Comandos de Espera en Selenium C#: Esperas Implícitas, Explícitas y Fluent Waits

Un framework de código abierto ampliamente utilizado para pruebas automatizadas es Selenium, que garantiza que las aplicaciones web se ejecuten correctamente en todos los navegadores y sistemas operativos. Como resultado, Selenium se ha vuelto muy popular en la industria de pruebas.

Para probar mejor nuestras aplicaciones web y resolver los desafíos que enfrentamos, es importante aprender ciertas buenas prácticas de Selenium al escribir el código de prueba. Uno de los factores más importantes en Selenium C# es la capacidad de usar comandos de espera (Wait commands) para reducir la inestabilidad de las pruebas.

Veamos cómo los comandos de espera en Selenium C y C# nos ayudan a resolver problemas de desfase de tiempo en nuestras aplicaciones web.

Tabla de Contenidos

¿Qué son los Comandos de Espera en Selenium?

Los comandos de espera en Selenium se utilizan en la automatización de pruebas para manejar la carga dinámica de elementos web. Como algunos elementos pueden haber sido creados con la ayuda de AJAX o frameworks frontend, tardan cierto tiempo en cargarse, y si los tests se ejecutan antes de que estén disponibles, fallan.

Los comandos de espera en Selenium pausan la ejecución de la prueba durante un tiempo determinado hasta que los elementos estén listos en el DOM. Como resultado, nuestras pruebas Selenium son más confiables.

Tipos de Comandos de Espera en Selenium

El framework Selenium ofrece tres tipos de comandos de espera.

Comando de Espera Implícita en C#

En Selenium C#, Implicit Wait se utiliza para indicar al WebDriver que espere un tiempo determinado antes de lanzar una excepción si no puede encontrar un elemento. Esta espera se aplica globalmente para todos los elementos, lo que significa que el WebDriver

verificará repetidamente la presencia de un elemento en el DOM durante el período especificado antes de continuar con el siguiente paso.

La espera implícita pausa la ejecución del WebDriver durante un período específico antes de lanzar cualquier error. El tiempo especificado se basa en el tiempo necesario para que los elementos web estén listos para la prueba y, por lo tanto, cargados en la página. Sin embargo, el tiempo total de ejecución de la prueba aumenta.

Si un elemento requiere más tiempo que el especificado, Selenium WebDriver lanza el error **"NoSuchElementException"**.

Sintaxis de la Espera Implícita en C#:

```
driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(Value);
```

Ejemplo de uso de Implicit Wait

```
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace Test
{
  class Program
  {
    Public static void Main(string[] args)
    {
      //Initialising ChromeDriver
      IWebDriver driver = new ChromeDriver();

      //Navigating to Google's homepage
      driver.Navigate().GoToUrl(" https://www.google.com ");

      //Applying Implicit Wait command for 20 seconds
      driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(20);

      //Clicking on an element after waiting for 20 seconds
      driver.FindElement(By.LinkText("I'm Feeling Lucky")).Click();
    }
```

```
  }
}
```

Cuándo usar Implicit Wait

- Para esperas generales cuando no estás seguro del tiempo de carga de los elementos.
- Para páginas estáticas donde los elementos suelen estar disponibles sin mucho retraso.
- Cuando quieres una configuración de espera global y simple para todas las búsquedas de elementos.

Cuándo NO usar Implicit Wait

- En páginas dinámicas o con llamadas AJAX donde los elementos cargan en distintos tiempos.
- Cuando necesitas control preciso del tiempo de espera: en estos casos, se prefiere Explicit Wait.

Comando de Espera Explícita en C#

En Selenium C#, Explicit Wait se utiliza para esperar a que ocurra una condición específica antes de continuar con las acciones. A diferencia de Implicit Wait, que se aplica globalmente, Explicit Wait es más preciso y permite esperar condiciones específicas, como que un elemento sea visible, clickeable o esté presente.

La espera implícita espera un período fijo, mientras que la espera explícita espera hasta que se cumpla una condición. No se basa en un tiempo fijo, sino en que el elemento esté listo en el DOM. Explicit Wait en C# también se conoce como **"smart wait"**.

El comando Explicit Wait verifica la condición cada **250 ms**. Además, Implicit Wait solo aplica a FindElement(), mientras que Explicit Wait permite múltiples condiciones.

Clases importantes para usar Explicit Wait

- **WebDriverWait**
- **ExpectedConditions**

WebDriverWait llama a los métodos de ExpectedConditions hasta que la condición pasa o se alcanza el tiempo máximo.

