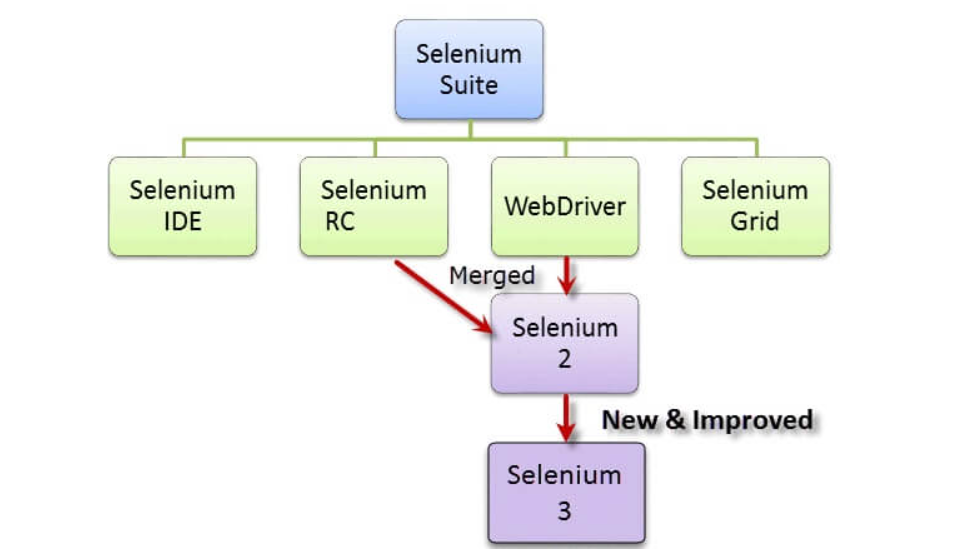
SELENIUM:

Selenium is an automated tool used for testing web applications.

Selenium supports a variety of programming languages through the use of drivers specific to each language.Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby.Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.



| **Tool** | **Why Choose?** |
| --- | --- |
| **Selenium IDE** | * To learn about concepts on automated testing and Selenium, including: * Selenese commands such as type, open, clickAndWait, assert, verify, etc. * Locators such as id, name, xpath, css selector, etc. * Executing customized JavaScript code using runScript * Exporting test cases in various formats. * To create tests with little or no prior knowledge in programming. * To create simple test cases and test suites that you can export later to RC or WebDriver. * To test a web application against Firefox and Chrome only. |
| **Selenium RC** | * To design a test using a more expressive language than Selenese * To run your test against different browsers (except HtmlUnit) on different operating systems. * To deploy your tests across multiple environments using Selenium Grid. * To test your application against a new browser that supports JavaScript. * To test web applications with complex AJAX-based scenarios. |
| **WebDriver** | * To use a certain programming language in designing your test case. * To test applications that are rich in AJAX-based functionalities. * To execute tests on the HtmlUnit browser. * To create customized test results. |
| **Selenium Grid** | * To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. * To run a huge test suite, that needs to complete in the soonest time possible. |

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser’s actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. **It controls the browser by directly communicating with it.**

**What is Selenium Webdriver?**

**Selenium Webdriver is an open-source collection of APIs which is used for testing web applications. The Selenium Webdriver tool is used for automating web application testing to verify that it works as expected or not. It mainly supports browsers like Firefox, Chrome, Safari and Internet Explorer. It also permits you to execute cross-browser testing.**

**WebDriver also enables you to use a programming language in creating your test scripts (not possible in Selenium IDE).**

**You can now use conditional operations like if-then-else or switch-case. You can also perform looping like do-while.**

To work with Selenium web driver the below are needed:

1. Eclipse
2. Java
3. Selenium Web driver APIs

[Downloads | Selenium](https://www.selenium.dev/downloads/)

To import all the jar files onto the project build path.

Simple example:

Example1.java

Public class Example1{

Psvm(String args[]){

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

String baseUrl = "http://www.facebook.com";

String tagName = "";

driver.get(baseUrl);

tagName = driver.findElement(By.id("email")).getTagName();

System.out.println(tagName);

driver.close();

System.exit(0);

}

}

What are Locators?

