**Selenium Web Driver Waits**

1. Implicit Wait
2. Explicit Wait
3. Fluent Wait

**VIMP** : The **default** **polling** **time** is 500 mili seconds. So until your wait condition is true and the specified time is not over, the WebDriver checks for the element once every 500 mili seconds.

As per the official Selenium documentation, it is suggested not to mix both Implicit waits and Explicit Waits. Mixing both of them can cause unpredictable wait times.

**Implicit Wait:**

The implicit wait will tell web driver to wait for certain amount of time before it throws a "**No Such Element Exception**". An implicit wait tells WebDriver to wait for a certain amount of time when trying to find an element or elements if they are not immediately available on web page. NOTE : Implicitly wait is applied globally, which means it is always available for all the web elements throughout the driver instance. It implies that if the driver is interacting with 100 elements, then implicitly wait is applicable for all the 100 elements.

**Implicit Wait**: An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to **find** an element or elements if they are not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object instance.

WebDriver driver = new FirefoxDriver();

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

driver.get("http://somedomain/url\_that\_delays\_loading");

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

**When to use:** Not recommended

Once the command is in place, Implicit Wait stays in place for the entire duration for which the browser is open. If one sets an implicit wait command, then the browser will wait for the same time frame before loading every web element.

**Explicit wait**: An explicit wait is code defined to wait for a certain condition to occur before proceeding further in the code. OR Explicit waits are used to halt the execution till the time a particular condition is met. WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully.

Example :

WebDriverWait wait = new WebDriverWait(dr,15);

WebElement PIM= wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//a[@id='menu\_pim\_viewPimModule']")));

PIM.click();

**When to use**: If element takes a long time to load. Also, used to check property of an element (presence, clickability. etc).

**FluentWait**: it checks the presence of element after given time interval for given time. Ex. : it checks presence of element for 30 seconds after every 5 seconds (we have to mention that 5 and 30 seconds)

For each FluentWait instance, you can specify:

1. Frequency with which FluentWait has to check the conditions defined.
2. Ignore specific types of exception waiting such as NoSuchElementExceptions while searching for an element on the page.
3. Maximum amount of time to wait for a condition
4. When a FluentWait instance is implemented it defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions while waiting, such as NoSuchElementExceptions when searching for an element on the page.

**When to use FluentWait**: When you try to test the presence of an element that may appear after every x seconds/minutes (Just an example, this is my guess of where such a thing can be used).

// Waiting 30 seconds for an element to be present on the page, checking

// for its presence once every 5 seconds.

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(30, SECONDS)

.pollingEvery(5, SECONDS)

.ignoring(NoSuchElementException.class);

WebElement foo = wait.until(new Function<WebDriver, WebElement>()

{

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("foo"));

}

});