Automation Test Framework and Test Scripts for WordPress Website

This project includes a sample of an automation test framework and some automated test scripts using C# and Selenium Webdriver (v3.141.0). These test scripts are to verify some basic functions of the Wordpress website (v5.2.2) which is an online, opensource website for creating websites or blogs.

For this project, Wordpress, local server (XAMPP) and MySQL database are installed in my computer.

Table of Contents

[1. Test Environment 2](#_Toc36039803)

[1.1. MySQL database 2](#_Toc36039804)

[1.2. Wordpress website 3](#_Toc36039805)

[1.3. A local server 4](#_Toc36039806)

[2. Some Basic Test Scripts 4](#_Toc36039807)

[2.1. Utilities: WordpressTest 4](#_Toc36039808)

[2.2. Login Tests: Admin\_User\_Can\_Login 4](#_Toc36039809)

[2.3. Create Post Tests: Can\_Create\_A\_Basic\_Post 5](#_Toc36039810)

[2.4. Page Tests: Can\_Edit\_A\_Page 6](#_Toc36039811)

[2.5. All Post Tests: Added\_Post\_Show\_Up 6](#_Toc36039812)

[2.6. All Post Tests: Can\_Search\_Posts 7](#_Toc36039813)

[3. Framework 8](#_Toc36039814)

[3.1. Selenium: Driver 8](#_Toc36039815)

[3.2. Workflows: PostCreator 8](#_Toc36039816)

[3.3. Pages: LoginPage 10](#_Toc36039817)

[3.4. Pages: NewPostPage 11](#_Toc36039818)

[3.5. Pages: ListPostPage 12](#_Toc36039819)

[3.2. Nagivation: LeftNavigation 14](#_Toc36039820)

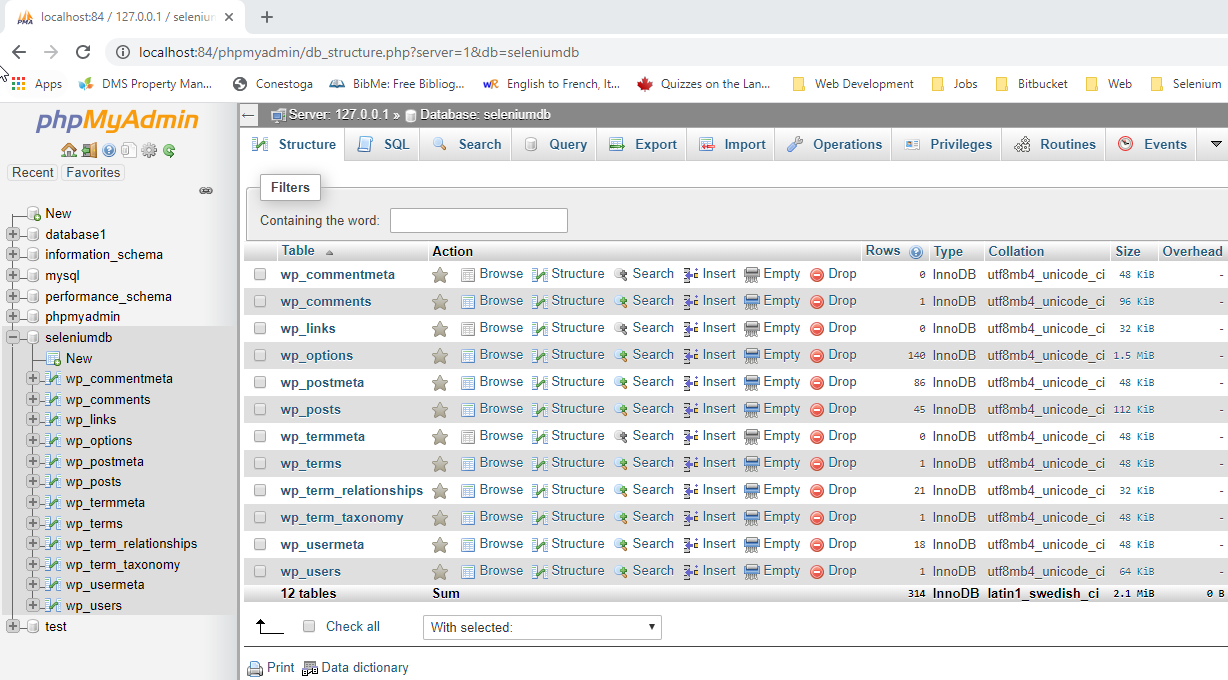
[3.2. Nagivation: MenuSelector 15](#_Toc36039821)

