1. **HOW TO CHECK IF ANY BROKEN LINK AVAILABLE IN THE PAGE?**

public void userCheckIfAnyBrokenLinkAvailableInThePage() {  
 WebDriverManager.*chromedriver*().setup();  
 WebDriver driver = new ChromeDriver();  
 driver.get("http://google.com/");  
 List<WebElement> allLinks = driver.findElements(By.*tagName*("a"));  
 for (WebElement element : allLinks){  
 String url = element.getAttribute("href");  
 try {  
 HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(url).openConnection();  
 httpURLConnection.setRequestMethod("HEAD");  
 httpURLConnection.connect();  
 int response = httpURLConnection.getResponseCode();  
 if (response>=400){  
 System.*out*.println("Broken Link: "+url);  
 }else {  
 System.*out*.println("Valid Link: "+url);  
 }  
 System.*out*.println(response);  
 driver.quit();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. **WHAT IS PAGE OBJECT MODEL IN SELENIUM?**

POM is a design pattern, which commonly used in Selenium for Automating the Test Cases. This design pattern can be used with any kind of framework like keyword-driven, Data-driven, hybrid framework, etc. The Page object is a java class, which acts as an interface for the page of our Application under test. Page class contains web elements and methods to interact with web elements. While automating the test cases, we create the object of these Page Classes and interact with web elements by calling the methods of these classes. In this Design Pattern, web pages are represented by a corresponding java 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.

We are defining all locator variable as a class member with @FindBy annotation. Those are actually Webelement variable. Those can be in different format like Id, name, class, xpath and so on. Here you can see the webelement declaration syntax using @findby annotation. Also, we are defining the actual java method in the page file where the scripts are interacting with web application.

**Advantages of POM model:**

Reusability: We can reuse the page class if required in different test cases which means we don’t need to write code for identifying the web elements and methods to interact with them for every test case.

Maintainability: As test case and page class are different from each other which means we can easily update the code if any new web element is added or existing one updated. We just need to go to that page class where changes needs to be done and update the locator and that will reflect in our entire automation project.

1. **WHAT IS PAGE FACTORY IN SELENIUM?**

Page Factory is a class provided by Selenium WebDriver to support Page Object model Design patterns. In Page Factory, we are using @FindBy annotation to defining the webelements. The initElements() is a static method of pageFactory class which is used to initialize all web elements under page class.

|  |  |
| --- | --- |
| **PAGE OBJECT MODEL** | **PAGE FACTORY** |
| Finding web elements using By | Finding web elements using @FindBy |
| POM does not provide lazy initialization | Page Factory does provide lazy initialization |
| Page Object Model is a design pattern | PageFactory is a class which provides implementation of Page Object Model design pattern |
| In POM, one needs to initialize every page object individually | In PageFactory, all page objects are initialized by using the initElements() method |

1. **IMPLICIT, EXPLICIT & FLUENT WAIT IN SELENIUM WEBDRIVER**

* The Implicit Wait in Selenium is used to tell the web driver to wait for a certain amount of time before it throws a "No Such Element Exception". The default setting is 0. Once we set the time, the web driver will wait for the element for that time before throwing an exception.

In the below example we have declared an implicit wait with the time frame of 10 seconds. It means that if the element is not located on the web page within that time frame, it will throw an exception.

*driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);*

* The Explicit Wait in Selenium is used to tell the Web Driver to wait for certain conditions or maximum time exceeded before throwing "ElementNotVisibleException" exception. It is an intelligent kind of wait, but it can be applied only for specified elements. It gives better options than implicit wait as it waits for dynamically loaded elements.

WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( "/html/ ")))

elementToBeClickable()

elementToBeSelected()

visibilityOfElementLocated()

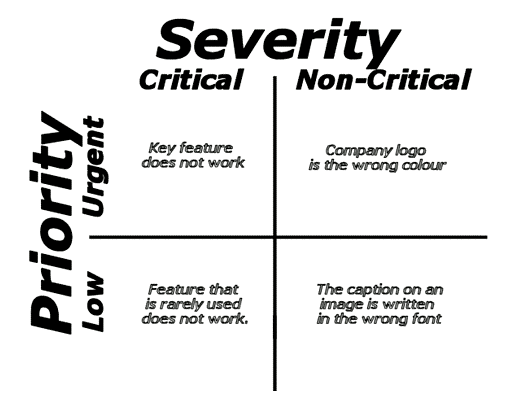
presenceOfAllElementsLocatedBy()

* The Fluent Wait in Selenium is used to define maximum time for the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException" exception. It checks for the web element at regular intervals until the object is found or timeout happens.

