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
* ElementNotInteractable: Occurs when the script unable to perform any action on webelement
* 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 three types of wait:

* Page Load Timeout
* Implicit Wait
* Explicit 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.

Syntax : driver.set\_page\_load\_timeout()

driver.set\_page\_load\_timeout(90)

**EXPLANATION-**In the above code ,page\_load\_timeout() 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.

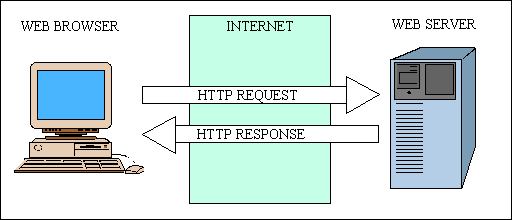


Fig. How page load works?

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.implicitly\_wait() *# seconds*

Eg : driver.implicitly\_wait(10) *# seconds*

An implicit wait tells WebDriver to poll the DOM for a certain amount of time when trying to find any element (or elements) not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object.

Code Snippet

**from** **selenium** **import** webdriver

driver = webdriver.Chrome()

driver.implicitly\_wait(10) *# seconds*

driver.get("http://www.google.com")

driver.find\_element\_by\_name("q").send\_keys(“Google”)

driver.find\_element\_by\_name("q").send\_keys(Keys.ENTER)

* In the above code, web driver will wait for 10 seconds to find the link named “Google Search bar” 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

An explicit wait is a code you define to wait for a certain condition to occur before proceeding further in the code. The extreme case of this is time.sleep(), which sets the condition to an exact time period to wait. There are some convenience methods provided that help you write code that will wait only as long as required. WebDriverWait in combination with ExpectedCondition is one way this can be accomplished.

**Syntax**: **from** **selenium.webdriver.support** **import** expected\_conditions **as** EC

wait = WebDriverWait(driver, 10)

element = wait.until(EC.element\_to\_be\_clickable((By.ID, 'someid')))

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

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

driver = webdriver.Firefox()

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

try:

element = WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.ID, "myDynamicElement"))

)

finally:

driver.quit()

* In the

This waits up to 10 seconds before throwing a TimeoutException unless it finds the element to return within 10 seconds. WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully. A successful return is for ExpectedCondition type is Boolean return true or not null return value for all other ExpectedCondition types.

**Expected Conditions**

There are some common conditions that are frequently of use when automating web browsers. Listed below are the names of each. Selenium Python binding provides some [convenience methods](http://selenium-python.readthedocs.io/api.html#module-selenium.webdriver.support.expected_conditions) so you don’t have to code an expected\_condition class yourself or create your own utility package for them.

* title\_is
* title\_contains
* presence\_of\_element\_located
* visibility\_of\_element\_located
* visibility\_of
* presence\_of\_all\_elements\_located
* text\_to\_be\_present\_in\_element
* text\_to\_be\_present\_in\_element\_value
* frame\_to\_be\_available\_and\_switch\_to\_it
* invisibility\_of\_element\_located
* element\_to\_be\_clickable
* staleness\_of
* element\_to\_be\_selected
* element\_located\_to\_be\_selected
* element\_selection\_state\_to\_be
* element\_located\_selection\_state\_to\_be
* alert\_is\_present

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

wait = WebDriverWait(driver, 10)

element = wait.until(EC.element\_to\_be\_clickable((By.ID, 'someid')))

The expected\_conditions module contains a set of predefined conditions to use with WebDriverWait.

**Fluent wait**

We don’t have fluent wait in Python, however it isn't packaged as simply as a FluentWait class. Some of this was covered in the documentation you provided by not extensively.

The WebDriverWait class has optional arguments for timeout, poll\_frequency, and ignored\_exceptions. So you could supply it there. Then combine it with an Expected Condition to wait for elements for appear, be clickable, etc...

Here is an example:

*from selenium import webdriver*

*from selenium.webdriver.common.by import By*

*from selenium.webdriver.support.ui import WebDriverWait*

*from selenium.webdriver.support import expected\_conditions as EC*

*from selenium.common.exceptions import \**

*driver = webdriver.Chrome()*

*# Load some webpage*

*wait = WebDriverWait(driver, 10, poll\_frequency=1, ignored\_exceptions=[ElementNotVisibleException, ElementNotSelectableException])*

*element = wait.until(EC.element\_to\_be\_clickable((By.XPATH, "//div")))*

Obviously you can combine the wait/element into one statement but I figured this way you can see where this is implemented.