Locator is a command that tells Selenium IDE which GUI elements ( say Text Box, Buttons, Check Boxes etc) its needs to operate on.

|  |  |  |
| --- | --- | --- |
| Method | Syntax | Description |
| By ID | driver.findElement(By.id (<element ID>)) | Locates an element using the ID attribute |
| By name | driver.findElement(By.name (<element name>)) | Locates an element using the Name attribute |
| By class name | driver.findElement(By.className (<element class>)) | Locates an element using the Class attribute |
| By tag name | driver.findElement(By.tagName (<htmltagname>)) | Locates an element using the HTML tag |
| By link text | driver.findElement(By.linkText (<linktext>)) | Locates a link using link text |
| By partial link text | driver.findElement(By.partialLinkText (<linktext>)) | Locates a link using the link's partial text |
| By CSS | driver.findElement(By.cssSelector (<css selector>)) | Locates an element using the CSS selector |
| By XPath | driver.findElement(By.xpath (<xpath>)) | Locates an element using XPath query |

**Try this also:**

**Selenium IDE installation:**

Selenium IDE is available only as Firefox and Chrome plug-in, we assume that you have already installed Mozilla Firefox browser in your system.

<https://addons.mozilla.org/en-us/firefox/addon/selenium-ide/I>

**EXAMPLES:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {

// TODO Auto-generated method stub

System.setProperty("webdriver.chrome.driver", "D:\\ chrome\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.manage().window().maximize();

driver.get("http://demo.guru99.com/test/ajax.html");

// Find the radio button for “No” using its ID and click on it

driver.findElement(By.id("no")).click();

//Click on Check Button

driver.findElement(By.id("buttoncheck")).click();

}

}

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class NameDemo {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "X://chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://demo.guru99.com/test/ajax.html");

List<WebElement> elements = driver.findElements(By.name("name"));

System.out.println("Number of elements:" +elements.size());

for (int i=0; i<elements.size();i++){

System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));

}

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class Form {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

String baseUrl = "http://demo.guru99.com/test/login.html";

driver.get(baseUrl);

// Get the WebElement corresponding to the Email Address(TextField)

WebElement email = driver.findElement(By.id("email"));

// Get the WebElement corresponding to the Password Field

WebElement password = driver.findElement(By.name("passwd"));

email.sendKeys("abcd@gmail.com");

password.sendKeys("abcdefghlkjl");

System.out.println("Text Field Set");

// Deleting values in the text box

email.clear();

password.clear();

System.out.println("Text Field Cleared");

// Find the submit button

WebElement login = driver.findElement(By.id("SubmitLogin"));

// Using click method to submit form

email.sendKeys("abcd@gmail.com");

password.sendKeys("abcdefghlkjl");

login.click();

System.out.println("Login Done with Click");

//using submit method to submit the form. Submit used on password field

driver.get(baseUrl);

driver.findElement(By.id("email")).sendKeys("abcd@gmail.com");

driver.findElement(By.name("passwd")).sendKeys("abcdefghlkjl");

driver.findElement(By.id("SubmitLogin")).submit();

System.out.println("Login Done with Submit");

//driver.close();

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class Form {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://demo.guru99.com/test/radio.html");

WebElement radio1 = driver.findElement(By.id("vfb-7-1"));

WebElement radio2 = driver.findElement(By.id("vfb-7-2"));

//Radio Button1 is selected

radio1.click();

System.out.println("Radio Button Option 1 Selected");

//Radio Button1 is de-selected and Radio Button2 is selected

radio2.click();

System.out.println("Radio Button Option 2 Selected");

// Selecting CheckBox

WebElement option1 = driver.findElement(By.id("vfb-6-0"));

// This will Toggle the Check box

option1.click();

// Check whether the Check box is toggled on

if (option1.isSelected()) {

System.out.println("Checkbox is Toggled On");

} else {

System.out.println("Checkbox is Toggled Off");

}

//Selecting Checkbox and using isSelected Method

driver.get("http://demo.guru99.com/test/facebook.html");

WebElement chkFBPersist = driver.findElement(By.id("persist\_box"));

for (int i=0; i<2; i++) {

chkFBPersist.click ();

