**SELENIUM INTRODUCTION**

**Introduction**

Selenium automates browsers. it is for automating web applications for testing purposes.

Selenium is a set of different software tools each with a different approach to supporting test automation.

Selenium was created by Jason Huggins in 2004.

Selenium supports the following types of testing:

* Functional Testing
* Regression Testing

**Features of selenium**

* Selenium is an open source and portable Web testing Framework.
* Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
* It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
* Selenium supports various operating systems, browsers and programming languages. Following is the list:
  + Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
  + Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
  + Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
* It also supports parallel test execution which reduces time and increases the efficiency of tests.
* Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
* Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
* Selenium requires fewer resources as compared to other automation test tools.
* WebDriver API has been indulged in selenium whichis one of the most important modifications done to selenium.
* Selenium web driver does not require server installation, test scripts interact directly with the browser.

**Difference between IDE, RC, Webdriver**

| **Selenium IDE** | **Selenium RC** | **Selenium WebDriver** |
| --- | --- | --- |
| **1.** It brings one of the most valuable record and playback feature. | **1.** You can’t use it for recording and playback. | **1.** You can’t use it for recording and playback. |
| **2.** It is independent of Selenium server to run the test script. | **2.**It uses the Selenium server before processing the test script. | **2.** Selenium server isn’t needed anymore to run the test script. |
| **3.**It has a UI interface to work with the test scripts. | **3.** It’s a standalone Java (jar) application which runs Html test suites in the browser. | **3.**It is a full-fledged API and languages like Java, Python, and CSharp implement Webdriver APIs. |
| **4.** Its core engine depends on the Javascript. | **4.**Javascript drives its core engine. | **4.** It has native integration with the browser. |
| **5.** You can readily use its record & playback feature. | **5.** You can easily learn its API set. | **5.**Its APIs are a bit complicated than the ones in Selenium RC. |
| **6.**It uses Selenese which is a procedural language. | **6.** Its API’s are not fully object oriented. | **6.** Its APIs fully support the object oriented approach. |
| **7.**You may not use it to move mouse cursors. | **7.** You may not use it to move mouse cursors. | **7.** It allows managing the movement of mouse cursors. |
| **8.** You’ve to use the absolute XPath value with ‘xpath=\\’ argument. | **8.** Requires full XPath for locating elements like ‘xpath=”\\html\div[1]\input’. | **8.** It’s optional to use the full XPath value with ‘xpath=\\’ syntax. |
| **9.** Listeners support- not allowed. | **9.** Listeners support- not applicable. | **9.**Listeners support- Yes. |
| **10.** iPhone/Android applications – not supported. | **10.** iPhone/Android applications – NA. | **10.** iPhone/Android applications – Yes. |

**SELENESE-Selenium Commands**

A *command* is what tells Selenium to do.

Selenium commands come in three flavors :

* + - Actions
    - Accessors
    - Assertions

**Actions**

Commands which change the state of the application are classified as actions, like click on some link, select some options on the page, select a value from drop down etc. When action is performed on page the test will fail if the action is not successful. Most common action commands of selenium are:

a.ClickAndWait  
b. Click

**Accessors**

These commands check the state of the application and stores the application state in some variable.

Some examples of accessors are as follows.

a.storeTitle  
b. StoreText Accessors are also used to evaluate whether the desired result is present on the page and store the result in the variable.

**Assertions**

Assertions are used like the checkpoints or verification points in automation. Assertion verifies the state of the application conforms to the expected state. **Example:**

a.verifyText  
b.verifyTitle  
c.assertText

All Selenium Assertions can be used in 3 modes: “assert”, “verify”, and ” waitFor”. For example, you can “assertText”, “verifyText” and “waitForText”. When an “assert” fails, the test is aborted. When a “verify” fails, the test will continue execution, logging the failure. This allows a single “assert” to ensure that the application is on the correct page, followed by a bunch of “verify” assertions to test form field values, labels, etc.

“waitFor” commands wait for some condition to become true .They will succeed immediately if the condition is already true. However, they will fail and halt the test if the condition does not become true within the current timeout setting.

**Commonly Used Selenium Commands**

**open**

opens a page using a URL.

**click/clickAndWait**

performs a click operation, and optionally waits for a new page to load.

**verifyTitle/assertTitle**

verifies an expected page title.

**verifyTextPresent**

verifies expected text is somewhere on the page.

