**Q #1) What is Automation Testing?**

Automation testing or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves the use of a separate testing tool which lets you create test scripts which can be executed repeatedly and doesn’t require any manual intervention.

**Q #2) What are the benefits of Automation Testing?**

Benefits of Automation testing are:

1. Supports execution of repeated test cases
2. Aids in testing a large test matrix
3. Enables parallel execution
4. Encourages unattended execution
5. Improves accuracy thereby reducing human-generated errors
6. Saves time and money

**Q #3) Why should Selenium be selected as a test tool?**

Selenium

1. is a free and open source
2. have a large user base and helping communities
3. have cross Browser compatibility (Firefox, Chrome, Internet Explorer, Safari etc.)
4. have great platform compatibility (Windows, Mac OS, Linux etc.)
5. supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
6. has fresh and regular repository developments
7. supports distributed testing

**Q #4) What is Selenium? What are the different Selenium components?**

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web-based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals.

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason, it is referred to as a Suite. Each of these tools is designed to cater different testing and test environment requirements.

The suite package constitutes the following sets of tools:

* [**Selenium Integrated Development Environment (IDE)**](https://www.softwaretestinghelp.com/selenium-ide-download-and-installation-selenium-tutorial-2/) – Selenium IDE is a record and playback tool. It is distributed as a Firefox Plugin.
* **Selenium Remote Control (RC)** – Selenium RC is a server that allows a user to create test scripts in the desired programming language. It also allows executing test scripts within the large spectrum of browsers.
* [**Selenium WebDriver**](https://www.softwaretestinghelp.com/selenium-webdriver-selenium-tutorial-8/) – WebDriver is a different tool altogether that has various advantages over Selenium RC. WebDriver directly communicates with the web browser and uses its native compatibility to automate.
* [**Selenium Grid**](https://www.softwaretestinghelp.com/selenium-grid-selenium-tutorial-29/) – Selenium Grid is used to distribute your test execution on multiple platforms and environments concurrently.

**Q #5) What are the testing types that can be supported by Selenium?**

Selenium supports the following types of testing:

1. Functional cases
2. Regression test cases
3. Acceptance tests
4. Sanity test cases
5. Smoke testing
6. End-to-end test cases
7. Cross-browser tests
8. Integration tests
9. Responsiveness cases

**Q #6) What are the limitations of Selenium?**

Following are the limitations of Selenium:

* Selenium supports testing of only web-based applications
* Mobile applications cannot be tested using Selenium
* Captcha and Barcode readers cannot be tested using Selenium
* Reports can only be generated using third-party tools like TestNG or JUnit.
* As Selenium is a free tool, thus there is no ready vendor support through the user can find numerous helping communities.
* The user is expected to possess prior programming language knowledge.

**Q #10)** **What are the different types of locators in Selenium?**

The locator can be termed as an address that identifies a web element uniquely within the webpage. Thus, to identify web elements accurately and precisely we have [different types of locators in Selenium](https://www.softwaretestinghelp.com/using-selenium-xpath-and-other-locators-selenium-tutorial-5/):

* ID
* ClassName
* Name
* TagName
* LinkText
* PartialLinkText
* Xpath
* CSS Selector
* DOM

**Q #11)** **What is the difference between assert and verify commands?**

**Assert:**Assert command checks whether the given condition is true or false. Let’s say we assert whether the given element is present on the web page or not. If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop and no further test would be executed.

**Verify:**Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn’t halt i.e. any failure during verification would not stop the execution and all the test steps would be executed.

**Q #12) What is an XPath?**

[XPath](https://www.softwaretestinghelp.com/using-selenium-xpath-and-other-locators-selenium-tutorial-5/) is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being markup languages and since they fall under the same umbrella, XPath can be used to locate HTML elements.

The fundamental behind locating elements using XPath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

**Q #13) What is the difference between “/” and “//” in Xpath?**

**Single Slash “/” –**Single slash is used to create Xpath with absolute path i.e. the xpath would be created to start selection from the document node/start node.