Sintaxis:

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

wait.Until(ExpectedConditions.ElementExists(By.Id("id")));
```

Métodos comunes de ExpectedConditions

- AlertIsPresent()

- ElementIsVisible()
- ElementExists()
- ElementToBeClickable(By)
- ElementToBeClickable(IWebElement)
- ElementToBeSelected(By)
- TitleContains()
- UrlMatches()
- TextToBePresentInElementValue(IWebElement, String)
- TextToBePresentInElement()

Ejemplo de Explicit Wait en C#

Requiere instalar:

- OpenQA.Selenium.Support.UI
- SeleniumExtras.WaitHelpers

```
using System;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;
using System;

namespace ExplicitWait
{
  class ExplicitWait
  {
    IWebDriver driver;

    [SetUp]
    public void open_gmail()
    {
```

```
        driver = new ChromeDriver();
        driver.Manage().Window.Maximize();
    }

    [Test]
    public void login_and_wait()
    {
        String email_site = "https://gmail.com";
        driver.Url = email_site;

        IWebElement Email = driver.FindElement(By.Id("Email"));
        Email.SendKeys("your_username");
        Email.SendKeys(Keys.Return);

        IWebElement Password = driver.FindElement(By.Id("Email"));
        Password.SendKeys("your_password");
        Password.SendKeys(Keys.Return);

        driver.FindElement(By.Id("signIn")).Click();

        String xpath = "//div[contains(text(),'COMPOSE')]";

        WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

        IWebElement ComposeButton =
wait.Until(ExpectedConditions.ElementExists(By.XPath(xpath)));

        ComposeButton.Click();

        driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(30);
    }
```

```
    [TearDown]
    public void close_Browser()
    {
      driver.Quit();
    }
  }
}
```

Cuándo usar Explicit Wait

- Cuando debes esperar condiciones específicas (visibilidad, clicabilidad, etc.).
- Contenido dinámico o cargado con AJAX.
- Para evitar errores de elementos obsoletos (stale element).
- Transiciones de página.
- Para mejorar la estabilidad cuando hay retrasos impredecibles.

Cuándo NO usar Explicit Wait

- En contenido estático.
- Pruebas simples donde los elementos aparecen rápidamente.
- Cuando una espera global ya cubre todos los casos.
- Cuando se abusa de ellos y se vuelve código lento y complejo.

Ignorar excepciones durante la espera

Excepciones comunes:

- NoSuchElementException
- StateElementReferenceException
- ElementNotVisibleException
- ElementNotSelectableException
- NoSuchFrameException

Sintaxis:

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.IgnoreExceptionTypes(typeof(ElementNotSelectableException));
```

Comando de Espera Fluent en C#

Fluent Wait es una versión más flexible de Explicit Wait. Permite definir:

- Tiempo máximo de espera
- Frecuencia de verificación (polling)
- Excepciones a ignorar

Es útil cuando un elemento puede aparecer en momentos variables.

La ventaja principal es controlar la frecuencia de verificación. Por defecto, Selenium verifica cada **500 ms**.

Sintaxis:

```
DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);


fluentWait.Timeout = TimeSpan.FromSeconds(5);


fluentWait.PollingInterval = TimeSpan.FromMilliseconds(polling_interval_in_ms);
```

Ejemplo de Fluent Wait

```
using System;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;
using System;

namespace Selenium_ExplicitWait_Demo
{
  class Selenium_ExplicitWait_Demo
  {
    IWebDriver driver;

    [SetUp]
```

```
public void open_gmail()
{
   driver = new ChromeDriver();
   driver.Manage().Window.Maximize();
}

[Test]
public void login_and_wait()
{
   DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);

   fluentWait.Timeout = TimeSpan.FromSeconds(5);
   fluentWait.PollingInterval = TimeSpan.FromMilliseconds(250);

   String email_site = "https://gmail.com";
   driver.Url = email_site;

   IWebElement Email = driver.FindElement(By.Id("Email"));
   Email.SendKeys("your_username");
   Email.SendKeys(Keys.Return);

   IWebElement Password = driver.FindElement(By.Id("Email"));
   Password.SendKeys("your_password");
   Password.SendKeys(Keys.Return);

   driver.FindElement(By.Id("signIn")).Click();

   String xpath = "//div[contains(text(),'COMPOSE')]";

   IWebElement ComposeButton = fluentWait.Until(dom =>
dom.FindElement(By.XPath(xpath)));
```

```
        ComposeButton.Click();


        driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(30);
    }


    [TearDown]
    public void close_Browser()
    {
        driver.Quit();
    }
  }
}
```

Cuándo usar Fluent Wait

- Cuando necesitas controlar la frecuencia de verificación.
- Manejo de contenido altamente dinámico.
- Ignorar excepciones como NoSuchElementException.
- Escenarios de espera complejos.

Cuándo NO usar Fluent Wait

- Escenarios simples o páginas estáticas.
- Cuando afecta negativamente el rendimiento por muchas verificaciones.
- Cuando una espera implícita ya es suficiente.
- Cuando un Explicit Wait resuelve el problema sin complejidad.

Sintaxis para ignorar excepciones en Fluent Wait

```
fluentWait.IgnoreExceptionTypes(typeof(ElementNotSelectableException));
```