Selenium Wait – Implicit, Explicit and Fluent Waits

 By Krishna Rungta Updated August 26, 2022

In Selenium, “Waits” play an important role in executing tests. In this tutorial, you will learn various aspects and difference between Implicit and Explicit wait in Selenium.
In this tutorial, you will learn about different types of waits in Selenium:

Why Do We Need Waits In Selenium?

Most of the web applications are developed using Ajax and Javascript. When a page is loaded by the browser the elements which we want to interact with may load at different time intervals.

Not only it makes this difficult to identify the element but also if the element is not located it will throw an “**ElementNotVisibleException**” exception. Using Selenium Waits, we can resolve this problem.

Let’s consider a scenario where we have to use both implicit and explicit waits in our test. Assume that implicit wait time is set to 20 seconds and explicit wait time is set to 10 seconds.

Suppose we are trying to find an element which has some **“ExpectedConditions** “(Explicit Wait), If the element is not located within the time frame defined by the Explicit wait(10 Seconds), It will use the time frame defined by implicit wait(20 seconds) before throwing an “**ElementNotVisibleException**“.

Selenium Web Driver Waits

1. Implicit Wait
2. Explicit Wait

Implicit Wait in Selenium

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.
Selenium Web Driver has borrowed the idea of implicit waits from Watir.

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.

To declare implicit wait in Selenium WebDriver:

Implicit Wait syntax:

```
driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
package guru.test99;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class AppTest {

        protected WebDriver driver;
        @Test
        public void guru99tutorials() throws InterruptedException
        {
        System.setProperty ("webdriver.chrome.driver",".\\chromedriver.exe"
);
        driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS) ;
        String eTitle = "Demo Guru99 Page";
        String aTitle = "" ;
        // launch Chrome and redirect it to the Base URL
        driver.get("http://demo.guru99.com/test/guru99home/" );
        //Maximizes the browser window
        driver.manage().window().maximize() ;
        //get the actual value of the title
        aTitle = driver.getTitle();
        //compare the actual title with the expected title
        if (aTitle.equals(eTitle))
        {
        System.out.println( "Test Passed") ;
        }
        else {
        System.out.println( "Test Failed" );
        }
        //close browser
        driver.close();
}
}
```

Explanation of Code

In the above example,

Consider Following Code:

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

Implicit wait will accept 2 parameters, the first parameter will accept the time as an integer value and the second parameter will accept the time measurement in terms of SECONDS, MINUTES, MILISECOND, MICROSECONDS, NANOSECONDS, DAYS, HOURS, etc.

Explicit Wait in Selenium

The **Explicit Wait in Selenium** is used to tell the Web Driver to wait for certain conditions (Expected 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 Ajax elements.
Once we declare explicit wait we have to use "**ExpectedConditions**" or we can configure how frequently we want to check the condition using **Fluent Wait**.
These days while implementing we are using **Thread.Sleep()** generally it is not recommended to use

In the below example, we are creating reference wait for "**WebDriverWait**" class and instantiating using "**WebDriver**" reference, and we are giving a maximum time frame of 20 seconds.

Explicit Wait syntax:

```
WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);
package guru.test99;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.Test;

public class AppTest2 {
        protected WebDriver driver;
        @Test
        public void guru99tutorials() throws InterruptedException
        {
        System.setProperty ("webdriver.chrome.driver",".\\chromedriver.exe"
);
```

```
        driver = new ChromeDriver();
        WebDriverWait wait=new WebDriverWait(driver, 20);
        String eTitle = "Demo Guru99 Page";
        String aTitle = "" ;
        // launch Chrome and redirect it to the Base URL
        driver.get("http://demo.guru99.com/test/guru99home/" );
        //Maximizes the browser window
        driver.manage().window().maximize() ;
        //get the actual value of the title
        aTitle = driver.getTitle();
        //compare the actual title with the expected title
        if (aTitle.contentEquals(eTitle))
        {
        System.out.println( "Test Passed") ;
        }
        else {
        System.out.println( "Test Failed" );
        }
        WebElement guru99seleniumlink;
        guru99seleniumlink=
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
"/html/body/div[1]/section/div[2]/div/div[1]/div/div[1]/div/div/div/div[2]/
div[2]/div/div/div/div/div[1]/div/div/a/i")));
        guru99seleniumlink.click();
        }

}
```

Explanation of Code

Consider Following Code:

```
WebElement guru99seleniumlink;
guru99seleniumlink =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/bo
dy/div[1]/section/div[2]/div/div[1]/div/div[1]/div/div/div/div[2]/div[2]/di
v/div/div/div/div[1]/div/div/a/i")));
guru99seleniumlink.click();
```

In this WebDriver wait example, wait for the amount of time defined in the "**WebDriverWait**" class or the "**ExpectedConditions**" to occur whichever occurs first.

