How to WPF Apps Automated with Appium

Every test automation program requires a framework that it can rely on. One such framework is Appium, which allows you to create test scripts for Windows Presentation Foundation (WPF) applications.

## Install WinAppDriver

* Download Windows Application Driver installer from [WinAppDriver](https://github.com/Microsoft/WinAppDriver/releases).
* Run the installer on a Windows 10 machine where your application under test is installed and will be tested
* Enable [Developer Mode](https://learn.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development) in Windows settings

## Write an Automation Script

You can begin writing an Automation script once WinAppDriver has been successfully installed.

### Creating a Test Project

* Open Microsoft Visual Studio
* Create the test project and solution. i.e., Select **New Project** > **Templates** > **Visual C#** > **Test** > **Unit Test Project**
* Once created, select **Project** > **Manage NuGet Packages**... > **Browse** and search for **Appium.WebDriver**
* Install the Appium.WebDriver NuGet packages for the test project

### Adding a Base Class and Tests Session

1. Create a new class called DiagramSession where all the Appium settings will be configured.
2. WinAppDriver need to be launched (in **administrator** mode) before running Appium test-scripts. You can manually launch the **WinAppDriver.exe** from the installation directory (E.g. C:\Program Files (x86)\Windows Application Driver) or you can include below code in you test project.

|  |
| --- |
| private static void StartWinAppDriver()  {  if (winAppDriverProcess == null)  {  var info = new ProcessStartInfo(winAppDriverPath)  {  UseShellExecute = true,  Verb = "runas",  WindowStyle = ProcessWindowStyle.Hidden  };  winAppDriverProcess = Process.Start(info);  }  } |

1. To test a WPF app, specify the **full executable path** for the app under test in the **app** capabilities entry when creating a new session.

|  |
| --- |
| private static Process winAppDriverProcess;  private const string winAppDriverPath = @"C:\Program Files (x86)\Windows Application Driver\WinAppDriver.exe";  private const string winAppDriverUrl = "http://127.0.0.1:4723";  private const string diagramAppId = @"D:\WinAppDriverTesting\DiagramApplication\bin\Debug\DiagramApplication.exe";  private const string deviceName = "WindowsPC";  private const int waitForAppLaunch = 2;  // Create a new session to launch Diagram application  var appiumOptions = new AppiumOptions();  appiumOptions.AddAdditionalCapability("app", diagramAppId);  appiumOptions.AddAdditionalCapability("deviceName", deviceName);  appiumOptions.AddAdditionalCapability("ms:waitForAppLaunch", waitForAppLaunch);  DiagramAppSession = new WindowsDriver<WindowsElement>(new Uri(winAppDriverUrl), appiumOptions);  Assert.IsNotNull(DiagramAppSession);  Assert.IsNotNull(DiagramAppSession.SessionId);  // Set implicit timeout to 1.5 seconds to make element search to retry every 500 ms for at most three times  DiagramAppSession.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(1.5); |

1. Create a new class called ScenarioDragandDrop to write a test scripts to related to drag and drop functionality.

|  |
| --- |
| [TestClass]  public class ScenarioDragandDrop: DiagramSession  {  private static WindowsElement diagram;    // ClassInitialize - called only once and any initialization operations it performs will apply to the entire TestMethod  [ClassInitialize]  public static void ClassInitialize(TestContext context)  {  Setup(context);  diagram = DiagramAppSession.FindElementByAccessibilityId("diagram");  Assert.IsNotNull(diagram);  }  // ClassCleanup - called once all the TestMethod from the class have been executed  [ClassCleanup]  public static void ClassCleanup()  {  diagram = null;  TearDown();  }  } |

1. Then you can add TestMethod to write scripts

|  |
| --- |
| [TestMethod]  public void DropNodeFromStencilToDiagram()  {  Thread.Sleep(TimeSpan.FromSeconds(.5));  var actions = new Actions(DiagramAppSession);  var stencil = DiagramAppSession.FindElementByAccessibilityId("stencil");  Assert.IsNotNull(stencil);  var rectangleNode = stencil.FindElement(new ByAccessibilityId("Rectangle"));  Assert.IsNotNull(rectangleNode);  // Select a node in stencil  actions.MoveToElement(rectangleNode).Click().Perform();  Thread.Sleep(TimeSpan.FromSeconds(.5));  // Click and hold the desired node for draging  actions.ClickAndHold().MoveByOffset(1, 0).MoveByOffset(0, 1).Perform();  Thread.Sleep(TimeSpan.FromSeconds(.5));  actions.MoveByOffset(diagram.Location.X + 200, diagram.Location.Y + 200).Release().Perform();  Thread.Sleep(TimeSpan.FromSeconds(10));  } |

**Note:** You can use [UI Recorder](https://github.com/microsoft/WinAppDriver/blob/master/Docs/UsingUIRecorder.md), standalone tool that allows users to creating automaton scripts by recording UI events performed by the user.

### FAQ & Documentation

You can find the documentation for WinAppDriver [here](https://github.com/microsoft/WinAppDriver#faq--documentation).