**FlaUI Summary**

**Basics of Windows Automation**

What is UI Automation

* Microsoft’s accessibility framework that works on Microsoft Windows.

The UI Automation is achieved using the following types:

* MSAA – Microsoft Active Accessibility
* UIA2 – For Windows 32 and Winforms
* UIA3 – For Windows store applications and WPF, also supports Windows32 and Winforms to an extent with some limitations (Doesn't support Touch/ Slider functionalities).

**Tools for Windows Automation**

* HP **UFT** (Unified Functional Testing) formerly known as Quick Test Professional (QTF) - functional and regression test automation. Install from microfocus.com. VB Scripting - **LICENSED**
* TestComplete – Functional testing platform. Smart Bear Software founded this. Supports many languages like Java, C#, VB etc.- **LICENSED**
* Ranorex – GUI test automation framework provided by ranorex GMBH company.

C# and VB for scripting - **LICENSED**

* Coded UI – part of Visual Studio developed by Microsoft. .NET languages are supported for scripting – **LICENSED**
* White/ TestStack.White - It is not a tool, instead it is .NET based library. Specific to windows automation. Supports Win32, WinForms, WPF, Silverlight, etc. - **OPENSOURCE**
* FlaUI – It is also a .NET library, which can be installed using NuGet packages.

C# for scripting. Supports windows desktop application. - **OPEN SOURCE**

**What is FlaUI ?**

* FlaUI is a wrapper on the Microsoft’s UI Automation libraries.
* If some feature is not available, the native functionality is available for developing that functionality.
* Flaui provides 3 libraries:
  + FlaUI.Core - for Core functionalities
  + FlaUI.UIA2 - For Microsoft’s UIA2
  + FlaUI.UIA3 - For Microsoft’s UIA3

**Installation of FlaUI**

* Install in 2 ways:
  + Compile the source code of FlaUI obtained from the GitHub repository.
  + Get from NuGet Package

**GUI Inspecting Tools for Windows Applications**

* To know the information of the elements, present in application UI.
* Like inspecting tool/ developer tool in Web browser.
* Since Microsoft does not provide any default inspect tools, such tools are:
  + UISPY – Microsoft - Obsolete
  + INSPECT – Microsoft – Windows SDK
  + VISUALUIAVERIFY – Microsoft – Windows SDK

(Program Files -> WindowsKits -> Version of WindowsSDK version.

Open the VISUALUIAVERIFY application in bin folder.)

* + FLAUINSPECT – Open source

**Installation of FlaUInspect**

* Pre-requisite – Chocolatey package manager.
* Install Chocolatey in cmd running as administrator.
* Install Flauinspect using the command – choco install flauinspect.
* Open the Flauinspect using:
  + Using cmd prompt – just type flauinspect
  + Go to C drive, Chocolatey, bin folder of it. - Application is found in it, create a shortcut of it for easy usage.

**Working with Accessibility Tools**

* 3 modes of inspecting elements using FlaUI:
  + Hover Mode (Ctrl key)
  + Focus Tracking
  + Show XPath
* In general, left panel shows the selected element being inspected.
* Right panel shows the selected element’s information.

*Hover Mode:*

- Hovering over the UI Elements with the Ctrl key being pressed shows the inspect details.

*Focus Tracking:*

- When the UI element gets focused, which is done by clicking.

*Show XPath:*

- Shows Absolute X path of the selected element is displayed. The inspecting details are not visible though (use Hover/ Focus).

- It does not show Relative X path – if required write own by following XPath standards.

**Launching Windows Application using FlaUI**

* Launch any application using the Application class.
* Launching an application is a common thing, so the Application class is part of the FlaUI.Core package.

Application application = new Application(“notepad.exe”);