**verifyElementPresent**

verifies an expected UI element, as defined by its HTML tag, is present on the page.

**verifyText**

verifies expected text and its corresponding HTML tag are present on the page.

**verifyTable**

verifies a table’s expected contents.

**waitForPageToLoad**

pauses execution until an expected new page loads. Called automatically when clickAndWait is used.

**waitForElementPresent**

pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

### verifyTextPresent

The command verifyTextPresent is used to verify specific text exists somewhere on the page. It takes a single argument–the text pattern to be verified. For example:

| **Command** | **Target** | **Value** |
| --- | --- | --- |
| verifyTextPresent | Marketing Analysis |  |

This would cause Selenium to search for, and verify, that the text string “Marketing Analysis” appears somewhere on the page currently being tested. Use verifyTextPresent when you are interested in only the text itself being present on the page. Do not use this when you also need to test where the text occurs on the page.

**Locators**

For many Selenium commands, a *target* is required.

This *target* identifies an *element* in the web page and consists of the location strategy followed by the location in the format *locatorType=location*

*Page code*

<html>

<body>

<form id="loginForm">

<input name="username" type="text" />

<input name="password" type="password" />

<input name="continue" type="submit" value="Login" />

</form>

</body>

<html>

* ***identifier***: Common and default method of locating elements.

Eg: identifier = loginForm----(3)

identifier = password---(5)

* ***id*:** This type of locator is more limited than the identifier locator type, but also more explicit. Use this when you know an element’s id attribute.

Eg: id=loginForm---(3)

* name :The name locator type will locate the first element with a matching name attribute..

Eg: name= username----(4)

Xpath

XPath known as the XML path is a language that helps to query the XML documents. It consists of expression for a path along with certain conditions to locate a particular element**.**

General syntax:

//tagname[@attribute name= ‘value’]

Here

1. **//** Denotes the current node
2. **Tagname :** Define the tagname you are referencing to locate the element.
3. **Attribute Value :** The attribute of the define tag through which you wish to narrow down the search.
4. Value: Represents the value of any chosen attribute.

## Types Of XPath In Selenium

* Absolute xpath
* Relative xpath

**Absolute xpath**

XPath expression is created using the selection from the root node. It starts with a single slash ‘/’ and traverses from the root to the whole DOM to reach to the desired element. The biggest disadvantage of using this as locating an element is, if during the course of development any changes made in the path, may lead to a failed XPath expression.

For eg:

**/html/body/div[1]/section/div/div[2]/div/form/div[2]/input[3]**

**Relative xpath**

XPath expression is generated from the middle of the DOM structure. It is represented by a double slash ‘//’ denoting the current node. In this case the search will start from the mentioned tagname and string value. It is more compact, easy to use and less prone to been broken. For example:

For eg :

**//input[@name=’email’]**

Basic xpath

Xpath=//input[@type='text']

Xpath= //label[@id='message23']

Xpath= //input[@value='RESET']

Xpath=//\*[@class='barone']

Xpath=//a[@href='http://demo.guru99.com/']

Xpath= //img[@src='//cdn.guru99.com/images/home/java.png']

**2) Contains():**

Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically.

The contain feature has an ability to find the element with partial text.

Xpath=//\*[contains(@type,'sub')]

Xpath=//\*[contains(@name,'btn')]

Xpath=//\*[contains(text(),'here')]

Xpath=//\*[contains(@href,'guru99.com')]

### Using OR & AND:

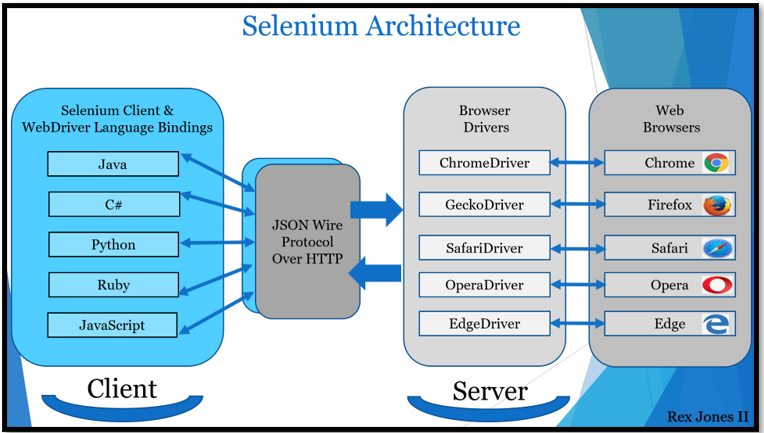
In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true.