[3.4. Pages: DashboardPage 15](#_Toc36039822)

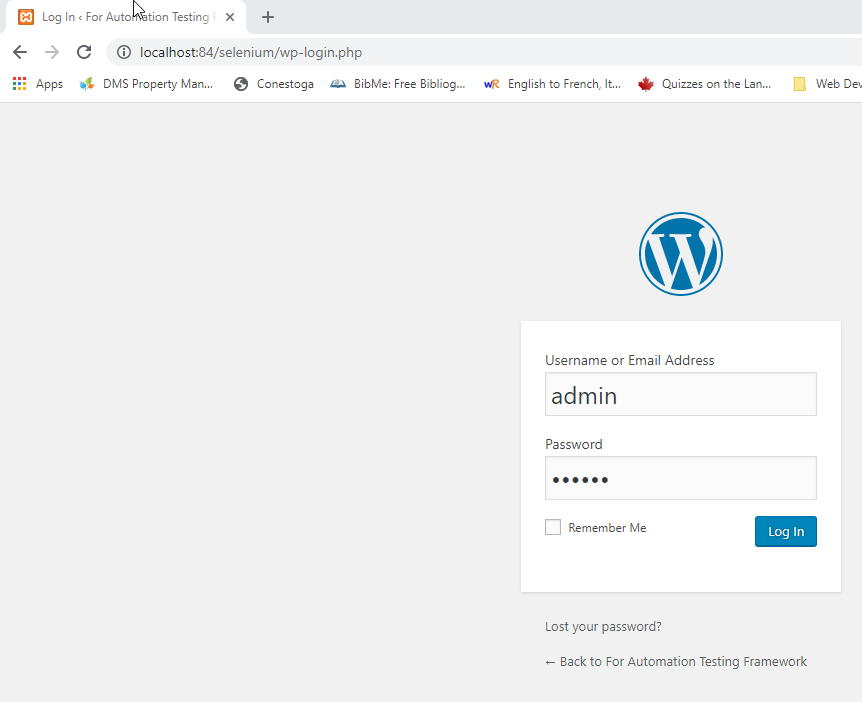
[3.5. Pages: PostPage 15](#_Toc36039823)

