SYNCHRONIZATION

While working on an automation project, the biggest challenge you will face is synchronization ie. syncing up automation scripts with the application under test. There are few web pages or web elements which load within no time and there are few which take comparatively longer time to load. In this tutorial, we will learn several types of wait statements that Selenium WebDriver offers.

**What is it -**

* Synchronization is when all the elements in page are in synchronized state such that the selenium script will be able to perform the operations smoothly

**Why is it used -**

* There are some cases when elements on the page takes time to load or the element loads in different time intervals. So, selenium script finds difficulty in searching for the element for 5 seconds. When the default time exceeds, the script will fail when it is unable to find the element within the specified time and exception will be thrown on the console window. Below are few common exceptions that occurs.
* NoSuchElementException: Occurs when the script is unable to find the element
* ElementNotVisibleException: Occurs when the script can find the element but is not visible on the page
* ElementNotSelectableException:Occurs when the script can find the element, but it is disabled for selection
* NoAlertPresentException:Occurs when the script tries to switch to the alert which is not available on the page
* NoSuchFrameException: Occurs when the script tries to switch to the frame which is not available on the page
* StaleElementReferenceException:  Occurs when the element is stale (i.e. no longer on the page due to page refresh or relocation of the element since it was loaded)
* TimeoutException: Occurs when the script doesn’t have enough time to locate the element or element is not identified within the specified time
* For more reference on selenium exceptions, refer to <https://seleniumhq.github.io/selenium/docs/api/py/common/selenium.common.exceptions.html>
* Above errors can occur either due to time shortage of loading elements (synchronization) or due to certain conditions such as frame unavailable, alert unavailable, element is stale and so on
* To avoid the failures while locating the elements that occurs due to time shortage problems, we need to wait for the element to get loaded on the page. Therefore, waits / synchronization comes into picture to avoid such difficulties and it’s classified into 3 categories.

**Types of Waits& how to useit –**

In Selenium WebDriver, to sync up scripts there are four types of wait:

* Page Load Timeout
* Implicit Wait
* Explicit Wait
* Fluent Wait

## ****1. PageLoadTimeout –****

This is the maximum time selenium waits for a page to load successfully on a browser. If the page takes more than this time, it will throw Page not found Exception.

 driver.manage().timeouts().pageLoadTimeout(90, TimeUnit.SECONDS);

**EXPLANATION-**In the above code , pageLoadTimeout() method is accepting two arguments, one is waiting time and  in another, we are specifying the Time Unit.

Here Selenium WebDriver instance will wait a maximum of 90 seconds for a webpage to load. If it is loaded before the specified wait time, the execution will move to the next line of the script. If it doesn’t get loaded in 90 seconds it will throw Timeout Exception.

**Now, let’s understand  h*ow a Web page gets loaded in a Web browser?***

* **WEB PAGE-** It is a document commonly written in Hypertext Markup Language (HTML) that is accessible through the Internet or other network using an Internet browser. A web page is accessed by entering a URL address and may contain text, graphics, and hyperlinks to other web pages and files. The page you are reading now is an example of a web page.
* **WEB BROWSER-**It knows how to go to a **Web server** on the Internet and request a page so that the browser can pull the page through the network and into your machine. A Web browser knows how to interpret the set of HTML tags within the page in order to display the page on your screen as the page’s creator intended it to be viewed.
* **WEB SERVER** – A Web server is a piece of computer software that can respond to a browser’s request for a page, and deliver the page to the Web browser through the Internet.

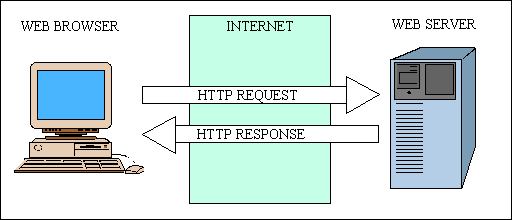


Fig. How page load works?

Whenever we type a URL in a browser, an HTTP request is triggered to the Web Server, which can be a GET or a POST request. WebServer responds to the browser with a GET response which has an attribute called content-length. The browser waits for the data which comes in chunks till data equal to this content-length attribute is received and then it generates an event trigger which signifies that page is loaded successfully. Selenium Page “load Timeout” command waits for this event trigger.