Xpath=//\*[@type='submit' or @name='btnReset']

Xpath=//input[@type='submit' and @name='btnLogin']

**WebDriver**

**SELENIUM 3 Architecture**



The architecture for Selenium 3 includes the json wire protocol. JSON stands for **J**ava**S**cript**O**bject**N**otation.

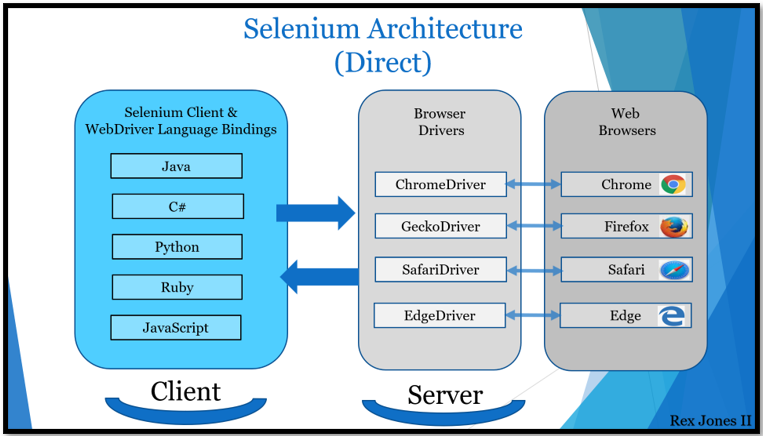
The JSON Wire Protocol has an assignment to transfer information from the client to the server over HTTP. A Selenium request is sent from the Selenium Client and WebDriver Language Bindings component. Next, the request is received by JSON Wire Protocol Over HTTP, then secured by the Browser Driver.

Afterwards, the request command is delivered to a Web Browser where the automation takes place. When the automation is complete, a response travels back to the Browser Driver, JSON Wire Protocol, and Selenium Client & WebDriver Language Bindings.

## **Selenium 4 Architecture**

Unlike Selenium 3, Selenium 4 has direct communication between the client and server. The *client* still has 2 parts (Selenium Client & WebDriver Language Bindings) while Browser Drivers are the *server*.

* **Selenium Client** sends out a request to perform a command.
* The **WebDriver Language Bindings** is a code library designed to drive actions.
* **Browser Drivers** receive the request and then return a response after an automation Test Script executes on the **Web Browser**:



The Selenium Client & WebDriver Language Bindings is a part of the architecture where each language has their own unique bindings. Bindings mean that the same commands written for one language are also written for another language. For example, Java has a set of commands that have also been written for other languages (C#, Python JavaScript, and Ruby).

When it comes to the Browser Drivers and Web Browsers, WebDriver drives each browser using the browser’s built-in automation support. A Browser Driver such as ChromeDriver controls the Chrome browser.

In Selenium 4, a change was made to the drivers. ChromeDriver and EdgeDriver extend ChromiumDriver while RemoteWebDriver is the parent to ChromiumDriver.

## **Selenium 4 Advantages**

The advantages are:

* **Standards**
* **Stability**
* **Updated Actions API**

The standards are an advantage because our Test Scripts run more consistently on each browser. All browser vendors have a standard. Since Selenium 4 is compliant with W3C WebDriver, there is no more required encoding and decoding of the request.

Stability is another advantage because of backwards compatibility. The Java Bindings and the Selenium Server provide a mechanism to use the old JSON Wire Protocol. There have been updates to the Actions API for keyboard and mouse events. It supports a way to carry out more than one action at the same time, like pressing 2 keys.

**Selenium waits**

when a page load on a browser various webelements on it with may load at different time intervals not only it makes the difficult to identify elements but also the element is not located it would throw an element not visible exception using selenium wait we can resolve this problem.

Waits in selenium

* implicit wait
* explicit wait
* fluent wait

**Implicit wait(global wait)**

In implicit wait we can tell selenium that we would like it to wait for a certain amount of time before throwing exception that it cannot find the element on the page.

**Synatx: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))**

**driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));**

The element is not located on the webpage within the timeframe it will throw an exception.

