Exercise: Selenium WebDriver POM

The objective of this exercise is to get you familiar with the concept of the **Page Object Model (POM**) in Selenium WebDriver using C#. You will learn to automate the login, product selection, cart management, and checkout process on the Swag Labs application (**https://www.saucedemo.com/**).

Application Overview:

Swag Labs is a demo e-commerce application designed to help QA engineers practice automation testing. The application includes a login page, an inventory page displaying various products, a shopping cart, and a checkout process.

0. Project Setup and Structure

- **Create a new C# project in Visual Studio**
- **Install the required NuGet packages:**
  - `Selenium.WebDriver`
  - `Selenium.Support`
  - `Selenium.WebDriver.ChromeDriver`
  - `SeleniumExtras.WaitHelpers`
  - `etc.`
- **Create the following directory structure:**

```
SwagLabsAutomation/
── Pages/
      ── BasePage.cs
      ── LoginPage.cs
      ── InventoryPage.cs
      ── CartPage.cs
      ── CheckoutPage.cs
      ── HiddenMenuPage.cs
── Tests/
      ── BaseTest.cs
      ── LoginTests.cs
      ── InventoryTests.cs
      ── CartTests.cs
      ── CheckoutTests.cs
      ── HiddenMenuTests.cs
```

1. BasePage

A **BasePage** class serves as a parent class for all page objects. It should contain **common methods** and **functionalities** that can be reused across different page objects, reducing code duplication and enhancing maintainability.

The **constructor** in the **BasePage** class should **initialize the WebDriver and set up an explicit wait mechanism**. This setup ensures that all page objects inheriting from BasePage have access to these functionalities

```
5 references
public BasePage(IWebDriver driver)
{
    this.driver = driver;
    wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
}
```

Typical functionalities included in a BasePage are:

- Find a **single web element** that matches the provided **By locator**

```
3 references
protected IWebElement FindElement(By by)
{
    return wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementIsVisible(by));
}
```

- Find **all web elements** that match the provided **By locator**

```
1 reference
protected IReadOnlyCollection<IWebElement> FindElements(By by)
{
    return driver.FindElements(by);
}
```

- **Clicking** element

```
1 reference
protected void Click(By by)
{
    FindElement(by).Click();
}
```

- **Typing** into input field

```
2 references
protected void Type(By by, string text)
{
    var element = FindElement(by);
    element.Clear();
    element.SendKeys(text);
}
```

- **Getting text** from elements

```
1 reference
protected string GetText(By by)
{
    return FindElement(by).Text;
}
```

By Class

Element Locators with By Class:

The By class in Selenium provides various methods to locate elements on a web page.

These methods include: **`By.Id`, `By.Name`, `By.ClassName`, `By.CssSelector`, `By.XPath`, etc**.

Using the **By class** is a more structured and efficient way to locate elements compared to the traditional FindElement method. It allows you to define locators separately and reuse them easily.

Comparing `By` Locators with Traditional `FindElement`:

Traditional FindElement: Directly finding elements in the methods where they are used.

```
IWebElement usernameField = driver.FindElement(By.Id("user-name"));
```

Using By Locators: Defining locators separately and using them in methods.

```
By usernameField = By.Id("user-name");
IWebElement element = driver.FindElement(usernameField);
```

2. BaseTest

The **`BaseTest`** class handles the **common setup** and **teardown** operations for initializing the WebDriver and performing **clean-up tasks** after each test. Here's a detailed explanation of each part of the **`BaseTest`** class:

- **WebDriver:** This field holds the WebDriver instance that will be used for browser interactions.

```
protected IWebDriver driver;
```

- **Setup Method:**
  - **[SetUp] Attribute:** Indicates that the Setup method should be run before each test.
  - **ChromeOptions:** Configures specific options for the Chrome browser **`chromeOptions.AddUserProfilePreference("profile.password_manager_enabled", false);`** This disables the Chrome password manager, preventing it from interfering with the test by showing password save prompts or compromised passwords pop-ups (Do you remember recording with Selenium IDE?).

```
var chromeOptions = new ChromeOptions();
chromeOptions.AddUserProfilePreference("profile.password_manager_enabled", false);
```

  - **ChromeDriver Path:** Specifies the path to the ChromeDriver executable.
  - **WebDriver Instance:** Initializes a new instance of ChromeDriver with the specified path and options.

```
string chromeDriverPath = @"C:\Program Files\ChromeDriver\chromedriver.exe";
driver = new ChromeDriver(chromeDriverPath, chromeOptions);
```

  - **Browser Window Management:** Maximizes the browser window to ensure that all elements are visible and to provide a consistent viewport size for the tests.