**Double Slash “//” –** Double slash is used to create Xpath with relative path i.e. the xpath would be created to start selection from anywhere within the document.

**Q #18) How do I launch the browser using WebDriver?**

The following syntax can be used to launch Browser:  
*WebDriver driver =****new****FirefoxDriver();*  
*WebDriver driver =****new****ChromeDriver();*  
*WebDriver driver =****new****InternetExplorerDriver();*

**Q #19) What are the different types of Drivers available in WebDriver?**

The different drivers available in WebDriver are:

* FirefoxDriver
* InternetExplorerDriver
* ChromeDriver
* SafariDriver
* OperaDriver
* AndroidDriver
* IPhoneDriver
* HtmlUnitDriver

**Q #20) What are the different types of waits available in WebDriver?**

There are two [types of waits available in WebDriver](https://www.softwaretestinghelp.com/selenium-webdriver-waits-selenium-tutorial-15/):

1. Implicit Wait
2. Explicit Wait

**Implicit Wait:**Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, the subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

**Explicit Wait:** Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

**Q #21)** **How to type in a textbox using Selenium?**

The user can use sendKeys(“String to be entered”) to enter the string in the textbox.

**Syntax:**  
*WebElement username = drv.findElement(By.id(“Email”));*  
*// entering username*  
*username.sendKeys(“sth”);*

**Q #22)** **How can you find if an element in displayed on the screen?**

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

1. isDisplayed()
2. isSelected()
3. isEnabled()

**Syntax:**

**isDisplayed():**  
***boolean****buttonPresence = driver.findElement(By.id(“gbqfba”)).isDisplayed();*

**isSelected():**  
***boolean****buttonSelected = driver.findElement(By.id(“gbqfba”)).isSelected();*

**isEnabled():**  
***boolean****searchIconEnabled = driver.findElement(By.id(“gbqfb”)).isEnabled();*

**Q #23)** **How can we get a text of a web element?**

Get command is used to retrieve the inner text of the specified web element. The command doesn’t require any parameter but returns a string value. It is also one of the extensively used commands for verification of messages, labels, errors etc displayed on the web pages.

**Syntax:**  
*String Text = driver.findElement(By.id(“Text”)).getText();*

**Q #24) How to select value in a dropdown?**

The value in the dropdown can be selected using WebDriver’s Select class.

**Syntax:**

**selectByValue:**  
*Select selectByValue =****new****Select(driver.findElement(By.id(“SelectID\_One”)));*  
*selectByValue.selectByValue(“greenvalue”);*

**selectByVisibleText:**  
*Select selectByVisibleText =****new****Select (driver.findElement(By.id(“SelectID\_Two”)));*  
*selectByVisibleText.selectByVisibleText(“Lime”);*

**selectByIndex:**  
*Select selectByIndex =****new****Select(driver.findElement(By.id(“SelectID\_Three”)));*  
*selectByIndex.selectByIndex(2);*

**Q #25) What are the different types of navigation commands?**

Following are the [navigation commands](https://www.softwaretestinghelp.com/selenium-webdriver-waits-selenium-tutorial-15/):  
**navigate().back()** – The above command requires no parameters and takes back the user to the previous webpage in the web browser’s history.

**Sample code:**  
*driver.navigate().back();*

**navigate().forward()** – This command lets the user to navigate to the next web page with reference to the browser’s history.

**Sample code:**  
*driver.navigate().forward();*

**navigate().refresh()** – This command lets the user to refresh the current web page there by reloading all the web elements.

**Sample code:**  
*driver.navigate().refresh();*

**navigate().to()** – This command lets the user to launch a new web browser window and navigate to the specified URL.

**Sample code:**  
*driver.navigate().to(“https://google.com”);*

**Q #26) How to click on a hyper link using linkText?**

*driver.findElement(By.linkText(“Google”)).click();*

The command finds the element using link text and then click on that element and thus the user would be re-directed to the corresponding page.

The above-mentioned link can also be accessed by using the following command.