However, implicit wait increases test script execution time. It makes each command wait for the defined time before resuming test execution. If the application responds normally, implicit wait can slow down the execution of test scripts.

**Explicit wait**

**//thread.sleep(3000)**

Explicit wait is used to tell webdriver to wait for certain conditions(expected conditions) or the maximum time exceeded before throwing an element visible exception.

**Syntax:**

**webDriverwait wait=new WebDriverWait(driver,Duration.ofSeconds)**

Here, the reference variable is named <wait> for the <WebDriverWait> class. It is instantiated using the WebDriver instance. The maximum wait time must be set for the execution to layoff. Note that the wait time is measured in seconds.

**wait.until(ExceptedConditions.visibilityofelementLocated(By.xpath(“ ”))**

The following Expected Conditions can be used in Explicit Wait.

* alertIsPresent()
* elementSelectionStateToBe()
* elementToBeClickable()
* elementToBeSelected()
* frameToBeAvaliableAndSwitchToIt()
* invisibilityOfTheElementLocated()
* invisibilityOfElementWithText()
* presenceOfAllElementsLocatedBy()
* presenceOfElementLocated()
* textToBePresentInElement()
* textToBePresentInElementLocated()
* textToBePresentInElementValue()
* titleIs()
* titleContains()
* visibilityOf()
* visibilityOfAllElements()
* visibilityOfAllElementsLocatedBy()
* visibilityOfElementLocated()

**Difference between implicit and explicitwait**

* Implicit waits are used to provide a default waiting time between each consecutive test step/command across the entire test script .
* Explicit waits are used to halt the execution until the time a particular condition is met or the maximum time has elapsed
* Implicit wait are applied to all the elements in the script
* Explicit wait is applied only to those elements which are intended by us
* In implicit wait we need not specify any expected conditionsnon the element to be located
* In explicit wait we need to specify the expected conditions on the element to be located.

**Javascriptexecutor in selenium**

* JavascriptExecutor is an interface that is used to execute javascript with selenium.JavaScript is a programming language that interacts with HTML in a browser, and to use this function in Selenium, JavascriptExecutor is required
* JavaScriptExecutor is used when Selenium Webdriver fails to click on any element due to some issue.
* JavaScriptExecutor provides two methods “executescript” & “executeAsyncScript” to handle.
* Actions we can perform using javascriptexecutor

Flashing an elements

* + Navigated to different page
  + Generate alert
  + Refreshing page
  + Scroll down etc

**Scroll down using javascriptexecutor by defining the pixel**

**syntax**

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("window.scrollBy(0,250)", "");

**Scroll down using javascriptexecutor until an element is visible**

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].scrollIntoView();", Element);

**Scrolldown to the bottom of the page**

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("window.scrollBy(0,document.body.scrollHeight)”);

**Assertions in selenium**

Assertions are used for validating a test case and helps us understand if a test case has passed or failed. The assertion is considered to be met if the actual result of an application matches with that of the expected result.While automating web applications using Selenium, we need to validate our tests to verify if they are working as expected or not.

There are two types of assertions in Selenium and the categorization depends on how the assertion behaves after a condition is pass or fail.

* Hard Assertions
* Soft Assertions

**Hard Assertions**

Hard assertions usually throw an Assertion Error whenever an assertion condition has not been met. The test case will be immediately marked as Failed when a hard assertion condition fails.

**Soft Assertions**

A soft assertion continues with the next step of the test execution even if the assertion condition is not met.

Soft Assertions are the type of assertions that do not throw an exception automatically when an assertion fails unless it is asked for. This is useful if you are doing multiple validations in a form, out of which only a few validations directly have an impact on deciding the test case status.

**Actions class in selenium**

The user-facing API for emulating complex user gestures. Use this class rather than using the Keyboard or Mouse directly.

Implements the builder pattern: Builds a CompositeAction containing all actions specified by the method calls.

Actions class is present in org.openqa.selenium.interactions.Actions package. It extends super class object.

**Action in Selenium** is an interface that provides us two methods: perform() and build(). These two methods or commands of action interface are implemented by the actions class

**build()**

The build() method of action interface generates a composite action that contains all the actions gathered which are ready to be performed. All actions to be performed are specified by method calls.