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

  - **Implicit Wait:** Wait up to 10 seconds for elements to be present in the DOM before throwing a **`NoSuchElementException`**.

```
driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);
```

- **Teardown Method:** Runs after each test. Ensures that the browser is closed and WebDriver resources are released.

```
[TearDown]
0 references
public void Teardown()
{
    if (driver != null)
    {
        driver.Quit();
    }
}
```

- **Login Method:** Since all Pages require to be logged in it is nice to have a reusable Login method in the BaseTest class. It simplifies the login process by providing a single method that can be called from any test that requires a user to be logged in.

```
6 references | 6/6 passing
protected void Login(string username, string password)
{
    driver.Navigate().GoToUrl("https://www.saucedemo.com/");
    var loginPage = new LoginPage(driver);
    loginPage.InputUsername(username);
    loginPage.InputPassword(password);
    loginPage.ClickLoginButton();
}
```

3. LoginPage

The **LoginPage** class represents the login page of the Swag Labs application. It inherits from the **BasePage** class, which provides common functionalities such as finding elements, clicking elements, and typing into input fields. Here's a detailed breakdown of what the **LoginPage** class should contain:

- **Inheritance from BasePage:**
  - The **LoginPage** has access to the common methods defined in **BasePage**, such as **FindElement**, **Click**, **Type**, and **GetText**.
- **Constructor**
  - The **constructor of LoginPage** calls the constructor of **BasePage** using **: base(driver).**

```
2 references | 1/1 passing
public LoginPage(IWebDriver driver) : base(driver) { }
```

- **Element Locators:**
  - **usernameField**, **passwordField**, **loginButton**, and **errorMessage** are defined using the By class.
  - These locators identify the elements on the login page by their attributes (ID or CSS Selector).

```
private readonly By usernameField = By.Id("user-name");
private readonly By passwordField = By.Id("password");
private readonly By loginButton = By.Id("login-button");
private readonly By errorMessage = By.CssSelector(".error-message-container.error");
```

- **Methods:**
  - **`InputUsername`:** This method types the given username into the username field. It uses the Type method inherited from BasePage.

```
2 references | 1/1 passing
public void InputUsername(string username)
{
    Type(usernameField, username);
}
```

  - **`InputPassword`:** This method types the given password into the password field. It uses the Type method inherited from BasePage.

```
2 references | 1/1 passing
public void InputPassword(string password)
{
    Type(passwordField, password);
}
```

  - **`ClickLoginButton`:** This method clicks the login button. It uses the Click method inherited from BasePage.

```
2 references | 1/1 passing
public void ClickLoginButton()
{
    Click(loginButton);
}
```

  - **`GetErrorMessage`:** This method retrieves the text of the error message displayed when login fails. It uses the GetText method inherited from BasePage.

4. LoginPageTests

With the **`BaseTest`** class and Login method in place, we can proceed to write the tests for the **`LoginPage`**.

- The **`LoginPageTest`**s class inherits from the **`BaseTest`** class, gaining access to the **`setup`**, **`teardown`**, and **`Login`** method.

```
[TestFixture]
    0 references
    public class LoginPageTests : BaseTest
```

- Write a test to log in with valid credentials (**`TestLoginWithValidCredentials`**).
- Write a test to log in with invalid credentials (**`TestLoginWithInvalidCredentials`**).
- Write a test to log in with locked out user (**`TestLoginWithLockedOutUser`**).

Note: Use Login method from the BaseTest class for each of the tests.