System.out.println("Facebook Persists Checkbox Status is - "+chkFBPersist.isSelected());

}

//driver.close();

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class MyClass {

public static void main(String[] args) {

String baseUrl = "https://www.facebook.com/login/identify?ctx=recover";

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

//click on the "Facebook" logo on the upper left portion

driver.findElement(By.cssSelector("a[title=\"Go to Facebook home\"]")).click();

//verify that we are now back on Facebook's homepage

if (driver.getTitle().equals("Facebook - log in or sign up")) {

System.out.println("We are back at Facebook's homepage");

} else {

System.out.println("We are NOT in Facebook's homepage");

}

driver.close();

}

}

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.By;

public class accessDropDown {

public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

String baseURL = "http://demo.guru99.com/test/newtours/register.php";

WebDriver driver = new FirefoxDriver();

driver.get(baseURL);

Select drpCountry = new Select(driver.findElement(By.name("country")));

drpCountry.selectByVisibleText("ANTARCTICA");

//Selecting Items in a Multiple SELECT elements

driver.get("http://jsbin.com/osebed/2");

Select fruits = new Select(driver.findElement(By.id("fruits")));

fruits.selectByVisibleText("Banana");

fruits.selectByIndex(1);

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class MyClass {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/link.html";

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

driver.findElement(By.linkText("click here")).click();

System.out.println("title of page is: " + driver.getTitle());

driver.quit();

}

}

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

public class PG7 {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/newtours/";

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement link\_Home = driver.findElement(By.linkText("Home"));

WebElement td\_Home = driver

.findElement(By

.xpath("//html/body/div"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr"));

Actions builder = new Actions(driver);

Action mouseOverHome = builder

.moveToElement(link\_Home)

.build();

String bgColor = td\_Home.getCssValue("background-color");

System.out.println("Before hover: " + bgColor);

mouseOverHome.perform();

bgColor = td\_Home.getCssValue("background-color");

System.out.println("After hover: " + bgColor);

driver.close();

}

}

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

public class PG9 {

public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

String baseUrl = "http://demo.guru99.com/test/upload/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement uploadElement = driver.findElement(By.id("uploadfile\_0"));

// enter the file path onto the file-selection input field

uploadElement.sendKeys("C:\\newhtml.html");

// check the "I accept the terms of service" check box

driver.findElement(By.id("terms")).click();

// click the "UploadFile" button

driver.findElement(By.name("send")).click();

}

}

## What is XPath in Selenium?

**XPath in Selenium** is an XML path used for navigation through the HTML structure of the page. It is a syntax or language for finding any element on a web page using XML path expression. XPath can be used for both HTML and XML documents to find the location of any element on a webpage using HTML DOM structure.

Absolute Xpath and Relative Xpath and all details:

[Complete Guide For Using XPath In Selenium With Examples (lambdatest.com)](https://www.lambdatest.com/blog/complete-guide-for-using-xpath-in-selenium-with-examples/)

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.NoAlertPresentException;

import org.openqa.selenium.Alert;

public class AlertDemo {

public static void main(String[] args) throws NoAlertPresentException,InterruptedException {

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

// Alert Message handling

driver.get("http://demo.guru99.com/test/delete\_customer.php");

driver.findElement(By.name("cusid")).sendKeys("53920");

driver.findElement(By.name("submit")).submit();

// Switching to Alert

Alert alert = driver.switchTo().alert();

// Capturing alert message.

String alertMessage= driver.switchTo().alert().getText();

// Displaying alert message

System.out.println(alertMessage);

Thread.sleep(5000);

// Accepting alert

alert.accept();

}

}

Dynamic web table:

import java.text.ParseException;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Noofrowsandcols {

public static void main(String[] args) throws ParseException {

WebDriver wd;

System.setProperty("webdriver.chrome.driver","G://chromedriver.exe");

wd= new ChromeDriver();

wd.get("http://demo.guru99.com/test/web-table-element.php");

//No.of Columns

List col = wd.findElements(By.xpath(".//\*[@id=\"leftcontainer\"]/table/thead/tr/th"));

System.out.println("No of cols are : " +col.size());

