Winium for automating Desktop Applications

Winium is a Automation framework for Windows Applications. It is a open source tool from 2GIS. Winium supports

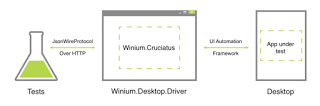
* Windows Desktop (WPF, WinForms) Apps
* Windows Store or Universal Apps for Windows Phone
* Windows Phone Silverlight Apps

But in this post we are going to discuss only about Winium for Desktop application automation.

We know that selenium supports only web applications. There are multiple other tools that we discussed in our previous blogs like AutoIT, Sikuli can be used to automate Windows based applications. But winium has the below advantages comparing to AutoIT and Sikuli.

1. As discussed in our previous blogs, to use Auto IT, We have to use some windows bridge to access the methods in it. But Winium comes as Java API and also it is implemented on JSONWire protocal that is used by selenium.
2. We can also write tests in any programming language that is supported by selenium.
3. It supports wide magnitude of applications comparing to AutoIT.

Technically, Winium.Desktop is an http client. It implements JSWP protocol and uses Cruciatus to work with UI elements. Essentially, this is an implementation of WebDriver for Windows-based desktop applications. It uses regular Selenium bindings with Winium. Desktop in order to test Windows-based desktop applications.

[](https://3.bp.blogspot.com/-9BTLEwbw27g/V2XoOW0aGMI/AAAAAAAAB1w/RAdHsPVR05cEUtmazARCWzh2IFcKXwmNQCKgB/s1600/winium.png)

Below are the features of Winium.

* Automates native windows desktop app(winforms, WPF, everyapp that uses MS accessibility)
* compatible with JSON wire protocal
* Full access to desktop and UI
* Protocal extensions for desktop specific elements
* We can write tests in any programming language which Selenium Web driver supports

Below are commands that are supported by Winium.

| **Command** | **Query** |
| --- | --- |
| NewSession | POST /session |
| FindElement | POST /session/:sessionId/element |
| FindChildElement | POST /session/:sessionId/element/:id/element |
| ClickElement | POST /session/:sessionId/element/:id/click |
| SendKeysToElement | POST /session/:sessionId/element/:id/value |
| GetElementText | GET /session/:sessionId/element/:id/text |
| GetElementAttribute | GET /session/:sessionId/element/:id/attribute/:name |
| Quit | DELETE /session/:sessionId |
| ClearElement | POST /session/:sessionId/element/:id/clear |
| Close | DELETE /session/:sessionId/window |
| ElementEquals | GET /session/:sessionId/element/:id/equals/:other |
| ExecuteScript | POST /session/:sessionId/execute |
| FindChildElements | POST /session/:sessionId/element/:id/elements |
| FindElements | POST /session/:sessionId/elements |
| GetActiveElement | POST /session/:sessionId/element/active |
| GetElementSize | GET /session/:sessionId/element/:id/size |
| ImplicitlyWait | POST /session/:sessionId/timeouts/implicit\_wait |
| IsElementDisplayed | GET /session/:sessionId/element/:id/displayed |
| IsElementEnabled | GET /session/:sessionId/element/:id/enabled |
| IsElementSelected | GET /session/:sessionId/element/:id/selected |
| MouseClick | POST /session/:sessionId/click |
| MouseDoubleClick | POST /session/:sessionId/doubleclick |
| MouseMoveTo | POST /session/:sessionId/moveto |
| Screenshot | GET /session/:sessionId/screenshot |
| SendKeysToActiveElement | POST /session/:sessionId/keys |
| Status | GET /status |
| SubmitElement | POST /session/:sessionId/element/:id/submit |

**Disadvantages:**

* It uses real mouse and key board events, i.e You can't run more than one session on same machine, or use mouse while tests are running.
* If the app runs in the background and remains running there, then it will not work appropriately.
* It is sometimes difficult to get the text from component that displays the output. Then you receive the error message:  NO Text property. The reason is probably that this type of component is defined as Image.

In this post we are going to learn about writing a text into notepad by using winium. Please follow the steps given below.  
  
**Step 1:**  
Download the below jars  
<http://mvnrepository.com/artifact/com.github.2gis.winium/winium-elements-desktop/0.1.0-1>  
<http://mvnrepository.com/artifact/com.github.2gis.winium/winium-elements-desktop/0.2.0-1>  
  
Along with these two jars you need to have selenium latest jar as well. You can get the latest selenium jar from the below path.  
<http://docs.seleniumhq.org/download/>  
  
**Step 2:**  
Download Winium.Desktop.Driver.exe file from <https://github.com/2gis/Winium.Desktop/releases> (Latest). Extract the zip file  
  
**Step 3:**  
Run the Winium.Desktop.Driver.exe file (by double clicking the Winium. Desktop.Driver.exe file). So that a Winium desktop server will be started and running at port 9999.  
  
**Step 4:**  
Create a sample Java project and add the downloaded jars into the class path of the project and also TestNG Library. Now create a class and use the below code.  
  
*package com.src.Nainappa.test;*  
*import java.io.IOException;*  
*import java.net.URL;*  
*import org.openqa.selenium.winium.DesktopOptions;*  
*import org.openqa.selenium.winium.WiniumDriver;*  
*import org.testng.annotations.Test;*  
 *public class NotepadTest {*  
 *@Test*  
 *public void test() throws IOException{*  
 *DesktopOptions options= new DesktopOptions();*  
 *options.setApplicationPath("C:\\WINDOWS\\system32\\notepad.exe");*  
 *try{*  
 *WiniumDriver driver=new WiniumDriver(new URL("http://localhost:9999"),options);*  
 *driver.findElementByClassName("Edit").sendKeys("This is sample test");*  
 *driver.close();*  
 *}*  
 *catch(Exception e){*  
 *System.out.println(e.getMessage());*  
 *}*  
 *}*  
*}*  
  
**Step 5:**  
Run this code as TestNG Method. It opens the notepad and enters the text on the note pad and also tries to close it.  
  
If anybody is wondering how to get to know the properties of the elements, we have multiple ways.  
  
1. Renorex offers free spy. We can download it from the below path and use it.  
   http://www.ranorex.com/test-automation-tools/ranorex-spy.html  
  
2. The other way is White tool. You can download the entire tool from the below path.  
   https://github.com/TestStack/White  
   Extract the Zip file and you will see a file called UISpy.exe. This can also be used for spying on the elements.