*driver.findElement(By.partialLinkText(“Goo”)).click();*

The above command finds the element based on the substring of the link provided in the parenthesis and thus partialLinkText() finds the web element with the specified substring and then clicks on it.

**Q #27)** **How to**[**handle frame in WebDriver**](https://www.softwaretestinghelp.com/selenium-tutorial-18/)**?**

An inline frame acronym as iframe is used to insert another document within the current HTML document or simply a web page into a web page by enabling nesting.

**Select iframe by id**  
*driver.switchTo().frame(“ID of the frame“);*

**Locating iframe using tagName**  
*driver.switchTo().frame(driver.findElements(By.tagName(“iframe”).get(0));*

**Locating iframe using index**

**frame(index)**  
*driver.switchTo().frame(0);*

**frame(Name of Frame)**  
*driver.switchTo().frame(“name of the frame”);*

**frame(WebElement element)**  
**Select Parent Window**  
*driver.switchTo().defaultContent();*

**Q #28) When do we use findElement() and findElements()?**

**findElement():**findElement() is used to find the first element in the current web page matching to the specified locator value. Take a note that only first matching element would be fetched.

**Syntax:**

*WebElement element = driver.findElements(By.xpath(“//div[@id='example']//ul//li”));*  
**findElements():**findElements() is used to find all the elements in the current web page matching to the specified locator value. Take a note that all the matching elements would be fetched and stored in the list of WebElements.

**Syntax:**  
*List <WebElement> elementList = driver.findElements(By.xpath(“//div[@id='example']//ul//li”));*

**Q #29)** **How to find more than one web element in the list?**

At times, we may come across elements of the same type like multiple hyperlinks, images etc arranged in an ordered or unordered list. Thus, it makes absolute sense to deal with such elements by a single piece of code and this can be done using WebElement List.

**Sample Code**

|  |  |
| --- | --- |
| 1 | // Storing the list |
| 2 | List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li")); | |

|  |  |
| --- | --- |
| 3 | // Fetching the size of the list |
| 4 | int listSize = elementList.size(); | |

|  |  |  |
| --- | --- | --- |
| 5 | for (int i=0; i<listSize; i++) | |
| 6 | { |

|  |  |  |
| --- | --- | --- |
| 7 | // Clicking on each service provider link | |
| 8 | serviceProviderLinks.get(i).click(); |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | // Navigating back to the previous page that stores link to service providers | | |
| 10 | | driver.navigate().back(); |

|  |  |
| --- | --- |
| 11 | } |

**Q #30) What is the difference between driver.close() and driver.quit command?**

**close()**: WebDriver’s close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does it return any value.

**quit()**: Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does is return any value.

**Q #31) Can Selenium handle windows based pop up?**

Selenium is an automation testing tool which supports only web application testing. Therefore, windows pop up cannot be handled using Selenium.

**Q #32) How can we handle web-based pop-up?**

WebDriver offers the users a very efficient way to [handle these pop-ups using Alert interface](https://www.softwaretestinghelp.com/handle-alerts-popups-selenium-webdriver-selenium-tutorial-16/). There are the four methods that we would be using along with the Alert interface.

* void dismiss() – The dismiss() method clicks on the “Cancel” button as soon as the pop-up window appears.
* void accept() – The accept() method clicks on the “Ok” button as soon as the pop-up window appears.
* String getText() – The getText() method returns the text displayed on the alert box.
* void sendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.

**Syntax:**  
*// accepting javascript alert*  
*Alert alert = driver.switchTo().alert();*  
*alert.accept();*

**Q #33) How can we handle windows based pop up?**

Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support testing of windows based applications. However Selenium alone can’t help the situation but along with some third-party intervention, this problem can be overcome. There are several third-party tools available for handling window based pop-ups along with the selenium like AutoIT, Robot class etc.

**Q #34) How to assert the title of the web page?**

*//verify the title of the web page*  
*assertTrue(“The title of the window is incorrect.”,driver.getTitle().equals(“Title of the page”));*

**Q #35) How to mouse hover on a web element using WebDriver?**

WebDriver offers a wide range of interaction utilities that the user can exploit to automate mouse and keyboard events. Action Interface is one such utility which simulates the single user interactions.