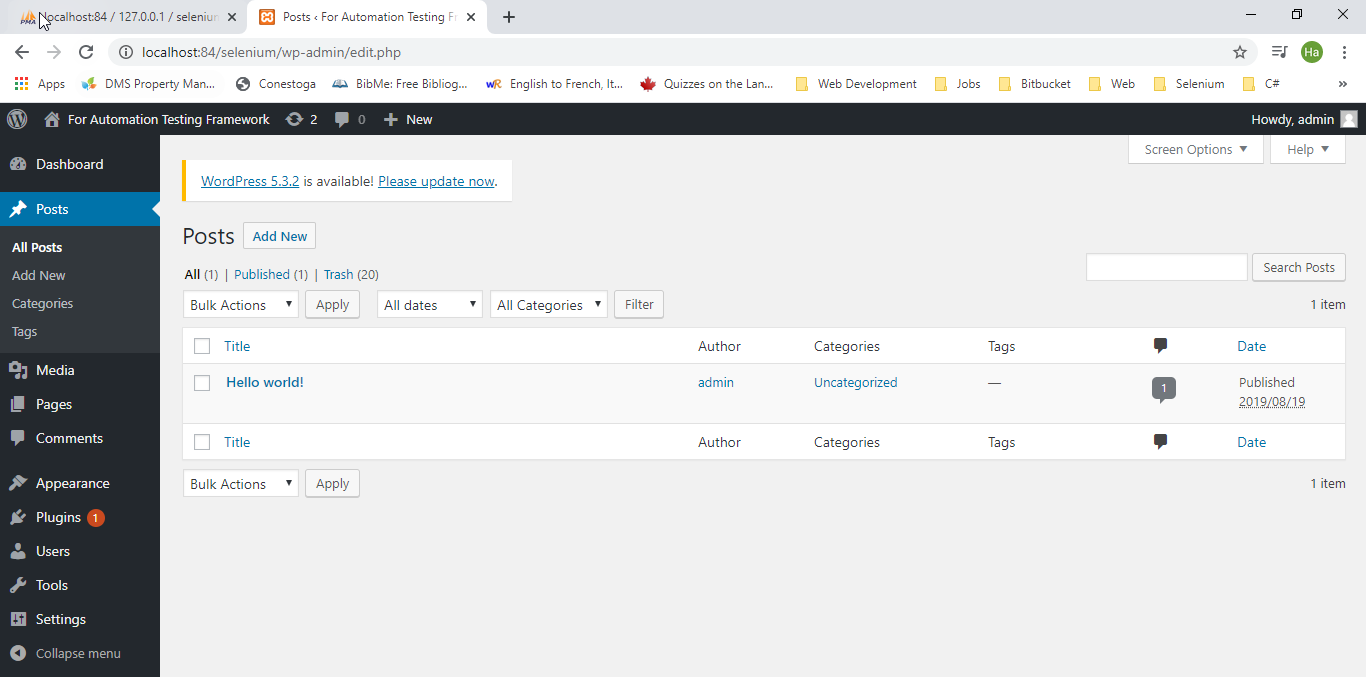
# 1. Test Environment

## 1.1. MySQL database

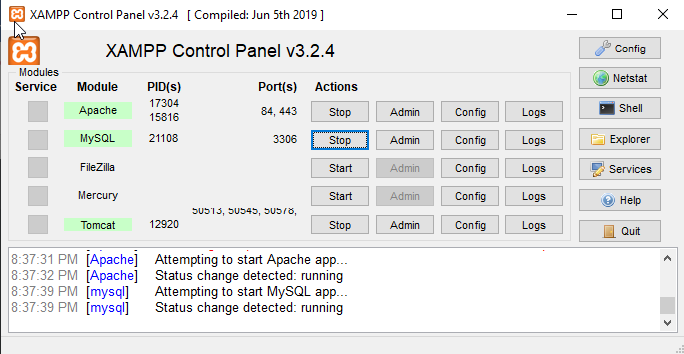


## 1.2. Wordpress website





## 1.3. A local server



# 2. Some Basic Test Scripts

## 2.1. Utilities: WordpressTest

using Microsoft.VisualStudio.TestTools.UnitTesting;

using WordpressAutomation;

using WordpressAutomation.Workflows;

namespace WordpressTests

{

public class WordpressTest

{

[TestInitialize]

public void Init()

{

Driver.Initialize();

PostCreator.Initialize();

LoginPage.GoTo();

LoginPage.LoginAs("admin").WithPassword("hatran").Login();

}

[TestCleanup]

public void CleanUp()

{

PostCreator.CleanUp();

Driver.Close();

}

}

}

## 2.2. Login Tests: Admin\_User\_Can\_Login

using OpenQA.Selenium;

using OpenQA.Selenium.Support.UI;

using System;

namespace WordpressTests

{

[TestClass]

public class LoginTests : WordpressTest

{

[TestMethod]

public void Admin\_User\_Can\_Login()

{

#region Manual test case

//Go to login page

//Login as “admin” user with correct password “ha tran”. Click Login button

//Verify that user can login, the Dashboard page is displayed

#endregion Manual test case

Assert.IsTrue(DashboardPage.IsAt, "Failed to login.");