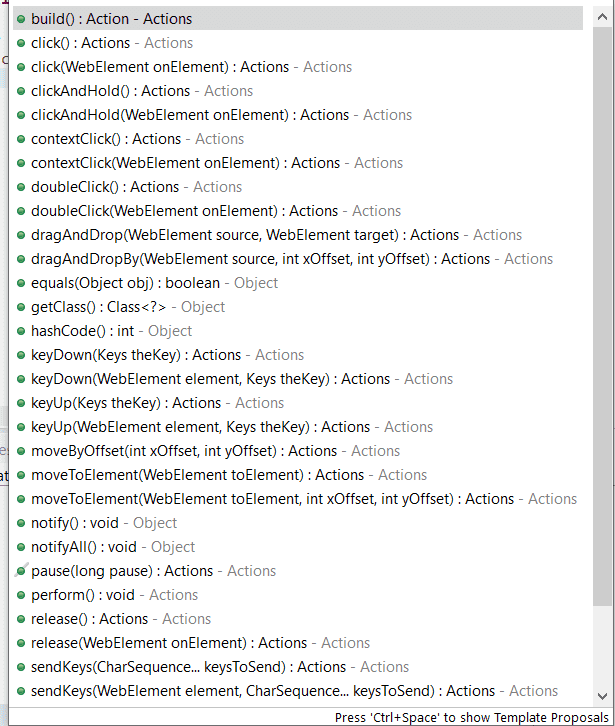
### **perform()**

This method is used to perform a sequence of actions without calling build() first.

The syntax to create an object of actions class is as follows:

Actions actions = new Actions( WebDriver driver);

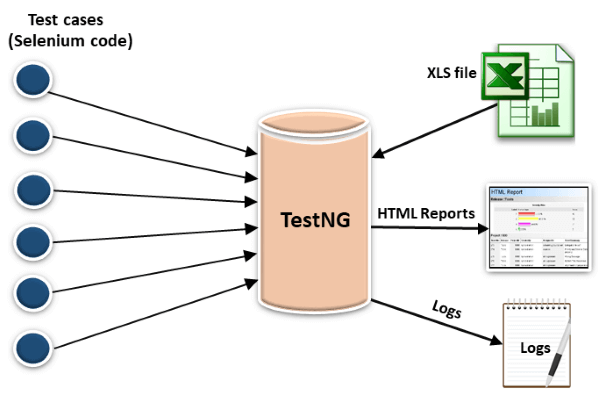
When you will create an object of the actions class and pass the WebDriver object as a parameter to its constructor, you can access all the methods provided by the Actions class by just typing actions + dot.



**TestNG**

* TestNG is a very important framework when you are actually developing the framework from scratch level.
* TestNG provides you full control over the test cases and the execution of the test cases. Due to this reason, TestNG is also known as a testing framework.
* Cedric Beust is the developer of a TestNG framework.
* If you want to run a test case A before that as a pre-request you need to run multiple test cases before you begin a test case A. You can set and map with the help of TestNG so that pre-request test cases run first and then only it will trigger a test case A. In such way, you can control the test cases.
* TestNG framework came after Junit, and TestNG framework adds more powerful functionality and easier to use.
* It is an open source automated TestNG framework. In TestNG, NG stands for "Next Generation".
* TestNG framework eliminates the limitations of the older framework by providing more powerful and flexible test cases with help of easy annotations, grouping, sequencing and parametrizing.

**Advantages of TestNG over Junit**



* In TestNG, annotations are easier to understand than Junit.
* It produces the HTML reports for implementation.
* It also generates the Logs.
* TestNG enables you to group the test cases easily which is not possible in JUnit.
* TestNG supports three additional levels such as @Before/After suite, @Before/AfterTest, and Before/AfterGroup.
* TestNG does not extend any class. TestNG framework allows you to define the test cases where each test case is independent of other test cases.
* It allows you to run the test cases of a particular group. Let's consider a scenario where we have created two groups such as 'Smoke' and 'Regression'. If you want to execute the test cases in a 'Regression' group, then this can only be possible in the TestNG framework.
* Parallel execution of test cases, i.e., running multiple test cases is only possible in the TestNG framework.

**POM**

* **Page Object Model**, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements. It is useful in reducing code duplication and improves test case maintenance .
* This Design Pattern is used in Selenium where web pages are represented by a corresponding class and web elements are represented by the variables of the class and all interactions are possible through the methods or say functions of the class.