Thus, In the following scenario, we have used Action Interface to mouse hover on a drop down which then opens a list of options.

**Sample Code:**

|  |  |
| --- | --- |
| 1 | // Instantiating Action Interface |
| 2 | Actions actions=new Actions(driver); | |

|  |  |
| --- | --- |
| 3 | // howering on the dropdown |
| 4 | actions.moveToElement(driver.findElement(By.id("id of the dropdown"))).perform(); | |

|  |  |
| --- | --- |
| 5 | // Clicking on one of the items in the list options |
| 6 | WebElement subLinkOption=driver.findElement(By.id("id of the sub link")); | |

|  |  |
| --- | --- |
| 7 | subLinkOption.click(); |
|  |  |

**Q #36) How to retrieve CSS properties of an element?**

The values of the css properties can be retrieved using a get() method:

**Syntax:**  
*driver.findElement(By.id(“id“)).getCssValue(“name of css attribute”);*  
*driver.findElement(By.id(“id“)).getCssValue(“font-size”);*

**Q #37) How to capture screenshot in WebDriver?**

|  |  |
| --- | --- |
| 1 | import org.junit.After; |
| 2 | import org.junit.Before; | |

|  |  |  |
| --- | --- | --- |
| 3 | import org.junit.Test; | |
| 4 | import java.io.File; |

|  |  |
| --- | --- |
| 5 | import java.io.IOException; |
| 6 | import org.apache.commons.io.FileUtils; | |

|  |  |
| --- | --- |
| 7 | import org.openqa.selenium.OutputType; |
| 8 | import org.openqa.selenium.TakesScreenshot; | |

|  |  |  |
| --- | --- | --- |
| 9 | import org.openqa.selenium.WebDriver; | |
| 10 | | import org.openqa.selenium.firefox.FirefoxDriver; | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | public class CaptureScreenshot { | |

|  |  |  |
| --- | --- | --- |
| 13 | WebDriver driver; | |
| 14 | @Before |

|  |  |  |
| --- | --- | --- |
| 15 | public void setUp() throws Exception { | |
| 16 | driver = new FirefoxDriver(); |

|  |  |  |
| --- | --- | --- |
| 17 | driver.get("https://google.com"); | |
| 18 | } |

|  |  |
| --- | --- |
| 19 | @After |
| 20 | public void tearDown() throws Exception { | |

|  |  |  |
| --- | --- | --- |
| 21 | driver.quit(); | |
| 22 | } |

|  |  |
| --- | --- |
| 23 |  |
| 24 | @Test | |

|  |  |  |
| --- | --- | --- |
| 25 | public void test() throws IOException { | |
| 26 | // Code to capture the screenshot |

|  |  |  |
| --- | --- | --- |
| 27 | File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); | |
| 28 | // Code to copy the screenshot in the desired location |

|  |  |  |
| --- | --- | --- |
| 29 | FileUtils.copyFile(scrFile, new File("C:\\CaptureScreenshot\\google.jpg")) | |
| 30 | } |

|  |  |
| --- | --- |
| 31 | } |

**Q #41)** **How to set test case priority in TestNG?**

**Setting Priority in TestNG**

**Code Snippet**

|  |  |
| --- | --- |
| 1 | package TestNG; |
| 2 | import org.testng.annotations.\*; | |

|  |  |  |
| --- | --- | --- |
| 3 | public class SettingPriority { | |
| 4 | @Test(priority=0) |

|  |  |  |
| --- | --- | --- |
| 5 | public void method1() { | |
| 6 | } |

|  |  |
| --- | --- |
| 7 | @Test(priority=1) |
| 8 | public void method2() { | |

|  |  |  |
| --- | --- | --- |
| 9 | } | |
| 10 | | @Test(priority=2) | |

|  |  |  |
| --- | --- | --- |
| 11 | public void method3() { | |
| 12 | } |

|  |  |
| --- | --- |
| 13 | } |

**Test Execution Sequence:**

1. Method1
2. Method2
3. Method3

**Q #42) What is a framework?**

The framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

**Q #43)** **What are the advantages of the Automation framework?**

**The advantage of**[**Test Automation framework**](https://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/)

* Reusability of code
* Maximum coverage
* Recovery scenario
* Low-cost maintenance
* Minimal manual intervention
* Easy Reporting

