same operation as clicking on the **Forward Button** of any browser.
- It neither accept nor returns anything.
- Takes you forward by one page on the browser's history.

driver.navigate().forward();

➔ **driver.navigate().back();**

- This method does the same operation as clicking on the **Back Button** of any browser.
- It neither accepted nor returns anything.
- Takes you back by one page on the browser's history.

driver.navigate().back();

Difference Between get() and navigate()

driver.get("URL")	**driver.navigate().to("URL")**
driver.get("http://www.google.com");	driver.navigate().to("http://www.google.com");
It's used to go to the particular website , But it doesn't maintain the browser History and cookies, we can't use forward and backward button , if we click on that , page will not get schedule	It's used to go to the particular website , but it maintains the browser history and cookies, we can use forward and backward button to navigate between the pages during the coding of Testcase
is used to navigate particular URL(website) and wait till page load.	is used to navigate to particular URL and does not wait to page load