//No.of rows

List rows = wd.findElements(By.xpath(".//\*[@id='leftcontainer']/table/tbody/tr/td[1]"));

System.out.println("No of rows are : " + rows.size());

wd.close();

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class ToolTip {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/social-icon.html";

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

String expectedTooltip = "Github";

// Find the Github icon at the top right of the header

WebElement github = driver.findElement(By.xpath(".//\*[@class='soc-ico show-round']/a[4]"));

//get the value of the "title" attribute of the github icon

String actualTooltip = github.getAttribute("title");

//Assert the tooltip's value is as expected

System.out.println("Actual Title of Tool Tip"+actualTooltip);

if(actualTooltip.equals(expectedTooltip)) {

System.out.println("Test Case Passed");

}

driver.close();

}

}

**How to identify iframes in the page using Selenium webdriver:**

public class SwitchToFrame\_ID {

public static void main(String[] args) {

WebDriver driver = new FirefoxDriver(); //navigates to the Browser

driver.get("http://demo.guru99.com/test/guru99home/");

// navigates to the page consisting an iframe

driver.manage().window().maximize();

driver.switchTo().frame("a077aa5e"); //switching the frame by ID

System.out.println("\*\*\*\*\*\*\*\*We are switch to the iframe\*\*\*\*\*\*\*");

driver.findElement(By.xpath("html/body/a/img")).click();

//Clicks the iframe

System.out.println("\*\*\*\*\*\*\*\*\*We are done\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

Nested frames:

public class FramesInsideFrames {

public static void main(String[] args) {

WebDriver driver=new FirefoxDriver();

driver.get("Url");

driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(2, TimeUnit.SECONDS);

int size = driver.findElements(By.tagName("iframe")).size();

System.out.println("Total Frames --" + size);

// prints the total number of frames

driver.switchTo().frame(0); // Switching the Outer Frame

System.out.println (driver.findElement(By.xpath("xpath of the outer element ")).getText());

//Printing the text in outer frame

size = driver.findElements(By.tagName("iframe")).size();

// prints the total number of frames inside outer frame

System.out.println("Total Frames --" + size);

driver.switchTo().frame(0); // Switching to innerframe

System.out.println(driver.findElement(By.xpath("xpath of the inner element ")).getText());

//Printing the text in inner frame

driver.switchTo().defaultContent();

}

}

**Sleep(), wait() in selenium:**

[Thread.sleep() Wait in Selenium WebDriver – Selenium Tutorial (automate-apps.com)](http://automate-apps.com/thread-sleep-wait-in-selenium-webdriver/)

**cssSelector and iframes**:

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumDemo {

public static WebDriver driver;

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Launching the chrome driver ");

// Set the chrome driver exe file path

System.setProperty("webdriver.chrome.driver","E:\\selenium\_sumit\\chromedriver\_win32\_2.36\\chromedriver.exe");

// Instantiate the chrome driver

driver = new ChromeDriver();

// wait time

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// set the browser URL in get() to load the webpage

driver.get("https://skptricks.github.io/learncoding/selenium-demo/login%20registration%20page/Register.html");

// locating an elements by its ID

driver.findElement(By.cssSelector("input#regUsername")).sendKeys("sumit");

driver.findElement(By.cssSelector("input#regEmail")).sendKeys("sumit@gmail.com");

driver.findElement(By.cssSelector("input#regPassword")).sendKeys("1234567890");

}

}

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumDemo {

public static WebDriver driver;

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("Launching the chrome driver ");

// Set the chrome driver exe file path

System.setProperty("webdriver.chrome.driver","E:\\selenium\_sumit\\chromedriver\_win32\_2.36\\chromedriver.exe");

// Instantiate the chrome driver

driver = new ChromeDriver();

// wait time

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// set the browser URL in get() to load the webpage

driver.get("https://skptricks.github.io/learncoding/selenium-demo/login%20registration%20page/Register.html");

// locating an elements using css selector

driver.findElement(By.cssSelector("input.button.style1.style2")).click();

}

}

public class FramesInWebdriver {

public static void main(String[] args) throws Exception {

// set the geckodriver.exe property

System.setProperty("webdriver.gecko.driver", "C:/PATH/geckodriver.exe");