**Q #44) What are the different types of frameworks?**

**Below are the different types of frameworks:**

1. **Module Based Testing Framework:** The framework divides the entire “Application Under Test” into the number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts have taken together builds a larger test script representing more than one module.
2. **Library Architecture Testing Framework:** The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.
3. Data Driven Testing Framework: Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.
4. **Keyword Driven Testing Framework:** The Keyword Driven testing framework is an extension to Data-driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.
5. **Hybrid Testing Framework:** Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
6. **Behavior Driven Development Framework:** Behavior Driven Development framework allows automation of functional validations in an easily readable and understandable format to Business Analysts, Developers, Testers, etc.

**Q #48) Can WebDriver test Mobile applications?**

WebDriver cannot test Mobile applications. WebDriver is a web-based testing tool, therefore applications on the mobile browsers can be tested.

**Q #49) Can captcha be automated?**

No, captcha and barcode reader cannot be automated.

**Q #50) What is Object Repository? How can we create an Object Repository in Selenium?**

Object Repository is a term used to refer to the collection of web elements belonging to Application Under Test (AUT) along with their locator values. Thus, whenever the element is required within the script, the locator value can be populated from the Object Repository. Object Repository is used to store locators in a centralized location instead of hardcoding them within the scripts.

In Selenium, objects can be stored in an excel sheet which can be populated inside the script whenever required.

That’s all for now.

Hope in this article you will find answers to most frequently asked Selenium and WebDriver Interview questions. The answers provided here are also helpful for understanding the Selenium basics and advanced WebDriver topics.

Which Of The Id, Name, XPath Or CSS Selector Should You Use?

If the page has unique names or identifiers available, then we should use them.

If they are not available, then go for a CSS selector as it is faster than the XPath.

When none of the preferred locators is present, then you may try the XPath.

What Is XPath? How Does It Work?

XPath is the most-used locator strategies Selenium uses to find web elements.

* It works by navigating through the DOM elements and attributes to locate the target object. For example – a text box or a button or checkboxes.
* Although, it guarantees to give you the element you are looking after. But it is slower than as compared to other locators like ID, name or CSS selectors.

What Is An Absolute XPath, Explain With Example?

An absolute XPath will always search from the root node until it reaches the target. Such an XPath expression includes the single forward slash (/) as the prefix.

/html/body/div[1]/div[5]/form/table/tbody/tr[3]/td/input

What Is A Relative XPath, Explain With Example?

A relative XPath doesn’t have a specific point to start. It can begin navigation from any node inside the DOM and continues. Such an XPath expression includes the double forward slash (//) as given below.

//input[@id='username']

How Do You Locate An Element By Partially Comparing Its Attributes In XPath?

XPath supports the contains() method. It allows partially matching of attribute’s value.

It helps when the attributes use dynamic values while having some fixed part.

See the below example-

xPath usage => //\*[contains(@category, 'tablet')]

The above expression would match all values of category attribute having the word ‘tablet’ in them.

How Do You Locate Elements Based On The Text In XPath?

We can call the text() method. The below expression will get elements that have text nodes that equal ‘Python.’

xPath usage = //\*[text()='Python']

How Do You Access The Parent Of A Node With XPath?

We can use the double dot (“..”) to point to the parent of any node using the XPath.

For example – The locator **//span[@id=”current”]/.**. will return the parent of the span element matching id value as ‘current’.

How Do You Get To The Nth Sub-Element Using The XPath?

We can modify the XPath expression to get to the nth element in the following ways:

**1.** Use XPath as an array by appending the square brackets with an index.

# Example

tr[2]

The above XPath expression will return the second row of a table.

