## Subject:

Computer-Aided Software Testing (CAST) using the Selenium Toolset

## Author:

Gerard Sczepura
Software Quality Assurance Analyst, Sr. Principal

## Date:

October 21, 2019

## Table of Contents

## Background

For the sake of clarity, in this document, the term Selenium is synonymous with Selenium 2 and Selenium WebDriver API all of which refer to the Selenium browser automation toolset. Selenium is the foundation for the construction of a test automation framework built around Microsoft Visual Studio / C#. The various framework components that were constructed to support automated testing of various IAS applications are described in this document.

Selenium provides a method for automating the playback of test cases in the form of scripts developed using C#. You could say that C# is the scripting language. Other languages are supported including Java, Perl, PHP, Python, and Ruby, but for our purposes here, the scripting language is C#

It should be obvious that automated script creation is a development activity. As a result, the tester's responsibility for test case design, test data creation, and test verification are not abrogated. All these manual test activities must still be considered when developing automated test scripts.

Again, Selenium is an API not an application. As a result, Selenium doesn't provide features normally found in commercial automated testing tools such as Reporting, Object Identification, and Object Repository.

Reference(s): https://www.seleniumhq.org/

## Prerequisites

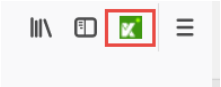
The Selenium browser automation toolset is not a test automation application as are popular commercial tool suites including IBM Rational Functional Tester (RFT), TestComplete, and Unified Functional Testing (UFT). As a result, the selenium toolset doesn't provide built-in object recognition, record and playback, and test reporting capabilities. The automated tester must utilize other tools and techniques which provide these capabilities. One other tool that was utilized for this effort was the Katalon Automation Recorder.

Katalon Automation Recorder is an extension for Chrome and Firefox browsers that provides record and playback capabilities which includes object recognition. Recorded scripts can be directly exported to C# (WebDriver + NUnit) format. Katalon was used to record browser actions that were later copied into Visual Studio. In this way, complex scripts can be recorded and debugged iteratively and incrementally.

### Installation

Firefox Browser:

1. From the Firefox Add-ons Manager, search for “Katalon” in the Search on addons.mozilla.org search box.
2. Select Katalon Recorder (Selenium tests generator) from the results list. Follow all installation prompts.
3. Successful installation is indicated by the Katalon icon appearing on the Firefox toolbar as shown in the following snippet:



Chrome Browser:

1. From the Chrome More Tools -> Extensions, click the Open Chrome web Store link in the lower left corner of the page.
2. Enter “katalon recorder” in the Search the store search box.
3. Click the Add to Chrome button corresponding to the Katalon Recorder (Selenium tests generator) results list. Follow all installation prompts.
4. Successful installation is indicated by the Katalon icon appearing on the Chrome toolbar.

Reference(s): https://addons.mozilla.org/en-US/firefox/addon/katalon-automation-record/

https://chrome.google.com/webstore/detail/katalon-recorder-selenium/ljdobmomdgdljniojadhoplhkpialdid?hl=en-US

## Visual Studio

Visual Studio Community 2019 is Microsoft’s current version 16.2.3. This version should be downloaded directly from Microsoft’s website, https://visualstudio.microsoft.com/downloads/

During installation, select and install the following Workloads:

- Windows section ⇔ Universal Windows Platform Development
- Other Toolsets section ⇔ Office/SharePoint Development
- Other Toolsets section ⇔ .NET Core cross-platform development

### Framework

This section describes the various files and folders that comprise the framework created for Selenium test automation as applied to IAS applications. Note that code examples will not be provided in this

document. The reader can examine the code by opening the Visual Studio solution 'AppTests.sln' located at <TBD>.

Project naming convention -- <*acronym*>_<*sysID*> Example: FIMS_SI217

Complete the following steps to create the initial Project/Solution:

1. Start Visual Studio 2019
2. Select Create a new project
3. Select Unit Test Project (.NET Framework)
4. Click Next
5. Enter a project name according to the aforementioned naming convention
6. Enter the location, <TBD>
7. Enter the Solution name, 'AppTests'
8. Accept the pre-filled Framework selection
9. Click Create

Complete the following steps to configure Visual Studio for Selenium:

1. In the Solution Explorer, right-click on the project and select Manage NuGet packages…
2. Click on the Browse tab
3. Search for and install the following packages (current versions at time of writing):
   NUnit v3.12.0
   NUnit3TestAdapter v3.15.1
   ExcelDataReader v3.6.0
   Selenium.WebDriver v3.141.0
   Selenium.Support v3.141.0
   Selenium.WebDriver.GeckoDriver v0.24.0
   Selenium.WebDriver.ChromeDriver v77.0.3865.4000

Complete the following steps to create a test:

