Selenium testcase Steps

Create New Project 🡺NunitTestProject

Nuget package solution

Selenium.WebDriver

Selenium.Support

DotNetSeleniumExtras.WaitHelpers

[Test]

public void Test1()

{

IWebDriver driver = new ChromeDriver();

driver.Navigate().GoToUrl("http://www.google.com");

driver.Manage().Window.Maximize();

IWebElement webElement = driver.FindElement(By.Name("q"));

webElement.SendKeys("Azure tutorials");

webElement.SendKeys(Keys.Return);

driver.Close();

}

[Test]

public void EAWebsiteTest()

{

IWebDriver driver=new ChromeDriver();

driver.Navigate().GoToUrl("http://eaapp.somee.com/");

IWebElement loginLink = driver.FindElement(By.Id("loginLink"));

loginLink.Click();

IWebElement txtUsername=driver.FindElement(By.Id("UserName"));

txtUsername.SendKeys("admin");

IWebElement txtPassword = driver.FindElement(By.Id("Password"));

txtPassword.SendKeys("password");

IWebElement btnLogin = driver.FindElement(By.Id("loginIn"));

btnLogin.Click();

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

wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(By.TagName("img")));

// Verify the image is present

try

{

IWebElement imageElement = driver.FindElement(By.CssSelector("img[src='/Image/EA\_banner\_white\_v1.jpg']"));

Assert.IsTrue(imageElement.Displayed, "Image is displayed on the page.");

Console.WriteLine("Image is present on the page.");

}

catch (NoSuchElementException)

{

Assert.Fail("Image not found on the page.");

}

}

**What is customMethod in Selenium?**

In Selenium, there is **no built-in method called customMethod**. However, the term "custom method" generally refers to **user-defined functions** that enhance test automation by simplifying repetitive tasks or improving code reusability.

**Why Use Custom Methods in Selenium?**

Custom methods are used to:

1. **Improve Code Reusability** – Instead of writing the same Selenium commands multiple times, you can create reusable methods.
2. **Increase Readability & Maintainability** – Makes test scripts cleaner and easier to understand.
3. **Handle Common Web Actions** – Methods can be created for frequently performed operations like clicking, entering text, waiting for elements, handling alerts, etc.
4. **Enhance Error Handling** – You can add custom exception handling or logging inside a method.
5. **Implement Wait Strategies** – Instead of repeatedly writing WebDriverWait, you can create a method to handle waits dynamically.

Create another class SeleniumHelper.cs

using OpenQA.Selenium;

using OpenQA.Selenium.Support.UI;

using SeleniumExtras.WaitHelpers;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SeleniumTestProject

{

public class SeleniumHelper

{

private readonly IWebDriver driver;

private readonly WebDriverWait \_wait;

public SeleniumHelper(IWebDriver driver,int timeoutInSeconds=10)

{

this.driver = driver;

\_wait = new WebDriverWait(driver, TimeSpan.FromSeconds(timeoutInSeconds));

}

public void NavigateUrl(string url)

{

driver.Navigate().GoToUrl(url);

}

// Method to click an element after waiting

public void ClickElement(By locator)

{

\_wait.Until(ExpectedConditions.ElementToBeClickable(locator)).Click();

}

// Method to enter text into an input field

public void EnterText(By locator, string text)

{

var element = \_wait.Until(ExpectedConditions.ElementIsVisible(locator));

element.Clear();

element.SendKeys(text);

}

// Method to wait for an element to be present

public void WaitForElement(By locator)

{

\_wait.Until(ExpectedConditions.ElementExists(locator));

}

// Method to check if an element is displayed

public bool IsElementDisplayed(By locator)

{

try

{

return driver.FindElement(locator).Displayed;

}

catch (NoSuchElementException)

{

return false;

}

}

}

}

UnitTest1.cs

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.Support.UI;

namespace SeleniumTestProject

{

public class Tests

{

private IWebDriver driver;

private SeleniumHelper helper;

[SetUp]

public void Setup()

{

driver = new ChromeDriver();

helper = new SeleniumHelper(driver);

}

[Test]

public void Test1()

{

IWebDriver driver = new ChromeDriver();

driver.Navigate().GoToUrl("http://www.google.com");

driver.Manage().Window.Maximize();

IWebElement webElement = driver.FindElement(By.Name("q"));

webElement.SendKeys("Azure tutorials");

webElement.SendKeys(Keys.Return);

driver.Close();

}

[Test]

public void EAWebsiteTest()

{

IWebDriver driver=new ChromeDriver();

driver.Navigate().GoToUrl("http://eaapp.somee.com/");

IWebElement loginLink = driver.FindElement(By.Id("loginLink"));

loginLink.Click();

IWebElement txtUsername=driver.FindElement(By.Id("UserName"));

txtUsername.SendKeys("admin");

IWebElement txtPassword = driver.FindElement(By.Id("Password"));

txtPassword.SendKeys("password");

IWebElement btnLogin = driver.FindElement(By.Id("loginIn"));

btnLogin.Click();

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

wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(By.TagName("img")));

// Verify the image is present

try

{

IWebElement imageElement = driver.FindElement(By.CssSelector("img[src='/Image/EA\_banner\_white\_v1.jpg']"));

Assert.IsTrue(imageElement.Displayed, "Image is displayed on the page.");

Console.WriteLine("Image is present on the page.");

}

catch (NoSuchElementException)

{

Assert.Fail("Image not found on the page.");

}

}

[Test]

public void LoginAndVerifyImage()

{

helper.NavigateUrl("http://eaapp.somee.com/");

// Click on login link

helper.ClickElement(By.Id("loginLink"));

// Enter username and password

helper.EnterText(By.Id("UserName"), "admin");

helper.EnterText(By.Id("Password"), "password");

// Click login button

helper.ClickElement(By.Id("loginIn"));

// Wait for image to be present

helper.WaitForElement(By.TagName("img"));

// Verify if the image is displayed

bool isImageDisplayed = helper.IsElementDisplayed(By.CssSelector("img[src='/Image/EA\_banner\_white\_v1.jpg']"));

Assert.IsTrue(isImageDisplayed, "Image is displayed on the page.");

}

[TearDown]

public void TearDown()

{

driver.Quit();

}

}

}