// open firefox

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

// open webpage

driver.get("https://chercher.tech/practice/frames");

// store the text value

String textValue = driver.findElement(By.xpath("//label/span")).getText();

// switch to frame1

driver.switchTo().frame("frame1");

// set the value of the textbar to the value stored

driver.findElement(By.xpath("//input[@type='text']")).sendKeys(textValue);

}

}

public class NestedFrame {

public static void main(String[] args) throws Exception {

// set the geckodriver.exe property

System.setProperty("webdriver.gecko.driver", "C:/PATH/geckodriver.exe");

// open firefox

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

// open webpage

driver.get("https://chercher.tech/practice/frames");

//find the frame1 and store it in webelement

WebElement frame1 = driver.findElement(By.id("frame1"));

// switch to frame1

driver.switchTo().frame(frame1);

// find the frame 3

WebElement frame3 = driver.findElement(By.xpath("//iframe[@id='frame3']"));

// switch to frame 3

driver.switchTo().frame(frame3);

// find the checkbox

WebElement checkbox = driver.findElement(By.xpath("//input[@type='checkbox']"));

// if checkbox is not selected then click the checkbox

if(! checkbox.isSelected()){

checkbox.click();

}

}

}

Cross browser testing:

[Cross Browser Testing using Selenium WebDriver (guru99.com)](https://www.guru99.com/cross-browser-testing-using-selenium.html#:~:text=%20%20%201%20Cross%20browser%20Testing%20is,case%2C%20we%20can%20create%20WebDriver%20reference...%20More%20)

Wait():

[Waits in Selenium (lambdatest.com)](https://www.lambdatest.com/blog/types-of-waits-in-selenium/)

Sleep():

[Using Thread.sleep() in Selenium WebDriver - Make Selenium Easy](http://makeseleniumeasy.com/2017/05/28/part-1-waits-in-selenium-webdriver-thread-sleep/)

POM Design Pattern:

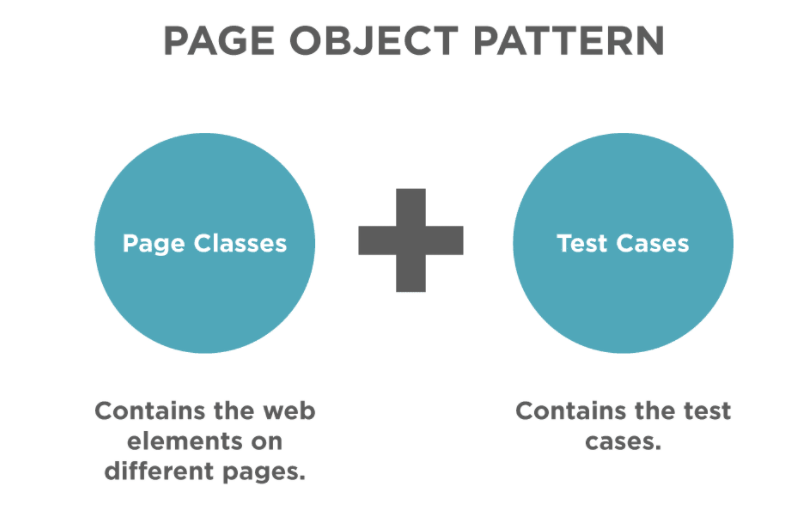
Page Object Model:

**Page Object Model (POM)** is a design pattern, popularly used in test automation that creates Object Repository for web UI elements.

Under this model, for each web page in the application, there should be a corresponding Page Class. This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements.

## What is Page Factory in Selenium?

**Page Factory in Selenium** is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized. It is used for initialization of Page objects or to instantiate the Page object itself. It is also used to initialize Page class elements without using “FindElement/s.”



Good one on PageFactory, try to understand several annotations available:

[Page Factory in Selenium WebDriver - ArtOfTesting](https://artoftesting.com/page-factory-in-selenium)

[Page Factory in Selenium Webdriver and Page Object Model (toolsqa.com)](https://www.toolsqa.com/selenium-webdriver/page-factory-in-selenium/)

TestNG:

TestNG is one of the most widely used open source testing framework used in automation testing suite.