1. **PRIORITY VS SEVERITY: KEY DIFFERENCE**

| PRIORITY | SEVERITY |
| --- | --- |
| Defect Priority has defined the order in which the developer should resolve a defect | Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| Priority is categorized into three types  Low  Medium  High | Severity is categorized into five types  Critical  Major  Moderate  Minor  Cosmetic |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |

1. **EXAMPLE OF DEFECT SEVERITY AND PRIORITY**



1. **SOFTWARE TESTING CONCEPTS:**

**What is Waterfall methodology?**

Waterfall Model methodology which is also known as Liner Sequential Life Cycle Model. Waterfall Model is followed in the sequential order, and so project development team only moves to next phase of development or testing if the previous step completed successfully.

**What is Agile methodology?**

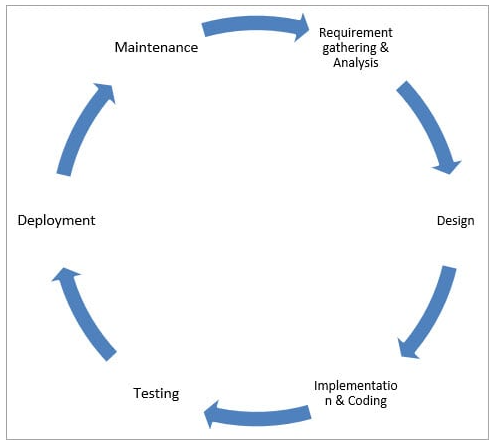
Agile methodology is a practice that helps continuous iteration of development and testing in the software development process. In this model, development and testing activities are concurrent, unlike the Waterfall model.

**What is a Sprint in Agile?**

Using Agile project management methodologies, projects are broken down into sprints or iterations. These are short, repeatable phases, typically one to four weeks in length. Each sprint should result in a draft, prototype or workable version of the final project deliverable.

**WHAT IS SDLC?**

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Life Cycle.



1. ***Requirement Gathering and Analysis***

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

1. ***Design***

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

1. ***Implementation or Coding***

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

1. ***Testing***

Testing starts once, the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and if any defects found in the application, that are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer’s expectation. Testers refer SRS document to make sure that the software is as per the customer’s standard.

1. ***Deployment***

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

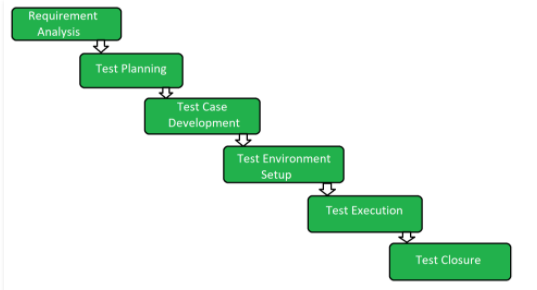
In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

1. ***Maintenance***

After the deployment of a product on the production environment, maintenance of the product starts i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers as well as testers.

**WHAT IS SOFTWARE TESTING LIFE CYCLE (STLC)?**

Software Testing Life Cycle (STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met.



1. **Requirement Analysis:**

Requirement Analysis is the first step of Software Testing Life Cycle (STLC). During this phase, the test team studies and analyses the requirements from a testing perspective with the help of BRS (Business Requirement Specification) document.

1. **Test Planning:**

In this phase, typically Test Manager/Test Lead are involved to determine the effort and cost estimates for the entire project. Preparation of the Test Plan will be done based on the requirement analysis. Activities like resource planning, determining roles and responsibilities, tool selection (for automation), training requirements, etc., carried out in this phase.

1. **Test Case Development:**

The test team starts with test case development activity in this phase. Testers prepares test cases, test scripts (for automation), and test data. Once the test cases are ready then these test cases are reviewed by peer members or team lead.

1. **Test Environment Setup:**

This phase can be started in parallel with the Test design phase.

The test environment setup is done based on the hardware and software requirement list. In some cases, the test team may not be involved in this phase. The development team or customer provides the test environment.

1. **Test Execution:**

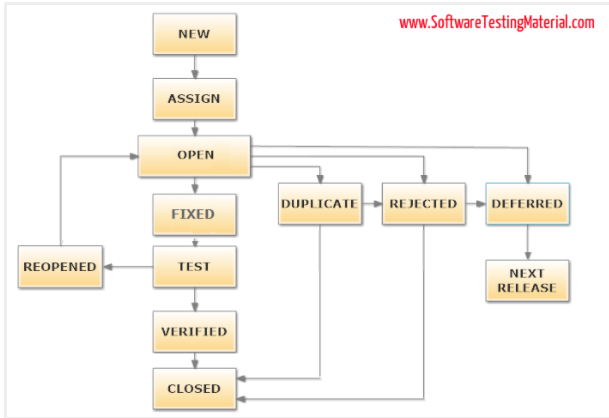
The test team starts executing the test cases based on the planned test cases. If a test case result is Pass/Fail then the same should be updated in the test cases. The defect report should be prepared for failed test cases and should be reported to the Development Team through a bug tracking tool for fixing the defects. Retesting will be performed once the defect was fixed.