}

}

}

## 2.3. Create Post Tests: Can\_Create\_A\_Basic\_Post

using Microsoft.VisualStudio.TestTools.UnitTesting;

using WordpressAutomation;

namespace WordpressTests

{

[TestClass]

public class CreatePostTests : WordpressTest

{

[TestMethod]

public void Can\_Create\_A\_Basic\_Post()

{

#region Manual test case

//Go to login page

//Login as “admin” user with correct password “ha tran”

//Click Login button

//Click Posts menu on the left navigation panel

//Click Add New button

//Enter the post title “This is the post title”

//Enter the post body “Hi, This is the post body”

//Click “Publish…” button

//Click “Publish” button

//Click “View Post” button

//Verify that the post is created by checking title “This is the post title”

#endregion Manual test case

NewPostPage.GoTo();

NewPostPage.CreatePost("This is the post title")

.WithBody("Hi, This is the post body")

.Publish();

NewPostPage.GoToNewPost();

Assert.AreEqual(PostPage.NewPostTitle, "This is the post title", "Title did not match new post.");

}

}

}

## 2.4. Page Tests: Can\_Edit\_A\_Page

using Microsoft.VisualStudio.TestTools.UnitTesting;

using WordpressAutomation;

namespace WordpressTests

{

[TestClass]

public class PageTests : WordpressTest

{

[TestMethod]

public void Can\_Edit\_A\_Page()

{

#region Manual test case

//Go to login page

//Login as “admin” user with correct password “ha tran”

//Click Login button

//Hover mouse on the “Pages” menu on the left navigation panel

//Click “All Pages” menu

//Click on the page post having title “Sample Page”

//Verify that the page post is in edit mode by checking

visibility of the “View Page” button

//Verify that the page title is “Sample Page”

#endregion

ListPostsPage.GoTo(PostType.Page);

ListPostsPage.SelectPost("Sample Page");

Assert.IsTrue(NewPostPage.IsInEditMode(), "Wasn't in edit mode");

Assert.AreEqual("Sample Page", NewPostPage.Title, "Title didn't match");

}

}

}

## 2.5. All Post Tests: Added\_Post\_Show\_Up

namespace WordpressTests

{

[TestClass]

public class AllPostsTests : WordpressTest

{

[TestMethod]

public void Added\_Post\_Show\_Up()

{

#region Manual test case

//Go to login page

//Login as “admin” user with correct password “ha tran”

//Click Login button

//Hover mouse on the “Posts” menu on the left navigation panel

//Click “All Posts” menu

//Observe and record posts count at the top right corner of the posts list

//Add a new post and publish it

//Go to All Posts list

//Verify that the posts count increase by 1

//Verify that the added post exists in the list

#endregion Manual test case

//Go to Posts, get post count, store

ListPostsPage.GoTo(PostType.Posts);

ListPostsPage.StoreCount();

//Add a new post

PostCreator.CreatePost();

//Go to Posts, get new post count

ListPostsPage.GoTo(PostType.Posts);

Assert.AreEqual(ListPostsPage.PreviousPostsCount + 1, ListPostsPage.CurrentPostsCount, "Count of posts didn't increase");

//Check for added post

Assert.IsTrue(ListPostsPage.DoesPostExistWithTitle(PostCreator.PreviousTitle));

//Trash post(clean up)

ListPostsPage.TrashPost(PostCreator.PreviousTitle);

Assert.AreEqual(ListPostsPage.PreviousPostsCount, ListPostsPage.CurrentPostsCount, "Couldn't trash post");

}

}

}

## 2.6. All Post Tests: Can\_Search\_Posts

public class AllPostsTests : WordpressTest

{

[TestMethod]

public void Can\_Search\_Posts()

{

#region Manual test case

//Go to login page

//Login as “admin” user with correct password “ha tran”

//Click Login button

//Hover mouse on the “Posts” menu on the left navigation panel

//Click “All Posts” menu

//Enter a search string that matches to at least a post title to the

Search text box

//Click “Search Posts” button

//Verify that the matching post titles show up in the list

#endregion Manual test case

//Create a new post

PostCreator.CreatePost();