5. InventoryPage

The **`InventoryPage`** class extends the **`BasePage`** and is responsible for **interacting with the inventory page** of the Swag Labs application. Here's a detailed explanation of what this class should contain:

- **Class Definition:** **`InventoryPage`** inherits from **`BasePage`**, gaining access to common methods and properties for interacting with web elements.
- **Locators:**

- CSS selector to locate the cart link element.
- Class name selector to locate the title of the products section.
- CSS selector to locate all inventory items on the page.

```
private readonly By cartLink = By.CssSelector(".shopping_cart_link");
private readonly By productsTitle = By.ClassName("title");
private readonly By inventoryItems = By.CssSelector(".inventory_item");
```

- **Constructor:** The constructor accepts an IWebDriver instance and passes it to the base class constructor to initialize the driver.
- **Methods:**
  - **`AddToCartByIndex`:** Allows adding an item to the cart based on its index.

```
2 references | 2/2 passing
public void AddToCartByIndex(int itemIndex)
{
    var itemAddToCartButton = By.CssSelector($".inventory_item:nth-child({itemIndex + 1}) .btn_inventory");
    Click(itemAddToCartButton);
}
```

  - **`AddToCartByName`:** Allows adding an item to the cart based on the product name.

```
1 reference | 1/1 passing
public void AddToCartByName(string productName)
{
    var itemAddToCartButton = By.XPath($"//div[text()='{productName}']" +
        $"/ancestor::div[@class='inventory_item']//button[contains(@class, 'btn_inventory')]");
    Click(itemAddToCartButton);
}
```

  - **`ClickCartLink`:** Uses the Click method from the BasePage class to click the cart link.

```
2 references | 2/2 passing
public void ClickCartLink()
{
    Click(cartLink);
}
```

  - **`IsInventoryDisplayed`:** Uses the FindElements method from the BasePage class to check if any inventory items are displayed.

```
1 reference | 1/1 passing
public bool IsInventoryDisplayed()
{
    return FindElements(inventoryItems).Any();
}
```

  - **`IsPageLoaded`:** Uses the GetText method from the BasePage class to verify that the page title is "Products" and the URL contains "inventory.html".

```
1 reference | 1/1 passing
public bool IsPageLoaded()
{
    return GetText(productsTitle) == "Products" && driver.Url.Contains("inventory.html");
}
```

6. LoginPageTests

With the **BaseTest** class and **InventoryPage** in place, we can proceed to write the tests for the **InventoryPage**.

The **InventoryPageTests** class inherits from the BaseTest class, gaining access to the setup, teardown, and Login method.

Tests to be written:

- **TestInventoryDisplay():** Ensures that the inventory items are displayed.
- **TestAddToCartByIndex():** Adds an item to the cart by its index.
- **TestAddToCartByName():** Adds an item to the cart by its name.
- **TestPageTitle():** Ensures that the Inventory Page is loaded correctly.

7. CartPage

The **CartPage** class represents the shopping cart page in the Swag Labs application. It inherits from the **BasePage** class and provides methods to interact with elements on the cart page, such as verifying the presence of items in the cart.

- **Locators:**
  - Locator for items in the cart.
  - Locator for the checkout button.

```
private readonly By cartItem = By.CssSelector(".cart_item");
private readonly By checkoutButton = By.CssSelector("#checkout");
```

- **Constructor:** Accepts an IWebDriver instance and passes it to the base class constructor to initialize the driver.
- **Methods:**
  - **IsCartItemDisplayed():** Checks if an item is displayed in the cart.

```
3 references | 3/3 passing
public bool IsCartItemDisplayed()
{
    return FindElement(cartItem).Displayed;
}
```

  - **ClickCheckout():** Clicks the checkout button.

```
1 reference | 1/1 passing
public void ClickCheckout()
{
    Click(checkoutButton);
}
```

8. CartPageTests

The **SetUp** method logs in with valid credentials, adds an item to the cart, and navigates to the cart page. This ensures that each test starts with an item already in the cart.

Tests to be written:

- **TestCartItemDisplayed():** verifies that the added item is displayed in the cart.

- **`TestClickCheckout():`** clicks the checkout button and verifies that the user is redirected to the checkout page by checking the URL.

9. CheckoutPage

The **CheckoutPage** class extends the **BasePage** class, inheriting common functionalities for interacting with web elements. This class specifically handles the elements and actions on the checkout page of the Swag Labs application.