1. **Implicit Wait**: Allows us to wait for certain amount of time for all elements in the script

**Syntax**: driver.manage().timeouts().implicitlyWait(timeout, TimeUnit.SECONDS);

* implicitlyWait method accepts 2 parameters, the first parameter accepts the time as an integer value whereas the second parameter accepts the time measurement and it’s an Enumeration which can be defined in terms of SECONDS, MINUTES, MILLISECOND, MICROSECONDS, NANOSECONDS, DAYS, HOURS and so on

Code Snippet

public static void main(String[] args)

{

System.setProperty("webdriver.chrome.driver","C:\\Downloads\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

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

driver.findElement(By.linkText("Gmail")).click();

driver.close();

}

* In the above code, web driver will wait for 10 seconds to find the link named “Gmail” before throwing any exceptions such as NoSuchElementException, etc

1. **Explicit Wait**: Allows us to wait for certain time for specific element until the element meets certain condition. Generally, it’s used for dynamicelements / AJAX elements which loads at different time intervals

**Syntax**: WebDriverWait wait = new WebDriverWait(driver,timeout);

wait.until(ExpectedConditions.yourCondition(By.locator()));

* WebDriverWait is a class & constructor which is used for explicit wait and so object needs to be created to instantiate
* The constructor accepts 2 parameters: driver object & time as an integer value in terms of seconds

[WebDriverWait](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/WebDriverWait.html#WebDriverWait-org.openqa.selenium.WebDriver-long-)([WebDriver](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/WebDriver.html" \o "interface in org.openqa.selenium) driver, long timeOutInSeconds)

* ExpectedConditions class can be used for specifying the conditions based on the need. yourCondition denotes the condition which you desire to work upon (Ex: elementToBeClickable is one of such conditions which is used to wait until the element becomes clickable)
* For more reference on all conditions that can be used, please refer <https://seleniumhq.github.io/selenium/docs/api/dotnet/html/T_OpenQA_Selenium_Support_UI_ExpectedConditions.htm>

Code Snippet

public static void main(String[] args)

{

System.setProperty("webdriver.chrome.driver","C:\\Downloads\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

WebDriverWait wait = new WebDriverWait(driver,7);

wait.until(ExpectedConditions.elementToBeClickable(By.id("q"))).sendKeys(“Hello”);

driver.close();

}

* In the above code, web driver will wait for 7 seconds until the element becomes clickable. Once it becomes clickable within 7 seconds, it will go ahead and pass the string as “Hello” in the search box.Just in case, if the element is not clickable within 7 seconds, it will throw ElementNotClickable or TimeOutException

1. **Fluent Wait**: Allows us to wait for certain time for specific element until the element meets certain condition, it also gives an option to search for the element at different frequency until it exceeds the certain time. We can also ignore certain exceptions when the script tries to find the element at different frequency

**Syntax**:

Wait wait = **new**FluentWait(driver)

     .withTimeout(timeout, TimeUnit.SECONDS)

     .pollingEvery(timeout, TimeUnit.SECONDS)

     .ignoring(ExceptionHandler.**class**);

  WebElement element = wait.until(**newFunction**(<WebDriver,WebElement>)

{

**public** WebElement apply(WebDriver driver)

{

**return**driver.findElement(By.locator());

    }

  } ) ;

element.click();

Code Snippet

public static void main(String[] args)

{

System.setProperty("webdriver.chrome.driver","C:\\Downloads\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

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

Wait wait = **new**FluentWait(driver)

     .withTimeout(30, TimeUnit.SECONDS)

     .pollingEvery(5, TimeUnit.SECONDS)

     .ignoring(NoSuchElementException.**class**);

 WebElement element = wait.until(**newFunction**(<WebDriver,WebElement>)

{

**public** WebElement apply(WebDriver driver)

{

**return**driver.findElement(By.id(“q"));

    }

  } ) ;

element.click();

driver.close();

}

* In the above code, web driver will wait for 30 seconds while polling the element for every 5 seconds. Some dynamic/AJAX element keeps on changing or loading for every few seconds, so we shouldn’t end up in any exceptions while trying to find the element using polling mechanism (frequency of 5 seconds). So, we are using ‘ignoring’ method to avoid such exceptions

driver.set\_page\_load\_timeout(30)  
  
driver.implicitly\_wait(30)

wait= WebDriverWait(driver,10)  
wait.until(EC.presence\_of\_all\_elements\_located(By.XPATH,**"//\*[@id='src']/parent::div//li"**))