Implicit, Explicit and Fluent Waits

In this lesson, we will understand the wait command in Selenium.

WE'LL COVER THE FOLLOWING

What are wait commands in Selenium?

Wait commands are very important for writing stable and robust tests; there are many factors that can impact the stability of the automated testing of websites like page loads, such as Ajax calls, variations in time lag for loading elements, slowness of the environment, etc. Wait commands help in troubleshooting these issues and make tests stable and reliable. They help adding waits before or after a certain operation in the script and throw `NoSuchElementException` in case the element is not found.

Types of wait command

- Implicit wait
- Explicit wait
- Fluent wait

The below example code can be downloaded and run locally; it is included as a part of the Sample UI test framework, please refer.

Implicit wait

Once the Implicit wait is set, it stays in place for the entire duration for which the browser is open. Its default setting is **0** (zero). Once we set a specific time, WebDriver will wait for the element until that time before the exception occurs.

```
// Create a WebDriver object
WebDriver driver = DriverManager.getWebDriver();

// setting the implicit timeout to 20 seconds
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

// open the web url
driver.get("https://www.google.com");

// Find element will wait for 20 seconds for the WebElement
WebElement element = driver.findElement(By.name("q"));
element.sendKeys("educative", Keys.RETURN);

By locator = By.xpath("//a[contains(@href,'https://www.educative.io')]");
List<WebElement> list = driver.findElements(locator);

Assert.assertTrue(!list.isEmpty(), "number of elements is zero");
```

Explicit wait

The explicit wait command will wait until a certain condition occurs before proceeding with the executing code.

This is important in cases where certain elements naturally take more time to load. We can set variable wait for different Web Elements instead of setting one for all (as in case of Implicit wait) so that the wait will be as and only when required.

```
        // Create a WebDriver object
WebDriver driver = DriverManager.getWebDriver();

// open the web url
driver.get("https://www.google.com");

// create a WebDriver wait object, max wait time of 30 seconds before thro
wing
// an exception
WebDriverWait waitA = new WebDriverWait(driver, 30);
```

```
// wait for the visibility of the element

// once the element is visible, perform actions
WebElement element = waitA.until(ExpectedConditions.visibilityOfElementLoc
ated(By.name("q")));
element.sendKeys("educative", Keys.RETURN);

// create a WebDriver wait object, max wait time of 15 seconds before thro
wing
// an exception
WebDriverWait waitB = new WebDriverWait(driver, 15);
By locator = By.xpath("//a[contains(@href, 'https://www.educative.io')]");
List<WebElement> list = waitB.until(ExpectedConditions.presenceOfAllElemen
tsLocatedBy(locator));

Assert.assertTrue(!list.isEmpty(), "number of elements is zero");


// using Java 8 Lambda

WebElement element = wait.until( driver -> driver.findElement(By.xpath("//
a[contains(@href, 'https://www.educative.io')]")) );
```

Fluent wait

The fluent wait repeatedly command looks for a web element at regular intervals for a condition until timeout happens or until the object is found. It also defines the frequency with which WebDriver will check if the condition appears before throwing the `ElementNotVisibleException`.

Fluent Wait commands are most useful when interacting with web elements that can sometimes take more time than usual to load. For example, in Ajax applications.

```
 // Create a WebDriver object
WebDriver driver = DriverManager.getWebDriver();

// open the web url
driver.get("https://www.google.com");

// create a WebDriver wait object, max wait time of 30 seconds before thro
wing
// an exception
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
        .withTimeout(Duration.ofSeconds(30))
```

```
        .pollingEvery(Duration.ofSeconds(2))
        .ignoring(NoSuchElementException.class);

// wait for the visibility of the element
// once the element is visible, perform actions
WebElement element = wait.until(ExpectedConditions.visibilityOfElementLoca
ted(By.name("q")));
element.sendKeys("educative", Keys.RETURN);

// create a WebDriver wait object, max wait time of 15 seconds before thro
wing
// an exception
Wait<WebDriver> waitB = new FluentWait<WebDriver>(driver)
        .withTimeout(Duration.ofSeconds(15))
        .pollingEvery(Duration.ofSeconds(2))
        .ignoring(NoSuchElementException.class);

By locator = By.xpath("//a[contains(@href, 'https://www.educative.io')]");
List<WebElement> list = waitB.until(ExpectedConditions.presenceOfAllElemen
tsLocatedBy(locator));

Assert.assertTrue(!list.isEmpty(), "number of elements is zero");


// using Java 8 Lambda
WebElement element = wait.until( driver -> driver.findElement(By.xpath("//
a[contains(@href, 'https://www.educative.io')]")) );
```

That’s all for adding the wait commands for a stable automated testing. The next lesson introduces detailed information on handling exceptions.