5. Waits

These days most of the web apps are using AJAX techniques. When a page is loaded by the browser, the elements within that page may load at different time intervals. This makes locating elements difficult: if an element is not yet present in the DOM, a locate function will raise an *ElementNotVisibleException* exception. Using waits, we can solve this issue. Waiting provides some slack between actions performed - mostly locating an element or any other operation with the element.

Selenium Webdriver provides two types of waits - implicit & explicit. An explicit wait makes WebDriver wait for a certain condition to occur before proceeding further with execution. An implicit wait makes WebDriver poll the DOM for a certain amount of time when trying to locate an element.

5.1. Explicit Waits

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.

**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()

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

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.

**Custom Wait Conditions**

You can also create custom wait conditions when none of the previous convenience methods fit your requirements. A custom wait condition can be created using a class with *\_\_call\_\_* method which returns *False* when the condition doesn’t match.

**class** **element\_has\_css\_class**(object):

*"""An expectation for checking that an element has a particular css class.*

*locator - used to find the element*

*returns the WebElement once it has the particular css class*

*"""*

**def** \_\_init\_\_(self, locator, css\_class):

self.locator = locator

self.css\_class = css\_class

**def** \_\_call\_\_(self, driver):

element = driver.find\_element(\*self.locator) *# Finding the referenced element*

**if** self.css\_class **in** element.get\_attribute("class"):

**return** element

**else**:

**return** **False**

*# Wait until an element with id='myNewInput' has class 'myCSSClass'*

wait = WebDriverWait(driver, 10)

element = wait.until(element\_has\_css\_class((By.ID, 'myNewInput'), "myCSSClass"))

5.2. Implicit Waits

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.

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

driver = webdriver.Firefox()

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

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

myDynamicElement = driver.find\_element\_by\_id("myDynamicElement")