//Go to list posts

ListPostsPage.GoTo(PostType.Posts);

//Search for posts

ListPostsPage.SearchForPosts(PostCreator.PreviousTitle);

//Check that post show up in results

Assert.IsTrue(ListPostsPage.DoesPostExistWithTitle(PostCreator.PreviousTitle));

}

}

}

# 3. Framework

## 3.1. Selenium: Driver

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.Firefox;

using System.IO;

namespace WordpressAutomation

{

public class Driver

{

public static IWebDriver Instance { get; set; }

public static string BaseAddress

{

get { return "http://localhost:84/selenium/"; }

}

static string projectPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;

static string relativeDriverPath = @"..\WebBrowserDrivers";

public static void Initialize()

{

//Instance = new FirefoxDriver(Path.GetFullPath(Path.Combine(projectPath, relativeDriverPath);

Instance = new ChromeDriver(Path.GetFullPath(Path.Combine(projectPath, relativeDriverPath)));

}

public static void Close()

{

Instance.Close();

}

}

}

## 3.2. Workflows: PostCreator

using System;

using System.Text;

namespace WordpressAutomation.Workflows

{

public class PostCreator

{

public static string PreviousTitle { get; set; }

public static string PreviousBody { get; set; }

public static bool createdAPost

{

get {return !String.IsNullOrEmpty(PreviousTitle);}

}

public static void CreatePost()

{

NewPostPage.GoTo();

PreviousTitle = CreateTitle();

PreviousBody = CreateBody();

NewPostPage.CreatePost(PreviousTitle).WithBody(PreviousBody).Publish();

}

public static void Initialize()

{

PreviousTitle = null;

PreviousBody = null;

}

public static void CleanUp()

{

if (createdAPost)

{

TrashPost();

}

}

private static void TrashPost()

{

ListPostsPage.TrashPost(PreviousTitle);

Initialize();

}

private static string CreateBody()

{

return CreateRandomString() + ", body";

}

private static string CreateTitle()

{

return CreateRandomString() + ", title";

}

private static string CreateRandomString()

{

var s = new StringBuilder();

var random = new Random();

var cycles = random.Next(5 + 1);

for (int i = 0; i < cycles; i++)

{

s.Append(Words[random.Next(Words.Length)]);

s.Append(" ");

s.Append(Articles[random.Next(Articles.Length)]);

s.Append(" ");

}

return s.ToString();

}

private static string[] Words = new[]

{

"boy", "cat", "dog", "rabbit", "baseball", "throw", "find", "mustard"

};

private static string[] Articles = new[]

{

"the", "and", "and", "a", "of", "to", "it", "as"

};

}

}

## 3.3. Pages: LoginPage

using OpenQA.Selenium;

using OpenQA.Selenium.Support.UI;

using System;

namespace WordpressAutomation

{

public class LoginPage

{

public static void GoTo()

{

Driver.Instance.Navigate().GoToUrl(Driver.BaseAddress + "wp-login.php");

var wait = new WebDriverWait(Driver.Instance, TimeSpan.FromSeconds(10));

wait.Until(d => d.SwitchTo().ActiveElement().GetAttribute("id") == "user\_login");

}

public static LoginCommand LoginAs(string userName)

{

return new LoginCommand(userName);

}

}

public class LoginCommand

{

private string userName;

private string password;

public LoginCommand(string userName)

{

this.userName = userName;

}

public LoginCommand WithPassword(string password)

{

this.password = password;

return this;

}

public void Login()

{

var userNameInput = Driver.Instance.FindElement(By.Id("user\_login"));

userNameInput.SendKeys(userName);

var passwordInput = Driver.Instance.FindElement(By.Id("user\_pass"));

passwordInput.SendKeys(password);

var loginButton = Driver.Instance.FindElement(By.Id("wp-submit"));

loginButton.Click();

