Waits in Selenium or Synchronization in Selenium:

Waits are useful to achieve the synchronization between the speed of the java code with the web based application.

Waits are dynamic in nature which means it first check the condition and if it is favourable to perform the action then it allow the code to proceed otherwise it hold the code and continuous poll over the webpage to decide whether the condition got matched or not.

The default time of polling is 0.5 sec.

The maximum time a wait can hold the code is called timeout duration.

There are 3 types of Dynamic wait:

1. implicit wait

2. explicit wait

3. fluent wait

Also java provides a static wait as well which is Thread.sleep();

1. Implicit wait: This will wait till the element is not present over the page. If it is not available over the page then it will wait till the maximum configured duration and it polls for every 0.5 sec to the web page to verify whether element is available or not.

Implicit wait is also known as global wait because once it get defines it will be applicable for all the web elements defined below to it.

Once the implicit wait get fails I.e after waiting the configured duration if it doesn’t get the element over the page then it throws **NoSuchElementException**.

Example for implicit wait:

**public** **static** **void** main(String[] args) {

System.*setProperty*("webdriver.chrome.driver", "E:\\desktop\\Katraj\\15 Jan\\Selenium\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver();// chrome browser will get open

driver.manage().window().maximize();// to maximize the window

driver.manage().timeouts().~~implicitlyWait~~(50, TimeUnit.***SECONDS***);

driver.get("https://www.google.com/");

WebElement searchbox = driver.findElement(By.*xpath*("//\*[@title='Search']"));

Actions act = **new** Actions(driver);

act.click(searchbox).sendKeys("India").build().perform();

act.sendKeys(Keys.***ENTER***).perform();

driver.findElement(By.*xpath*("//\*[text()='Central Bank of India']")).click();

}

2. Explicit wait: The wait which holds the code till a particular condition of webeleemnt then we have to use Explicit wait.

We have some set of methods available in ExpectedConditions class and those methods can be used for particular condition of webelement.

Once Explicit wait gets fail that is the condition doesn’t meet with the webpage in that duration then after passing of maximum configured duration it throws **TimeOutException.**

Example:

**public** **static** **void** main(String[] args) {

System.*setProperty*("webdriver.chrome.driver", "E:\\desktop\\Katraj\\15 Jan\\Selenium\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver();// chrome browser will get open

driver.manage().window().maximize();// to maximize the window

driver.get("https://chercher.tech/practice/explicit-wait-sample-selenium-webdriver");

WebElement disablebutton = driver.findElement(By.*xpath*("//\*[@id='disable']"));

driver.findElement(By.*xpath*("//\*[@id='enable-button']")).click();

WebDriverWait wait = **new** WebDriverWait(driver, Duration.*ofSeconds*(60));

// wait till the element is not clickable over the page

wait.until(ExpectedConditions.*elementToBeClickable*(disablebutton));

driver.findElement(By.*xpath*("//\*[@id='enable-button']")).click();

Scenario 2:

Waiting until the element is visible over the page or not.

// waiting until the element is visible over the webpage

wait.until(ExpectedConditions.*visibilityOf*(hiddenbutton));

System.***out***.println("button is displayed on the page");

|  |  |  |
| --- | --- | --- |
| Sr no | **Implicit Wait** | **Explicit Wait** |
| 1 | The driver is asked to wait for a specific amount of time for the element to be available on the DOM of the page. | The driver is asked to wait till a certain condition is satisfied. |
| 2 | It is a global wait and applied to all elements on the webpage. | It is not a global wait and applied to a particular scenario. |
| 3 | It does not require you to meet any condition. | It is required to satisfy a particular condition. Some of the expected conditions include −   * visibility\_of * element\_selection\_state\_to\_be * presence\_of\_all\_elements\_located * element\_located\_to\_be\_selected * alert\_is\_present * element\_located\_selection\_state\_to\_b e * element\_to\_be\_clickable * text\_to\_be\_present\_in\_element   element\_to\_be\_selected |
| 4 | It throws NoSuchElementException on time out. | It throws TimeOutException on timeout. |

3. Fluent wait: Through this wait we have more customization options are available in terms of configuration of polling time, timeout etc.

**public** **static** **void** main(String[] args) {

System.*setProperty*("webdriver.chrome.driver", "E:\\desktop\\Katraj\\15 Jan\\Selenium\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver();// chrome browser will get open

driver.manage().window().maximize();// to maximize the window

driver.get("https://chercher.tech/practice/explicit-wait-sample-selenium-webdriver");

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

.withTimeout(Duration.*ofSeconds*(60))

.pollingEvery(Duration.*ofMillis*(20));

WebElement disablebutton = driver.findElement(By.*xpath*("//\*[@id='disable']"));

driver.findElement(By.*xpath*("//\*[@id='enable-button']")).click();

// wait till the element is not clickable over the page

wait.until(ExpectedConditions.*elementToBeClickable*(disablebutton));

driver.findElement(By.*xpath*("//\*[@id='enable-button']")).click();

// wait till the checkbox got selected

driver.findElement(By.*xpath*("//\*[@id='checkbox']")).click();

wait.until(ExpectedConditions.*elementSelectionStateToBe*(By.*xpath*("//\*[@id='ch']"), **true**));

System.***out***.println("verified check box has been checked");

}