It’s always good to start off with some example which we will try to solve using the explicit wait. On our Practice Page we have a button element near the bottom of the page. Locator of this element is id=”colorVar“ , this element changes color of the text from white to red after a few seconds. Our problem to solve will be to wait for the element till the color of the element is White and click on the element as soon as the color becomes Red.

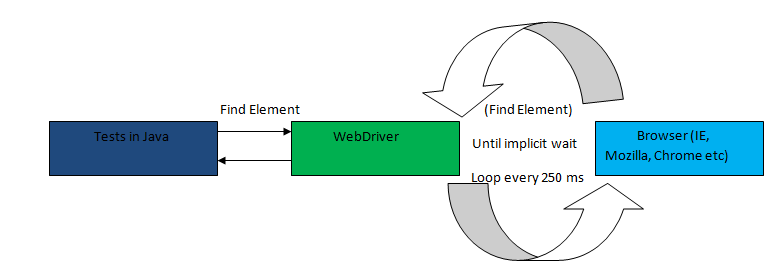
We will solve this problem using following waits

* WebDriverWait
* DefaultWait

**General logic of Explicit waits fluentwaits selenium C#**

The above two waits, in general the explicit waits, are smart waits. They are called smart primarily because they don’t wait for the max time out. Instead it waits for the time till the condition specified in .until(YourCondition) method becomes true.

A flow diagram explaining the working of Fluent wait is explained in the below diagram.



When the until method is called, following things happen in strictly this sequence

Step 1: In this step smart/explicit wait captures the wait start time.

Step 2: Smart/Explicit wait checks the condition that is mentioned in the .until() method

Step 3: If the condition is not met, a thread sleep is applied with time out of the value mentioned in the .pollingInterval property call. In the example above it is of 250 milliseconds.

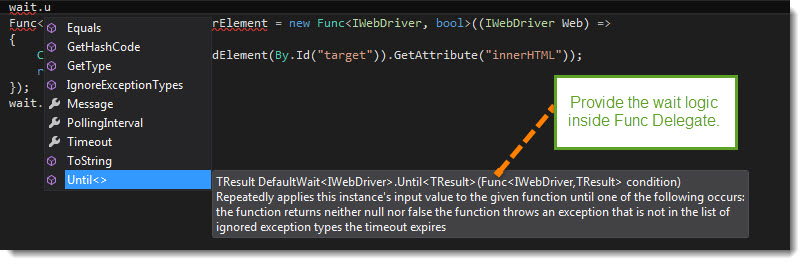
Step 4: Once the thread sleep of step 3 expires, a check of start time is made with the current time. If the difference between wait start time, as captured in step 1, and the current time is less than time specified in .Timeout property then step 2 is repeated.

This process keeps on happening till the time either the time out expires or the condition comes out to be true. Important point here is that your wait condition is evaluated after end of every polling time duration. So we don’t actually have to necessarily wait for the complete timeout.

**WebDriver Wait:**

WebDriverWait is present in the OpenQA.Selenium.Support.UI namespace. This wait is a specialized form of DefaultWait class. We will discuss DefaultWait in the next section. Below is the code which uses a WebDriverWait

Here the whole logic of wait is encapsulated inside the Func delegate, which is present in System namespace. We have to pass Func delegate to the WebDriverWait.until method.



You can define a Func delegate like this Func<Type1, Type2> where Type1 is the input parameter and Type2 is the output parameter. WebDriverWait is tied to WebDriver, hence Type1 will always be equal to WebDriver and Type2, which is the return type, can be anything you want. In this example we have choosen the return type to be bool.

Here you can clearly see that the Func delegate return the same type as specified in Type2.

A small change in this Func definition can make us return a WebElement. With a small change we can now wait till the element is not present in Dom and as soon as the element is present we can return it.

**DefaultWait:**

Default wait is more generic wait in the sense that it is not bound to WebDriver. This wait can be used to pass in any object to wait on. Let’s try to solve the above problem via DefaultWait now. We will first try to wait on the WebElement, here is the code.

**Exercise 1:**

1. Go to <http://toolsqa.wpengine.com/automation-practice-switch-windows/>
2. There is a clock on the page that counts down till 0 from 60 second.
3. You have to wait for the clock to show text “Buzz Buzz”
4. Try to use both WebDriverWait and DefaultWait. Solution with WebDriverWait is show below