1. **Test Closure:**

The final stage where we prepare Test Closure Report, Test Metrics.

**DEFECT LIFE CYCLE STATES**

The different states of a bug in the bug life cycle are as follows:



**New**

When a tester finds a new defect. He should provide a proper Defect document to the Development team to reproduce and fix the defect. In this state, the status of the defect posted by the tester is “New”

**Assigned**

Defects that are in the status of New will be approved (if valid) and assigned to the development team by Test Lead/Project Lead/Project Manager. Once the defect is assigned then the status of the bug changes to “Assigned”

**Open/Development**

The development team starts analysing and works on the defect fix

**Fixed**

When a developer makes the necessary code change and verifies the change, then the status of the bug will be changed as “Fixed” and the bug is passed to the testing team.

**Test**

If the status is “Test”, it means the defect is fixed and ready to do test whether it is fixed or not.

**Verified**

The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”

**Closed**

After verified the fix, if the bug is no longer exits then the status of the bug will be assigned as “Closed.”

**Reopen**

If the defect remains the same after the retest, then the tester posts the defect using the defect retesting document and changes the status to “Reopen”. Again the bug goes through the life cycle to be fixed.

**Duplicate**

If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate” by the development team.

**Deferred**

In some cases, the Project Manager/Lead may set the bug status as deferred.

* If the bug found during the end of the release and the bug is minor or not important to fix immediately.
* If the bug is not related to the current build.
* If it is expected to get fixed in the next release.
* The customer is thinking to change the requirement.
* In such cases the status will be changed as “deferred” and it will be fixed in the next release.

**Rejected**

If the system is working according to specifications and the bug is just due to some misinterpretation (such as referring to old requirements or extra features) then the Team lead or developers can mark such bugs as “Rejected”

Some other statuses are:

**Cannot be fixed**

Technology not supporting, Root of the product issue, Cost of fixing a bug is more

**Not Reproducible**

Platform mismatch, improper defect document, data mismatch, build mismatch, inconsistent defects

**Unit Testing:**

Unit testing is performed using the unit test cases that are designed and is done in the Low-level design phase. Unit testing is performed by the developer itself. It is performed on individual components which lead to early defect detection.

**Integration Testing:**

Integration testing is performed using integration test cases in High-level Design phase. Integration testing is the testing that is done on integrated modules. It is performed by testers.

**System Testing:**

System testing is performed in the System Design phase. In this phase, the complete system is tested i.e. the entire system functionality is tested.

**Acceptance Testing:**

Acceptance testing is associated with the Requirement Analysis phase and is done in the customer’s environment.

1. **EXPLAIN THE DIFFERENCE BETWEEN SINGLE AND DOUBLE SLASH IN X-PATH?**

**Single slash ‘/ ’**

* Single slash ( / ) start selection from the document node
* It allows you to create ‘absolute’ path expressions

**Double Slash ‘// ’**

* Double slash ( // ) start selection matching anywhere in the document
* It enables to create ‘relative’ path expressions

1. **EXPLAIN USING WEBDRIVER HOW YOU CAN PERFORM DOUBLE CLICK?**

You can perform double click by using

Syntax- Actions act = new Actions (driver);

act.doubleClick(webelement);

1. **EXPLAIN USING WEBDRIVER HOW YOU CAN PERFORM RIGHT CLICK?**

Actions actions = new Actions(driver);

WebElement elementLocator = driver.findElement(By.id("ID"));

actions.contextClick(elementLocator).perform();

1. **What is Alert in Selenium?**

An Alert in Selenium is a small message box, which appears on screen to give the user some information or notification. It notifies the user with some specific information or error, asks for permission to perform certain tasks and it provides warning messages as well.

Alert interface provides the below few methods which are widely used in Selenium Webdriver.

* void dismiss() // To click on the 'Cancel' button of the alert.

driver.switchTo().alert().dismiss();

* void accept() // To click on the 'OK' button of the alert.

driver.switchTo().alert().accept();

* String getText() // To capture the alert message.

driver.switchTo().alert().getText();

* void sendKeys(String stringToSend) // To send some data to alert box.

driver.switchTo().alert().sendKeys("Text");

1. **HOW WILL YOU HANDLE WORKING WITH MULTIPLE WINDOWS IN SELENIUM ?**

In Selenium web driver, there are methods through which we can handle multiple windows.

