Tricentis

Automation Engineer Level 1

Exercise 4

Scan SUT using UIA Engine

Objective

By the end of this exercise, you will be able to use the UIA Engine to scan and steer the SUT based on the Microsoft UI Automation framework in Tosca.

Why is this Important?

Tricentis recommends using the Tosca UIA Engine 3.0 if your test application has controls that aren't supported by the corresponding TBox standard engine, but can be steered with the UI Automation Framework.

Project Perspective

Many organizations use multiple SUTs that communicate with each other. The entire infrastructure landscape might involve Microsoft UI Automation applications, which are difficult to scan and steer. UIA Engine can prove useful in this situation.

Instructions

1. Launch the calculator app on your system
2. Log in to Tosca Commander workspace and navigate to the path **AE1 Exercises>>Modules>>UIA Engine** and launch XScan
3. Right click on the calculator app window in XScan and select **UIA** and click **Scan**
4. Scan and save the following controls
    a. Buttons – **One**, **Plus** and **Equals**
    b. Generic GUI - **Display is 0**
5. Navigate to the **Display is 0** module attribute in the Module created and change the value of its Technical Id **Name** from **Display is 0** to **Display is ***
6. Navigate to the path **AE1 Exercises>>TestCases>>UIA Engine**, create a new TestCase and name it **Perform addition using calculator app**

7. Add the **Calculator** Module to this TestCase and rename it as **Click One**

8. Input **Values** as shared in the table below:

TestStep Value	Value	ActionMode
One	X	Input

9. Add the **Calculator** Module again to this TestCase and rename it as **Click Plus**

10. Input **Values** as shared in the table below:

TestStep Value	Value	ActionMode
Plus	X	Input

11. Add the **Calculator** Module again to this TestCase and rename it as **Click One**

12. Input **Values** as shared in the table below:

TestStep Value	Value	ActionMode
One	X	Input

13. Add the **Calculator** Module again to this TestCase and rename it as **Click Equals**

14. Input **Values** as shared in the table below:

TestStep Value	Value	ActionMode
Equals	X	Input

15. Add the **Calculator** Module again to this TestCase and rename it as **Verify result**

16. Input **Values** as shared in the table below:

TestStep Value	Value	ActionMode
Display is 0	.Name==Display is 2	Verify

17. Mark the TestCase **Completed** and run it in ScratchBook.

Expected outcome

The TestCase should be executed successfully and completion of the specified addition operation should occur.

Hint

Similarly, WinX engine should be used to steer applications that utilize the Windows UI API.