**2.** By calling position() in the XPath expression

# Example

tr[position()=4]

The above XPath will give the fourth row.

How Do You Use “Class” As A CSS Selector?

We can use the below syntax to access elements using the class CSS selector.

.<class>

e.g. .color

It can help to select all elements related to the specified class.

How Do You Use “ID” As A CSS Selector?

We can use the below syntax to access elements using ID as the CSS selector.

#<ID>

e.g. #name

How To Specify Attribute Value While Using The CSS Selector?

Here is the syntax to provide the attribute value with the CSS selector.

[attribute=value]

e.g. [type=submit]

How To Access The Nth Element Using The CSS Selector?

Here is the syntax to access the nth attribute using the CSS selector.

<type>:nth-child(n)

e.g. tr:nth-child(4)

What Is The Primary Difference Between The XPath And CSS Selector?

With the XPath, we can traverse both forward and backward whereas CSS selector only moves forward.

What Is The Selenium Command To Fetch The Current Page URL?

To retrieve the current page URL, we can call the getCurrentURL() function.

webdriver.getCurrentUrl();

What Is The Selenium Command To Set The Browser Maximized?

We can maximize the browser window by calling Selenium’s maximize() method.

It sets the current window in the maximized state.

webdriver.manage().window().maximize();

What Is The Selenium Command To Delete Session Cookies?

To delete session cookies, we can invoke the deleteAllCookies() method.

webdriver.manage().deleteAllCookies();

State The Difference Between Web Driver’s GetWindowHandle() And GetWindowHandles() Methods?

webdriver.getWindowHandle() – It gets the handle of the active web page.

webdriver.getWindowHandles() – It gets the list of handles for all the pages opened at a time.

State The Difference Between Web Driver’s Close() And Quit() Methods?

How Do You Check If An Object Is Present On Multiple Pages?

We can use the isElementPresent() command to verify the object on all pages.

assertTrue(selenium.isElementPresent(locator));

How Do You Check For The Presence Of A Web Element After The Successful Page Load?

We can verify the presence of a web element with the following code.

While using the below function, do supply some timeout value (in seconds) to check the element in a regular interval.

public void checkIfElementPresent(String element, int timeout) throws Exception {

for (int sec = 0;; sec++) {

if (sec >= timeout)

fail("Timeout! Couldn't locate element." + element);

try {

if (selenium.isElementPresent(element))

break;

} catch (Exception ex) {

}

Thread.sleep(1000);

}

}

What Are The Challenges Have You Faced With Selenium? And How Did You Overcome Them?

Here are some of the problems that testers usually face while doing automation with Selenium.

* **Wrong implementation:** I used the [**page object model**](https://www.techbeamers.com/implement-page-object-model-pom-with-selenium-and-web-driver-2-0/) but had it implemented incorrectly. My classes were focussing on the web elements rather than they should have resembled the user actions.
* **Duplicate code:** The project had many category pages. Each category had a different search function instead of handling them at a central place.
* **Ineffective use of wait:** I used implicit wait with a fixed timeout. But some pages were timing out due to higher load time. I had to adopt the Fluent wait (with a variable timeout) to overcome this problem.
* **Improper error handling:** It was getting hard to debug the cause of a failed test. At some places, the {try-catch} blocks were missing and hence cases were skipping w/o giving a proper message. Therefore, I had to refactor the code by adding asserts and exception handling.
* **Inconsistent XPath:** Most of the locators were using the XPath method. And the developers kept them changed while fixing new defects. I called up a discussion with them and agreed to have a fixed XPath or an ID for the web elements.
* **Performance & Localization:** We were using the flat files (CSV) initially to feed data to test cases. However, it had us failed in testing localization as well as beaten us on the performance. Ee migrated all of our test data to MySQL and fixed both issues.
* **Monolithic tests:** Earlier tests weren’t using the labeling. Honestly, there wasn’t a way to do it. Hence, we integrated our test suite with TestNG and got away with this limitation. Now, we have many test groups like features-based (F1, F2, F3…), priority-based (P1, P2, P3).