Explicit Wait

Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed.

- Unlike Implicit waits, explicit waits are applied for a particular instance only.

- WebDriver introduces classes like WebDriverWait and Expected Conditions to enforce explicit waits into the test scripts.

Explicit waits are implemented using WebDriverWait and Expected Conditions class.

To implement the explicit wait,

First we need to create the object of WebDriverWait class, which takes two arguments.

1. Object of WebDriver reference variable.
2. Time out

Syntax:

```
WebDriverWait wait = new WebDriverWait(driver, 30);
```

- WebDriverWait provides two methods
  1. Until
  2. Until_not

Until -> Calls the method provided with the driver as an argument until the return value is not false.
Until_not -> Calls the method provided with the driver as an argument until the return value is not false.

- Until method takes one argument called condition methods.
- To pass the condition to the **until** method we use **Expected Condition class.**

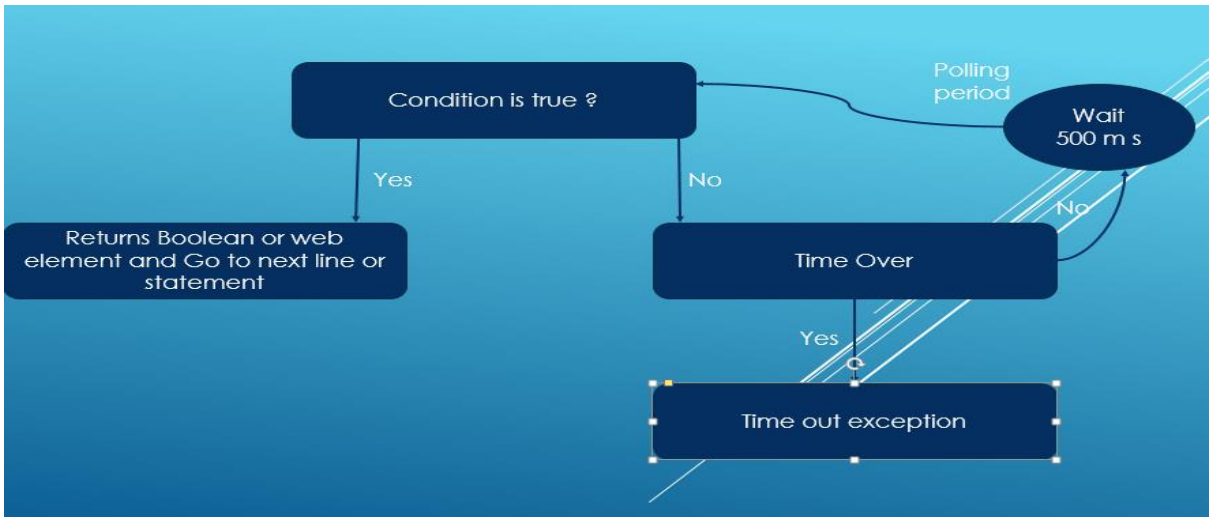
Expected Condition

There are some common conditions that are frequently came across when automating web browsers. Listed below are implementations of each. All the below mentioned condition methods are static method.

The conditions are present in a class called expected conditions. These conditions are also called as predicate.

1. Title is
2. Title contains
3. Alert is present

4. Visibility of element located
5. Element to be clickable
6. Element to be selected
7. Invisibility of element located
8. Presence of element located
9. Text to be present in element
10. Text to be present in element value
11. URL changes
12. URL contains



Explanation:

1. When the control comes to wait until statement it will check the condition.
2. If the condition is true it will go to next statement.
3. If the condition is false it will check for the timeout, if the time is over we get the timeout exception.
4. If the time is not over it will wait for 500 mille seconds and it will continue to check the condition.
5. Even after the specified time, if the condition is not satisfied it will throw the time out exception.

Note: When the condition is fails, explicit wait will always throw the time out exception for all the conditions.

When you are using explicit wait following are the two packages which we need to import.

```
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
```

Can we handle no such element exception using explicit wait?
Or
Can we handle synchronization of find element method using explicit wait?
Answer: Yes

- **title_is**
  - This condition has a string parameter.
  - The wait applies the condition to the parameter.
  - If the result of applying the condition is true, true is returned as result.
  - If the result is false, the wait tries the condition again after a short delay.
  - While the explicit wait applies the expected condition, the condition code may generate time out exceptions.

```
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class EC_title_is {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");

        try {
            WebDriverWait wait = new WebDriverWait(driver, 30);
            Boolean returnValue = wait.until(ExpectedConditions.titleIs("actiTIME - Login"));
            if (returnValue) {
                System.out.println("Login page is displayed");
            }
        } catch (TimeoutException e) {
            System.out.println("Login page is not displayed: "+e.toString());
        }
        driver.close();
    }
}
```

- **title_contains**
  - Expect to check that the title contains a sub-string (parameter: title)
  - Returns "**True**", if the page contains the title or else throw Timeout exception.

```
package synchronization;

import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class Title_contains {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");

        try {
            WebDriverWait wait = new WebDriverWait(driver, 30);
            Boolean returnValue = wait.until(ExpectedConditions.titleIs("actiTIME - Login"));
            if (returnValue) {
                System.out.println("Login page is displayed");
            }
        } catch (TimeoutException e) {
            System.out.println("Login page is not displayed: "+e.toString());
        }
        driver.close();
    }
}
```

- **element_to_be_clickable**
  - Expect for the element to become visible and enabled so that you can click it.
  - Two overloaded methods are there, one method takes "**By**" as an arguments and another method takes "**web element**" as an argument.
  - The wait applies the condition which tries finding the web element, depending on its status.
  - If the condition can find the element, it returns the element as result.
  - If it cannot find the element, the wait tries the condition again after a short delay.
  - If the condition fails even after the time out, it will through time out exception.

```
package synchronization;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class EC_elementToBeClickable
{
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

        WebDriverWait wait = new WebDriverWait(driver, 30);
        WebElement returnValue = wait.until(ExpectedConditions.elementToBeClickable(By.id("loginButton")));
        returnValue.click();

        driver.close();
    }
}
```

- **presence_of_element_located**
  - expect to check that an element is present on the DOM of a page
  - (parameter: locator)
  - Returns "**Web Element**", if the element is present on the web page else throws Timeout exception.

```
package synchronization;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class EC_PresenceOfElement {
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.vtiger.com/");
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

        WebDriverWait wait = new WebDriverWait(driver, 30);
        WebElement returnValue = wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("(//div[@class='bx-wrapper'])[1]")));
        System.out.println(returnValue.getText());

        driver.close();
    }
}
```

Difference between implicit wait and explicit wait

Implicit wait	Explicit wait
Waiting condition is built-in	We should mention the waiting condition
We can handle synchronization of all find element and find elements	We can handle synchronization of any methods but one at a time
If time is over we get no such element exception	If time is over we get time out exception
Time unit can be set to days, hours, minutes, milli seconds, micro seconds and Nano seconds	Here time unit can be only seconds