It is an open source automated TestNG framework. In TestNG, NG stands for "**Next Generation**".

* In TestNG, annotations are easier to understand than Junit.
* It produces the HTML reports for implementation.
* It also generates the Logs.
* In TestNG, there is no constraint available such as @beforeclass and @afterclass which is present in Junit.
* 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.

**TestNG installation:**

In Eclipse, open eclipse marketplace and type ‘TestNG’ and install it.

Create TestNG type of class.

**TestNG with Selenium:**

[TestNG Annotations, Framework, Examples in Selenium – Rahul Shetty Academy Blog](https://rahulshettyacademy.com/blog/index.php/2021/05/17/testng-annotations-framework-examples-in-selenium/#t-1635321801707)

Add respective browser exe file accordingly:

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class TestNGClass {

WebDriver driver = new FirefoxDriver();

@BeforeTest

public void launchapp() {

// Puts an Implicit wait, Will wait for 10 seconds before throwing exception

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// Launch website

driver.navigate().to("http://www.calculator.net");

driver.manage().window().maximize();

}

@Test

public void calculatepercent() {

// Click on Math Calculators

driver.findElement(By.xpath(".//\*[@id='menu']/div[3]/a")).click();

// Click on Percent Calculators

driver.findElement(By.xpath(".//\*[@id='menu']/div[4]/div[3]/a")).click();

// Enter value 10 in the first number of the percent Calculator

driver.findElement(By.id("cpar1")).sendKeys("10");

// Enter value 50 in the second number of the percent Calculator

driver.findElement(By.id("cpar2")).sendKeys("50");

// Click Calculate Button

driver.findElement(By.xpath(".//\*[@id='content']/table/tbody/tr/td[2]/input")).click();

// Get the Result Text based on its xpath

String result =

driver.findElement(By.xpath(".//\*[@id='content']/p[2]/span/font/b")).getText();

// Print a Log In message to the screen

System.out.println(" The Result is " + result);

if(result.equals("5")) {

System.out.println(" The Result is Pass");

} else {

System.out.println(" The Result is Fail");

}

}

@AfterTest

public void terminatetest() {

driver.close();

}

}

**How to prioritize the testcases:**

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.\*;

@Test

public class firsttestngfile {

public String baseUrl = "http://demo.guru99.com/test/newtours/";

String driverPath = "C:\\geckodriver.exe";

public WebDriver driver;

public String expected = null;

public String actual = null;

@BeforeTest

public void launchBrowser() {

System.out.println("launching firefox browser");

System.setProperty("webdriver.gecko.driver", driverPath);

driver= new FirefoxDriver();

driver.get(baseUrl);

}

@BeforeMethod

public void verifyHomepageTitle() {

String expectedTitle = "Welcome: Mercury Tours";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@Test(priority = 0)

public void register(){

driver.findElement(By.linkText("REGISTER")).click() ;

expected = "Register: Mercury Tours";

actual = driver.getTitle();

Assert.assertEquals(actual, expected);

}

@Test(priority = 1)

public void support() {

driver.findElement(By.linkText("SUPPORT")).click() ;

expected = "Under Construction: Mercury Tours";

actual = driver.getTitle();

Assert.assertEquals(actual, expected);

}

@AfterMethod

public void goBackToHomepage ( ) {

driver.findElement(By.linkText("Home")).click() ;

}

@AfterTest

public void terminateBrowser(){

driver.close();

}

}

**Selenium with Jasmine:**

Npm init

npm install -g jasmine

npm install --save chromedriver

npm install --save selenium-webdriver

Jasmine.Async is an add-on library for Jasmine that provides additional functionality to do asynchronous testing. Modeled after Mocha’s async test support, it brings the done() function to the Jasmine unit testing environment.

Thus, the Jasmine.Async library was born by borrowing Mocha’s use of the done() when the asynchronous code has completed.