**Selenium Maven**

**Maven**

Maven is a POM (project object model) based build automation and project management tool written in Java. However, it is compatible with projects written in C#,python, Ruby. Developers or Automation testers face a common problem while using versions of JAR’s/dependencies as all of their code is placed in Eclipse, IntelliJ.

* Maven is a build automation tool provided by Apache
* It builds your script
* It automatically downloads all jars u need in project
* It creates a standard structure of project that differentiate test files,resources,flat files,library files,main files
* It can be added as a plugin in eclipse
* In maven project all configurations are written in pom.xml file
* Its plugin are very useful - core plugin are clean,deploy,compiler-Additional plugins are - postman plugin which can send test report to an email

### ****Maven Dependency****

For a smaller project with few functionalities or modules, a user can manually download JAR’s or dependencies and can add in the project. However, if a user is working on a medium or large project with many modules and a big team, then Maven plays a vital role in project management. It is a challenging task to ensure that all the team members are using the same dependency version. This issue can be solved by standardizing the dependency version for all.

Maven is widely used for dependency management in Java. It also provides a predefined folder structure to write the code. We can add different plugins and JARs in our project.

### ****Key Advantages of Maven****

* One can easily configure dependencies required for building, testing and running code using pom.xml
* Maven downloads the required file automatically from the repository
* Using Maven, one can manage the entire life-cycle of a test project

## ****Selenium framework.****

Selenium framework is a code structure for making code maintenance simpler, and code readability better. A framework involves breaking the entire code into smaller pieces of code, which test a particular functionality.

The code is structured such that, the “data set” is separated from the actual “test case” which will test the functionality of the web application. It can also be structured in a way wherein, the test cases which need to be executed are called (invoked) from an external application (like a .csv).

There are a number of frameworks out there, but 3 commonly used Selenium framework (s) are:

* [Data Driven framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks" \l "DataDrivenFramework)
* [Keyword Driven framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks" \l "KeywordDrivenFramework)
* [Hybrid framework](https://www.edureka.co/blog/selenium-framework-data-keyword-hybrid-frameworks" \l "HybridFramework)

## ****Why do we need a Selenium framework?****

Without a framework in place, there will be one test case which will comprise the entire test functionality. The scary part is, this single test case has the capability to rise up to a million lines of code. So its pretty obvious that a test case so huge will be tough to read. Even if you want to modify any functionality later, then you will have a tough time modifying the code.

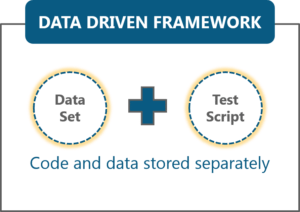
Since the implementation of a framework, will result in smaller but multiple code pieces, there are various benefits.

### ****Benefits of Selenium framework****

* Increased code reusage
* Improved code readability
* Higher portability
* Reduced script maintenance

## ****Data Driven Framework****

A Data Driven framework in Selenium is the technique of separating the “data set” from the actual “test case” (code). This framework completely depends on the input test data. The test data is fed from external sources such as an excel file, .CSV file or any database.

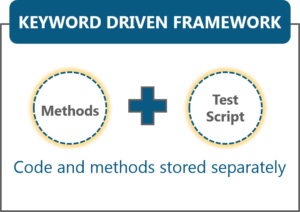


Since the test case is separated from the data set, we can easily modify the test case of a particular functionality without making wholesale changes to your code. For example, if you want to modify the code for login functionality, then you can modify just that instead of having to also modify any other dependent portion in the same code.

Besides this, you can also easily control how much data needs to be tested. You can easily increase the number of test parameters by adding more username and password fields to the excel file (or other sources).

## ****Keyword Driven Framework****

Keyword Driven framework is a technique in which all the operations & instructions to be performed are written separately from the actual test case. The similarity it has with Data Driven framework is that, the operations to be performed is again stored in an external file like Excel sheet.



The operations I’m talking about is nothing but the methods that need to be executed as part of a test case. The benefit with Keyword Driven framework is that you can easily control the functionalities you want to test. You can specify the methods which test the functionality of the application in the excel file. Thus, only those method names which are specified in the excel will be tested.

For example, for logging into the web application, we can write multiple methods in the main test case, in which each test case will test certain functionality. For instantiating the browser driver there could be one method, for finding the username & password fields, there could be methods, for navigating to a web page there could be another method, etc.