**Driver.getWindowHandle();**

When the site opens, we need to handle the main window by driver.getWindowHandle(). This will handle the current window that uniquely identifies it within this driver instance. Its return type is String.

**Driver.getWindowHandles();**

To handle all opened windows by web driver, we can use "Driver.getWindowHandles()" and then we can switch window from one window to another in a web application. Its return type is Set<String>.

public void windowHandle(){  
 String parentWindow = scenarioContext.driver.getWindowHandle();  
 Set<String> childWindows = scenarioContext.driver.getWindowHandles();  
 for (String window : childWindows){  
 if (!window.equalsIgnoreCase(parentWindow)){  
 scenarioContext.driver.switchTo().window(window);  
 scenarioContext.driver.close();  
 }  
 }  
 scenarioContext.driver.switchTo().window(parentWindow);  
}

1. **EXPLAIN HOW YOU CAN SWITCH BETWEEN FRAMES?**

The iframe is defined with the <iframe> tag. We can even identify total number of iframes by using below snippet.

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

Basically, we can switch over the elements and handle frames in Selenium using 3 ways.

* By Index
* By Name or Id
* By Web Element

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

We have to come out of the iframe.

To move back to the parent frame, you can either use switchTo().parentFrame() or if you want to get back to the main (or most parent) frame, you can use switchTo().defaultContent();

driver.switchTo().parentFrame();

driver.switchTo().defaultContent();

1. **STATE THE DIFFERENCES BETWEEN CLOSE() AND QUIT()**

| **close()** | **quit()** |
| --- | --- |
| close() method shall close the browser which is in focus. | quit() method closes all the browsers. |
| close() method closes the active WebDriver instance. | quit() method closes all the active WebDriver instances. |

1. **DIFFERENCE BETWEEN EXAMPLES TABLE & DATA TABLE**

The Scenario Outline keyword can be used to run the same Scenario multiple times, with different combinations of values.

Scenario Outline is run once for each row in the Examples section beneath it (not counting the first header row).

This works for the whole test case or test scenario.

Cucumber automatically run the complete test the number of times equal to the number of data in the Test Set

Example tables always have a header row, because the compiler needs to match the header columns to the placeholders in the Scenario Outline’s steps.

\* Data Table:

Sample 1:

Given the following tic tac toe board:

| x | | |

| o | x | |

| x | | o |

Sample 2:

Given user is logged in to application

| username | admin |

| password | admin123 |

No keyword is used to define the test data

This works only for the single step, below which it is defined

A separate code is need to understand the test data and then it can be run single or multiple times but again just for the single step, not for the complete test

Data tables are passed wholesale to the step definitions, and it’s up to the user to interpret them. They don’t necessarily have a header row

\* So What To Use When?

Use Example Table where ENTIRE scenario needs to be tested with different/multiple test data.

Use Data table where test data is Explicitly meant for specific steps and user would like to interpret based on use case internally.

1. **TESTNG ANNOTATION**

**@Test**

@Test annotation describes method as a test method or part of your test.

**@BeforeMethod**

Any method which is marked with @BeforeMethod annotation will be executed before each and every @test annotated method.

**@AfterMethod**

Same as @BeforeMethod, If any method is annotated with @AfterMethod annotation then it will be executed after execution of each and every @test annotated method.

**@BeforeClass**

Method annotated using @BeforeClass will be executed before first @Test method execution. @BeforeClass annotated method will be executed once only per class so don't be confused.

**@AfterClass**

Same as @BeforeClass, Method annotated with @AfterClass annotation will be executed once only per class after execution of all @Test annotated methods of that class.

**@BeforeTest**

@BeforeTest annotated method will be executed before the any @Test annotated method of those classes which are inside <test> tag in testng.xml file.

**@AfterTest**

@AfterTest annotated method will be executed when all @Test annotated methods completes its execution of those classes which are inside <test> tag in testng.xml file.

**@BeforeSuite**

Method marked with @BeforeSuite annotation will run before the all suites from test.

**@AfterSuite**

@AfterSuite annotated method will start running when execution of all tests executed from current test suite.

**@DataProvider**

When you use @DataProvider annotation for any method that means you are using that method as a data supplier. Configuration of @DataProvider annotated method must be like it always return Object[][] which we can use in @Test annotated method.

**@BeforeGroups**

@BeforeGroups annotated method will run before the first test run of that specific group.

**@AfterGroups**

@AfterGroups annotated method will run after all test methods of that group completes its execution.

**@Parameters**

When we wants to pass parameters in our test methods, we need to use @Parameters annotation.

**@Factory**

When we wants to execute specific group of test cases with different values, we need to use @Factory annotation. An array of class objects is returned by @Factory annotated method and those TestNG will those objects as test classes.

**@Listeners**

@Listeners are used to with test class. It is helpful for logging purpose.