**Mocha is another test framework.**

[Using the done() Method in Your Jasmine-driven Asynchronous JavaScript Tests | HTML Goodies](https://www.htmlgoodies.com/javascript/using-the-done-method-in-your-jasmine-driven-asynchronous-javascript-tests/)

[jasmine-await - npm (npmjs.com)](https://www.npmjs.com/package/jasmine-await)

“RECALL WHAT WE HAD LEARNT ABOUT PROMISE”.

|  |
| --- |
|  |
|  | const {Builder, By, Key, until} = require('selenium-webdriver'); |
|  |  |
|  | // You can use a remote Selenium Hub, but we are not doing that here |
|  | require('chromedriver'); |
|  | const driver = new Builder() |
|  | .forBrowser('chrome') |
|  | .build(); |
|  |  |
|  | // Setting variables for our testcase |
|  | const baseUrl = 'https://accounts.lambdatest.com/login' |
|  |  |
|  | // function to check for login elements and do login |
|  | var loginToLamdbatest = async function() { |
|  |  |
|  | let loginButton = By.xpath('//button'); |
|  |  |
|  | // navigate to the login page |
|  | await driver.get(baseUrl); |
|  |  |
|  | // wait for login page to be loaded |
|  | await driver.wait(until.elementLocated(loginButton), 10 \* 1000); |
|  | console.log('Login screen loaded.') |
|  | } |
|  |  |
|  | //to set jasmine default timeout |
|  | jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 20 \* 1000; |
|  |  |
|  | // Start to write the first test case |
|  | describe("Selenium test case for login page", function() { |
|  | it("verify page elements", async function() { |
|  | console.log('<----- Starting to execute test case ----->'); |
|  |  |
|  | //to do login |
|  | await loginToLamdbatest(); |
|  |  |
|  | var welcomeMessage = By.xpath('//\*[@class="form\_title"]'); |
|  |  |
|  | //verify welcome message on login page |
|  | expect(await driver.findElement(welcomeMessage).getText()).toBe('Welcome Back !'); |
|  |  |
|  | //to quit the web driver at end of test case execution |
|  | await driver.quit(); |
|  |  |
|  | console.log('<----- Test case execution completed ----->'); |
|  | }); |
|  | }); |

var selenium = require('selenium-webdriver');

describe('Selenium Tutorial', function() {

// Open the TECH.insight website in the browser before each test is run

beforeEach(function(done) {

this.driver = new selenium.Builder().

withCapabilities(selenium.Capabilities.chrome()).

build();

this.driver.get('http://www.techinsight.io/').then(done);

});

// Close the website after each test is run (so that it is opened fresh each time)

afterEach(function(done) {

this.driver.quit().then(done);

});

// Test to ensure we are on the home page by checking the <body> tag id attribute

it('Should be on the home page', function(done) {

var element = this.driver.findElement(selenium.By.tagName('body'));

element.getAttribute('id').then(function(id) {

expect(id).toBe('home');

done();

});

});

// Test the navigation bar by clicking on the 'REVIEW' link and checking the URL changes to '/review'

it('Has a working nav', function(done) {

var element = this.driver.findElement(selenium.By.linkText('REVIEW'));

element.click();

this.driver.getCurrentUrl().then(function(value) {

expect(value).toContain('/review');

done();

});

});

});

**Zephyr:**

Zephyr is a test Management tool used as a plugin with JIRA tool:

**JIRA tool**:

JIRA is an Incident Management Tool used for Project Management, Bug Tracking, Issue Tracking and Workflow. JIRA is based on the following three concepts – Project, Issue and Workflow.

* JIRA is used in Bugs, Issues and Change Request Tracking.
* JIRA can be used in Help desk, Support and Customer Services to create tickets and track the resolution and status of the created tickets.
* JIRA is useful in Project Management, Task Tracking and Requirement Management.
* JIRA is very useful in Workflow and Process management.

Since earlier JIRA was just used as a bug tracking tool, Zephyr was bought in as a plugin to support the test management like other tools such as QC (Quality Center) etc.

Using Zephyr we can create test cases, track them and also manage similar to that of a defect life cycle.

Installation:

[Sign up - Try Atlassian Cloud | Atlassian](https://www.atlassian.com/try/cloud/signup?bundle=jira-software&edition=free)

Install JIRA and inside JIRA look for Apps and search for Zephyr for JIRA to be plugged in.