}

}

}

## 3.4. Pages: NewPostPage

using OpenQA.Selenium;

using OpenQA.Selenium.Interactions;

using OpenQA.Selenium.Support.UI;

using System;

using System.Threading;

namespace WordpressAutomation

{

public class NewPostPage

{

public static void GoTo()

{

LeftNavigation.Posts.AddNew.Select();

}

public static CreatePostCommand CreatePost(string title)

{

return new CreatePostCommand(title);

}

public static void GoToNewPost()

{

var wait = new WebDriverWait(Driver.Instance, TimeSpan.FromSeconds(30));

wait.Until(d => d.SwitchTo().ActiveElement().FindElement(By.XPath("//a[@class = 'components-button is-button is-default' and (text() = 'View Post')]")))

.Click();

}

public static bool IsInEditMode()

{

var viewPageLink = Driver.Instance.FindElement(By.XPath("//a[@class = 'ab-item' and text() = 'View Page']"));

bool isViewPageLinkVisible = viewPageLink.Displayed;

if (isViewPageLinkVisible)

return isViewPageLinkVisible;

return false;

}

public static string Title

{

get

{

var title = Driver.Instance.FindElement((By.Id("post-title-0")));

if (title != null)

return title.GetAttribute("value");

return string.Empty;

}

}

}

public class CreatePostCommand

{

private string title;

private string body;

public CreatePostCommand(string title)

{

this.title = title;

}

public CreatePostCommand WithBody(string body)

{

this.body = body;

return this;

}

public void Publish()

{

Thread.Sleep(3000);

bool isDisableTipsDisplayed = Driver.Instance.FindElement(By.XPath("//button[@aria-label = 'Disable tips']")).Displayed;

if (isDisableTipsDisplayed)

{

Driver.Instance.FindElement(By.XPath("//button[@aria-label = 'Disable tips']")).Click();

}

Driver.Instance.FindElement(By.Id("post-title-0")).SendKeys(title);

IWebElement postBody = Driver.Instance.FindElement(By.XPath("//div[@class='editor-block-list\_\_layout block-editor-block-list\_\_layout']"));

Actions actions = new Actions(Driver.Instance);

actions.MoveToElement(postBody);

actions.Click();

actions.SendKeys(body);

actions.Build().Perform();

Driver.Instance.FindElement(By.XPath("//button[text() = 'Publish…']")).Click();

Driver.Instance.FindElement(By.XPath("//button[text() = 'Publish']")).Click();

}

}

}

## 3.5. Pages: ListPostPage

using OpenQA.Selenium;

using OpenQA.Selenium.Interactions;

using OpenQA.Selenium.Support.UI;

using System;

using System.Collections.ObjectModel;

using System.Linq;

namespace WordpressAutomation

{

public class ListPostsPage

{

private static int lastCount;

public static int PreviousPostsCount { get { return lastCount; } }

public static int CurrentPostsCount { get { return GetPostsCount(); } }

public static void GoTo(PostType postType)

{

switch (postType)

{

case PostType.Page:

LeftNavigation.Pages.AllPages.Select();

break;

case PostType.Posts:

LeftNavigation.Posts.AllPosts.Select();

break;

}

}

public static void SelectPost(string title)

{

var wait = new WebDriverWait(Driver.Instance, TimeSpan.FromSeconds(30));

var postPageList = wait.Until(d => d.SwitchTo().ActiveElement().FindElement(By.Id("the-list")));

var postLink = Driver.Instance.FindElement(By.LinkText("Sample Page"));

postLink.Click();

}

public static void StoreCount()

{

lastCount = GetPostsCount();

}

private static int GetPostsCount()

{

var wait = new WebDriverWait(Driver.Instance, TimeSpan.FromSeconds(40));

var countText = wait.Until(d => d.SwitchTo().ActiveElement().FindElement(By.ClassName("displaying-num"))).Text;

return int.Parse(countText.Split(' ')[0]);

}

public static bool DoesPostExistWithTitle(string title)

{

//find any element that matches the title

return Driver.Instance.FindElements(By.LinkText(title)).Any();

}

public static void TrashPost(string title)