Key Functionalities:

- **Field Locators:**
  - Locator for the first name input field.
  - Locator for the last name input field.
  - Locator for the postal code input field.
  - Locator for the continue button.
  - Locator for the finish button.
  - Locator for the completion header.

```
private readonly By firstNameField = By.Id("first-name");
private readonly By lastNameField = By.Id("last-name");
private readonly By postalCodeField = By.Id("postal-code");
private readonly By continueButton = By.CssSelector(".cart_button");
private readonly By finishButton = By.Id("finish");
private readonly By completeHeader = By.CssSelector(".complete-header");
```

- **Constructor**: Initializes the **CheckoutPage** with the provided WebDriver instance and calls the **BasePage** constructor.
- **Methods**:
  - **`EnterFirstName`**, **`EnterLastName`**, and **`EnterPostalCode`**: These methods use the inherited Type method from **BasePage** to enter text into the respective fields.

```
1 reference | 1/1 passing
public void EnterFirstName(string firstName)
{
    Type(firstNameField, firstName);
}

1 reference | 1/1 passing
public void EnterLastName(string lastName)
{
    Type(lastNameField, lastName);
}

1 reference | 1/1 passing
public void EnterPostalCode(string postalCode)
{
    Type(postalCodeField, postalCode);
}
```

  - **`ClickContinue`** and **`ClickFinish`**: These methods use the inherited Click method from **BasePage** to interact with the buttons.

```
1 reference | 1/1 passing
public void ClickContinue()
{
    Click(continueButton);
}

0 references
public void ClickFinish()
{
    Click(finishButton);
}
```

- **IsPageLoaded**: This method checks the URL to determine if the current page is a checkout page.

```
0 references
public bool IsPageLoaded()
{
    return driver.Url.Contains("checkout-step-one.html") ||
    driver.Url.Contains("checkout-step-two.html");
}
```

- **IsCheckoutComplete**: This method verifies that the order completion message is displayed by checking the text of the completion header element.

```
0 references
public bool IsCheckoutComplete()
{
    return GetText(completeHeader) == "Thank you for your order!";
}
```

10. CheckoutPageTests

The **CheckoutPageTests** class contains tests for the checkout process in the Swag Labs application. This class inherits from **the BaseTest** class, allowing it to use the setup and login methods defined there. Here's a detailed explanation of the **CheckoutPageTest**s class and its methods:

- **SetUp Method:**
  - **Login:** Logs in using the **Login** method from the **BaseTest** class with valid credentials.
  - **Add Item to Cart:** Uses **AddToCartByIndex** method to add the first item to the cart.
  - **Navigate to Cart Page:** Clicks the cart link to go to the cart page.
  - **Proceed to Checkout:** Clicks the checkout button on the cart page to navigate to the checkout page.

Tests to be written:

- **TestCheckoutPageLoaded():** Verifies that the checkout page is loaded correctly.
- **TestContinueToNextStep():** Verifies that the user can continue to the next step in the checkout process.
- **TestCompleteOrder():** Verifies that the user can complete an order.

11. HiddenMenuPage

- **Field Locators:**
  - Locator for the menu button using CSS selector.
  - Locator for the logout button using its ID.

```
private readonly By menuButton = By.CssSelector(".bm-burger-button");
private readonly By logoutButton = By.Id("logout_sidebar_link");
```

- Locator for the container for the menu using its class name.
- **Constructor:** The constructor initializes the driver using the base class constructor.
- **Methods:**
  - **`ClickMenuButton`**: Uses the Click method from the BasePage to click the menu button.
  - **`ClickLogoutButton`**: Uses the Click method from the BasePage to click the logout button.
  - **`IsMenuOpen`**: Checks if the logout button is visible to determine if the menu is open.

```
2 references | 2/2 passing
public void ClickMenuButton()
{
    Click(menuButton);
}

1 reference | 1/1 passing
public void ClickLogoutButton()
{
    Click(logoutButton);
}

1 reference | 1/1 passing
public bool IsMenuOpen()
{
    return FindElement(logoutButton).Displayed;
}
```

12. HiddenMenuPageTests

Tests to be written:

- **`TestOpenMenu():`** Verifies that the hidden menu opens correctly.
- **`TestLogout():`** Verifies that the user can log out from the hidden menu.

13. Run Tests