[Zephyr for JIRA Tutorial: Test Management Tool (guru99.com)](https://www.guru99.com/zephyr-agile-jira.html#3)

**Mockito:**

Mockito is a mocking framework. It is a Java-based library used to create simple and basic test APIs for performing unit testing of Java applications. It can also be used with other frameworks such as **JUnit** and **TestNG**.

The static import feature of Java 5 facilitate the java programmer to access any static member of a class directly. There is no need to qualify it by the class name.

It internally uses Java Reflection API and allows to create objects of a service. A mock object returns a dummy data and avoids external dependencies.

[New Message! (eyebuzzgallery.com)](https://eyebuzzgallery.com/what-is-the-use-of-when-in-mockito)

mock() -> It is used to create mock objects of a given class or interface.

When()->  It should be used when we want to mock to return specific values when particular methods are called. In simple terms, "**When** the XYZ() method is called, **then** return ABC."

[Methods of Mockito - Javatpoint](https://www.javatpoint.com/methods-of-mockito)

[Mockito Annotations - Javatpoint](https://www.javatpoint.com/mockito-annotations)

**Examples:**

import java.util.List;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

public class MockitoMockMethodExample {

@SuppressWarnings("unchecked")

@Test

public void test() {

// using Mockito.mock() method

List<String> mockList = mock(List.class);

when(mockList.size()).thenReturn(5);

assertTrue(mockList.size()==5);

}

}

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class MockitoMockAnnotationExample {

@Mock

List<String> mockList;

@BeforeEach

public void setup() {

//if we don't call below, we will get NullPointerException

MockitoAnnotations.initMocks(this);

}

@SuppressWarnings("unchecked")

@Test

public void test() {

when(mockList.get(0)).thenReturn("Sample");

assertEquals("Sample", mockList.get(0));

}

}

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class MockitoInjectMockAnnotationExample {

@Mock

List<String> mockList;

//@InjectMock creates an instance of the class and

//injects the mocks that are marked with the annotations @Mock into it.

@InjectMocks

Fruits mockFruits;

@BeforeEach

public void setup() {

//if we don't call below, we will get NullPointerException

MockitoAnnotations.initMocks(this);

}

@SuppressWarnings("unchecked")

@Test

public void test() {

when(mockList.get(0)).thenReturn("Apple");

when(mockList.size()).thenReturn(1);

assertEquals("Apple", mockList.get(0));

assertEquals(1, mockList.size());

//mockFruits names is using mockList, below asserts confirm it

assertEquals("Apple", mockFruits.getNames().get(0));

assertEquals(1, mockFruits.getNames().size());

mockList.add(1, "Mango");

//below will print null because mockList.get(1) is not stubbed

System.out.println(mockList.get(1));

}

}

class Fruits{

private List<String> names;

public List<String> getNames() {

return names;

}

public void setNames(List<String> names) {

this.names = names;

}

}

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.spy;

import static org.mockito.Mockito.when;

import java.util.ArrayList;

import java.util.List;

import org.junit.jupiter.api.Test;

public class MockitoSpyMethodExample {

@Test

public void test() {

List<String> list = new ArrayList<>();

List<String> spyOnList = spy(list);

when(spyOnList.size()).thenReturn(10);

assertEquals(10, spyOnList.size());

//calling real methods since below methods are not stubbed

spyOnList.add("XXX");

spyOnList.add("YYY");

assertEquals("XXX", spyOnList.get(0));

assertEquals("YYY", spyOnList.get(1));

}

}

import static org.mockito.Mockito.verify;

import java.util.List;

import org.mockito.Mockito;

import org.mockito.internal.verification.VerificationModeFactory;

import org.testng.annotations.Test;

public class TestNGMockitoVerify {

@Test

public void test\_mockito\_verify() {

@SuppressWarnings("unchecked")

List<String> mockedList = Mockito.mock(List.class);

mockedList.add("first-element");

mockedList.add("second-element");

mockedList.add("third-element");

mockedList.add("third-element");

mockedList.clear();

verify(mockedList).add("first-element");

verify(mockedList).add("second-element");

verify(mockedList, VerificationModeFactory.times(2)).add("third-element");

verify(mockedList).clear();

}

}