{

var rows = Driver.Instance.FindElements(By.TagName("tr"));//Find all the rows

foreach (var row in rows)

{

ReadOnlyCollection<IWebElement> links = null;

links = row.FindElements(By.LinkText(title));//look through each row and try to find any element that matches the title

if(links.Count > 0)

{

Actions action = new Actions(Driver.Instance);//use Actions class for hover over action

action.MoveToElement(links[0]);

action.Perform();

row.FindElement(By.ClassName("submitdelete")).Click();

return;//return down to there because we found the one that we were looking for

}

}

}

public static void SearchForPosts(string searchString)

{

var wait = new WebDriverWait(Driver.Instance, TimeSpan.FromSeconds(30));

var searchInput = wait.Until(d => d.SwitchTo().ActiveElement().FindElement(By.Id("post-search-input")));

searchInput.SendKeys(searchString);

var searchPostsButton = Driver.Instance.FindElement(By.Id("search-submit"));

searchPostsButton.Click();

}

}

public enum PostType

{

Page,

Posts

}

}

## 3.2. Nagivation: LeftNavigation

namespace WordpressAutomation

{

public class LeftNavigation

{

public class Posts

{

public class AllPosts

{

public static void Select()

{

MenuSelector.Select("//div[@class = 'wp-menu-name' and text() = 'Posts']", "//a[@class = 'wp-first-item' and text() = 'All Posts']");

}

}

public class AddNew

{

public static void Select()

{

MenuSelector.Select("//div[@class='wp-menu-name' and text() = 'Posts']", "//a[text() = 'Add New']");

}

}

}

public class Pages

{

public class AllPages

{

public static void Select()

{

MenuSelector.Select("//div[@class = 'wp-menu-name' and text() = 'Pages']", "//a[@class = 'wp-first-item' and text() = 'All Pages']");

}

}

}

}

}

## 3.2. Nagivation: MenuSelector

using OpenQA.Selenium;

using OpenQA.Selenium.Interactions;

namespace WordpressAutomation

{

public class MenuSelector

{

public static void Select(string firstLevelMenu, string secondLevelMenu)

{

//locate element by mouse hovering

var firstMenuLevel = Driver.Instance.FindElement(By.XPath(firstLevelMenu));

Actions action = new Actions(Driver.Instance);

action.MoveToElement(firstMenuLevel).Perform();

var secondMenuLevel = Driver.Instance.FindElement(By.XPath(secondLevelMenu));

action.MoveToElement(secondMenuLevel).Click().Perform();

}

}

}

## 3.4. Pages: DashboardPage

using OpenQA.Selenium;

namespace WordpressAutomation

{

public class DashboardPage

{

public static bool IsAt

{

get

{

var h1s = Driver.Instance.FindElements(By.TagName("h1"));

if (h1s.Count > 0)

return h1s[0].Text == "Dashboard";

return false;

}

}

}

}

## 3.5. Pages: PostPage

using OpenQA.Selenium;

namespace WordpressAutomation

{

public class PostPage

{

public static string Title

{

get

{

var title = Driver.Instance.FindElement(By.Id("post-title-0"));

if (title != null)

return title.Text;

return string.Empty;

}

}

public static object NewPostTitle

{

get

{

var newPostTitle = Driver.Instance.FindElement(By.XPath("//h1[text() = 'This is the post title']"));

if (newPostTitle != null)

return newPostTitle.Text;

return string.Empty;

}

}

public static string Body

{

get

{

var body = Driver.Instance.FindElement(By.XPath("//div[@class='components-autocomplete']/p[@aria-label='Paragraph block']"));

if (body != null)

return body.Text;

return string.Empty;

}

}

public static void ViewAPost()

{

var postList = Driver.Instance.FindElement(By.Id("the-list"));

var postLink = postList.FindElements(By.TagName("a"))[0];

postLink.Click();

}

public static void GoTo()

{

var menuPosts = Driver.Instance.FindElement(By.XPath("//div[@class='wp-menu-name' and text() = 'Posts']"));

menuPosts.Click();

}

}

}