1. In Visual Studio, right-click on the project name
2. Select Add -> Class…
3. Select Class and enter a test name in the Name text field
4. Click Add
5. In Firefox browser, open the application under test
6. Click on the Katalon Recorder icon in the toolbar
7. Record some actions on the application
8. Click on the Export tab
9. In the Format dropdown select 'C# (WebDriver + NUnit)'
10. Click Copy to Clipboard

11. In the class created in steps 1-4, paste from the Clipboard overwriting existing code
12. In the [Test] section of the class, change the method name
13. Save the file and build the solution
14. Return to Katalon Recorder to incrementally build the test script

Complete the following steps to create a shared UI Map (object repository):

1. In Visual Studio, select the project folder
2. Right-click and select Add -> New Item…
3. Click on Application Configuration File
4. Accept the default Name 'App.config'
5. Click Add
6. Select the project folder right-click and select Add -> New Item…
7. Click on Application Configuration File
8. Enter the Name 'SharedUIMap'
9. Click Add
10. In the SharedUIMap.config file, remove the xml reference line
11. Select the project folder, right-click and select Add -> New Folder
12. Name the folder 'Configuration'
13. Move SharedUIMap.config file to the Configuration folder
14. Open the App.config file and add a reference to the SharedUIMap.config
15. Within the project, right-click on References
16. Select Add Reference…
17. In the Reference Manager, select Assemblies ⇔ Framework
18. Search for and check the checkboxes for the following entries:
    System.Configuration
    System.Configuration.Install
    System.Core
    System.Data
19. Click OK
20. Build the solution
21. Right-click on the SharedUIMap.config file and select Properties
22. In File Properties, change the Copy to Output Directory Build Action from 'Do not copy' to 'Copy Always
23. Repeat step 22 for the App.config file

Complete the following steps to create a project to hold shared functions:

1. Click the solution name then right-click and select Add -> New Project…
2. Select C# Class Library (.NET Standard)
3. Click Next

4. Enter ‘Globals’ in Project name text field
5. Click Create
6. In the class library file, change the namespace to ‘Globals’ and the public class to ‘SharedFuncs’
7. Click the test project name, right-click on References -> Add References…
8. In the Reference Manager, expand Projects and check the Globals checkbox
9. Click OK

Complete the following steps to enable reading test data from an Excel spreadsheet:

1. Select the project folder, right-click and select Add -> New Folder
2. Name the folder ‘TestData’
3. Create an Excel spreadsheet file with a descriptive name
4. Rename the sheet with a descriptive name
5. Enter data in the sheet according to the following format:
    a. Row 1 contains column labels
    b. Row 2-n contains test data
    c. Row n+1 contains the value ‘EOD’ in the first column
6. Save the file in the TestData folder under the solution’s path in the file system
7. The spreadsheet name should appear under the TestData folder in Visual Studio
8. Each project in the solution will contain a TestData folder
9. Code for reading from the Excel file is included in the Globals project class file

Complete the following steps to enable test reporting:

1. Select the project folder, right-click and select Add -> New Folder
2. Name the folder ‘Results’
3. Using a text editor, create an empty Log.txt file and save the file in the Results folder under the solution’s path in the file system
4. Each project in the solution will contain a Results folder
5. Code for handling the Log.txt file is included in the Globals project class file

## Notes

The keys in the SharedUIMap.config file are named according to Hungarian notation. The following list contains suggested prefixes:

btn - Button
cb - CheckBox
cbl - CheckBoxList
dd - DropDownList
hl - Hyperlink
img - Image
ib - ImageButton

lbl - Label
lbtn - LinkButton
lb - ListBox
lit - Literal
pnl - Panel
ph - PlaceHolder
rb - RadioButton
rbl - RadioButtonList
tb - Textbox

Additional tests can be created within a project by selecting an existing test, right-click select Copy then click on the project right-click select Paste. The class name and test name in the copied test should be changed appropriately.

Additional projects can be added to a solution using the following procedure:

1. Right click on the solution and select 'Open Folder in File Explorer'
2. In File Explorer, Copy the project folder
3. Rename the new Project folder
4. Open the new project folder and rename the .csproj file
5. Right click on the solution and select Add -> Existing Project…
6. Select the newly copied project folder
7. Open the Properties for the new project
8. Under Application, change the assembly name and default namespace
9. Under Properties, open AssemblyInfo.cs, change the AssemblyTitle and AssemblyProduct
10. Select Tools -> Create GUID and select Option 6.
11. Click Copy then click Exit
12. Paste in the GUID and clear the "" suffix
13. Rename the namespace in the source files to reflect the new namespace from above

## Test Databases

The Selenium test framework assumes the Test database is in a known state. This design decision was arrived at in order to significantly reduce test script complexity. At some point in the script development process, the automated tester requests a restore of the Test database from Production. The automated tester performs the necessary configuration changes in the application to support the automation effort. After these changes are made, a request to the DBAs is submitted to create a Baseline backup of the Test database. Going forward, this Baseline backup is used to restore the Test database to a known state.