The above Java code states that we are waiting for an element for the time frame of 20 seconds as defined in the "**WebDriverWait**" class on the webpage until the "**ExpectedConditions**" are met and the condition is "**visibilityofElementLocated**".

The following are the Expected Conditions that can be used in Selenium Explicit Wait

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()

5. frameToBeAvaliableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()

Fluent Wait in Selenium

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.
Frequency: Setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time

Let’s consider a scenario where an element is loaded at different intervals of time. The element might load within 10 seconds, 20 seconds or even more then that if we declare an explicit wait of 20 seconds. It will wait till the specified time before throwing an exception. In such scenarios, the fluent wait is the ideal wait to use as this will try to find the element at different frequency until it finds it or the final timer runs out.

Fluent Wait syntax:

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(timeout, SECONDS)
.pollingEvery(timeout, SECONDS)
.ignoring(Exception.class);
```

Above code is deprecated in Selenium v3.11 and above. You need to use

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(Duration.ofSeconds(SECONDS))
.pollingEvery(Duration.ofSeconds(SECONDS))
.ignoring(Exception.class);
```

```
package guru.test99;

import org.testng.annotations.Test;
import java.util.NoSuchElementException;
import java.util.concurrent.TimeUnit;
import java.util.function.Function;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.FluentWait;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.Test;

public class AppTest3 {
        protected WebDriver driver;
        @Test
        public void guru99tutorials() throws InterruptedException
        {
        System.setProperty ("webdriver.chrome.driver",".\\chromedriver.exe"
);
        String eTitle = "Demo Guru99 Page";
        String aTitle = "" ;
        driver = new ChromeDriver();
        // launch Chrome and redirect it to the Base URL
        driver.get("http://demo.guru99.com/test/guru99home/" );
        //Maximizes the browser window
        driver.manage().window().maximize() ;
        //get the actual value of the title
        aTitle = driver.getTitle();
        //compare the actual title with the expected title
        if (aTitle.contentEquals(eTitle))
        {
        System.out.println( "Test Passed") ;
        }
        else {
        System.out.println( "Test Failed" );
                }

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

                        .withTimeout(30, TimeUnit.SECONDS)

                        .pollingEvery(5, TimeUnit.SECONDS)

                        .ignoring(NoSuchElementException.class);
        WebElement clickseleniumlink = wait.until(new Function<WebDriver,
WebElement>(){

                public WebElement apply(WebDriver driver ) {
                        return
driver.findElement(By.xpath("/html/body/div[1]/section/div[2]/div/div[1]/di
v/div[1]/div/div/div/div[2]/div[2]/div/div/div/div/div[1]/div/div/a/i"));
                }
        });
        //click on the selenium link
        clickseleniumlink.click();
        //close~ browser
        driver.close() ;
        }
```

```
}
```

Explanation of Code

Consider Following Code:

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

        .withTimeout(30, TimeUnit.SECONDS)
        .pollingEvery(5, TimeUnit.SECONDS)
        .ignoring(NoSuchElementException.class);
```

In the above example, we are declaring a fluent wait with the timeout of 30 seconds and the frequency is set to 5 seconds by ignoring "**NoSuchElementException**"

Consider Following Code:

```
public WebElement apply(WebDriver driver) {
        return
driver.findElement(By.xpath("/html/body/div[1]/section/div[2]/div/div[1]/di
v/div[1]/div/div/div/div[2]/div[2]/div/div/div/div/div[1]/div/div/a/i"));
```

We have created a new function to identify the Web Element on the page. (Ex: Here Web Element is nothing but the Selenium link on the webpage).

Frequency is set to 5 seconds and the maximum time is set to 30 seconds. Thus this means that it will check for the element on the web page at every 5 seconds for the maximum time of 30 seconds. If the element is located within this time frame it will perform the operations else it will throw an "**ElementNotVisibleException**"

Also Check:- Selenium IDE Tutorial for Beginners

Difference Between Implicit Wait Vs Explicit Wait

Following is the main difference between implicit wait and explicit wait in Selenium:

Implicit Wait	Explicit Wait
• Implicit Wait time is applied to all the elements in the script	Explicit Wait time is applied only to those elements which are intended by us
• In Implicit Wait, we need **not** specify "ExpectedConditions" on the element to be located	• In Explicit Wait, we need to specify "ExpectedConditions" on the element to be located

• It is recommended to use when the elements are located with the time frame specified in Selenium implicit wait	It is recommended to use when the elements are tak long time to load and also for verifying the property like(visibilityOfElementLocated, elementToBeClickable,elementToBeSelected)

Conclusion:

Implicit, Explicit and Fluent Wait are the different waits used in Selenium. Usage of these waits are totally based on the elements which are loaded at different intervals of time. It is always not recommended to use Thread.Sleep() while Testing our application or building our framework.