

# Implicit wait :-

def demo\_implicit\_wait(self):

        driver = webdriver.Firefox()

        driver.get("https://login.salesforce.com/?locale=au")

        driver.implicitly\_wait(10)

        # The above implicit wait will be applicable to all the

        # web elements mentioned in the test script below

        user\_name = driver.find\_element(by=By.ID, value="username")

        user\_name.send\_keys("santosh")

        pass\_word = driver.find\_element(by=By.ID, value="passwonjrd")

        pass\_word.send\_keys("santosh")

By implicitly waiting, WebDriver polls the DOM for a certain duration when trying to find any element. This can be useful when certain elements on the webpage are not available immediately and need some time to load. Once set, the implicit wait is set for the life of the session.

# Explicit wait :-

def demo\_explicit\_wait(self):

        driver = webdriver.Firefox()

        driver.get("https://www.yatra.com/")

        depart\_from = driver.find\_element(by=By.XPATH,value="//input[@id='BE\_flight\_origin\_city']")

        depart\_from.click()

        depart\_from.send\_keys("New Delhi")

        depart\_from.send\_keys(Keys.ENTER)

        going\_to = driver.find\_element(by=By.XPATH,value="//input[@id='BE\_flight\_arrival\_city']")

        going\_to.click()

        going\_to.send\_keys("New York")

        #from selenium.webdriver.support.wait import WebDriverWait

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

        wait = WebDriverWait(driver,10)

        wait.until(EC.element\_to\_be\_clickable((By.XPATH, "//input[@id='BE\_flight\_origin\_date']"))).click()

        # The above will check till the element is Clickable or not

        # and for that checking we are giving a time limit of 10 sec

        # Once it finds the element is clickable then it clicks

        # The below commented will be replaced by above lines

        # origin = driver.find\_element(by=By.XPATH,value="//input[@id='BE\_flight\_origin\_date']")

        # origin.click()

        all\_dates = wait.until(EC.element\_to\_be\_clickable((By.XPATH,"//div[@id='monthWrapper']//tbody//td[@class!='inActiveTD']")))\

            .find\_elements(By.XPATH,"//div[@id='monthWrapper']//tbody//td[@class!='inActiveTD']")

        # from the above line, before clicking an element we are checking the

        # the element is whether clickable or not

        for date in all\_dates:

            if date.get\_attribute("data-date") == "22/08/2022":

                date.click()

                break

        driver.find\_element(by=By.XPATH,value="change the xpath here").click()

They allow your code to halt program execution, or freeze the thread, until the condition you pass it resolves.

Since explicit waits allow you to wait for a condition to occur, they make a good fit for synchronising the state between the browser and its DOM, and your WebDriver script.

# Fluent wait :-

def demo\_fluent\_wait(self):

        driver = webdriver.Firefox()

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

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

        #from selenium.webdriver.support.wait import WebDriverWait

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

        #from selenium.common.exceptions import ElementNotVisibleException

        """Fluent wait is also similar to explicit wait

        but the difference is that fluent wait uses poll\_frequency

        poll\_frquency is nothing but imagine tha a web element will be activated

        after 10 sec of web page load so without throwing an exception

        we give a poll\_frequency = 3 i.e for every 3 sec it will check until 10sec

        mean while to avoid the exception we give another parameter as

        ignored\_exceptions= and mention the type of exception

        as shown in the above """

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

FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition.

Users may configure the wait to ignore specific types of exceptions whilst waiting, such as NoSuchElementException when searching for an element on the page.