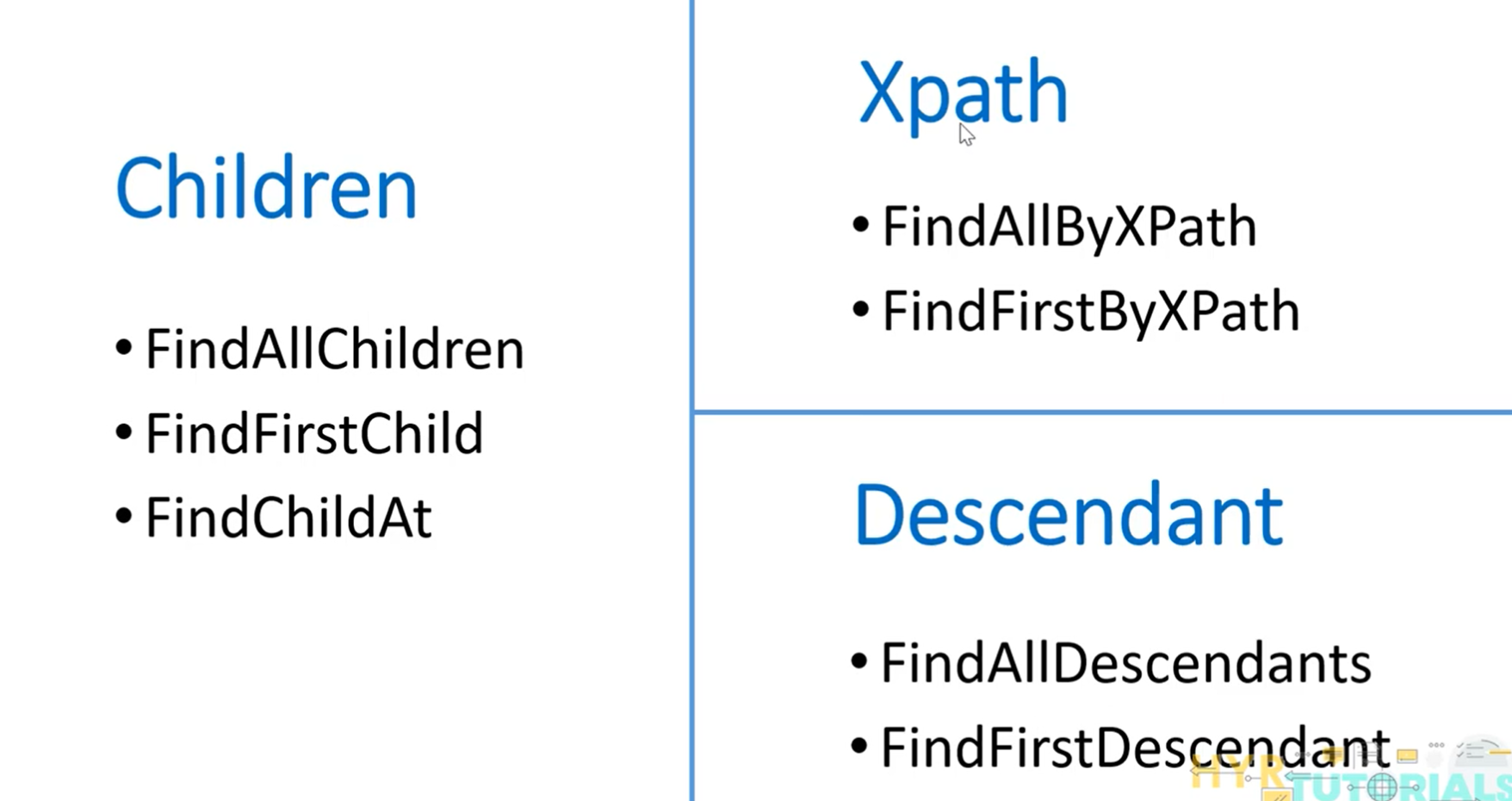
**Handling Different types of Controls with FlaUI**

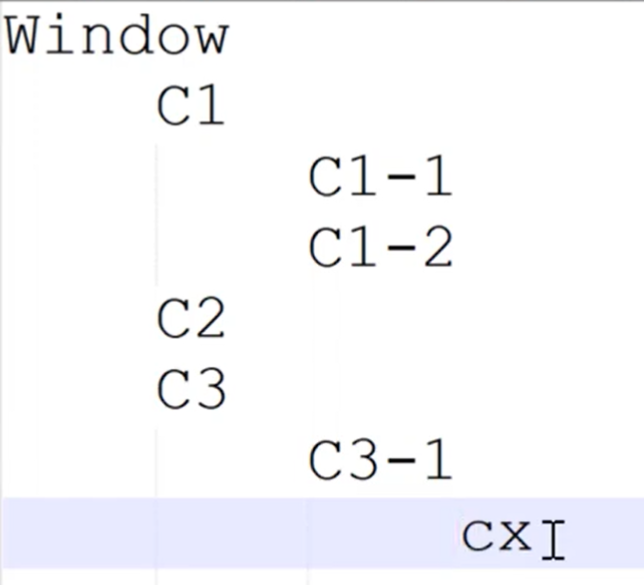
* Launch the application
* Extract mainwindow by specifying the UIAutomation type.
* After getting the mainwindow, Use Find methods to get extract UI element present in the window.
* To get the specific UI element needed, use the condition factory to specify the UI element needed.
* Conditional Factory can be created using two ways:
  + Traditional model conditional factory

ConditionFactory cf = new ConditionFactory(new UIA3PropertyLibrary ())

* + Using Lambda expressions
* Using the FindFirstDescendant() with specifying the conditionfactory, extract the specified UI element and returns it in the form of AutomationElement
* This generic automation element can be converted to specific UI automation element like button or textbox to access the Click () or Enter () operation respectively.
* Use AsButton() or AsTextBox() methods for that.
* Incase of sub-windows, the elements present in it can be accessed from the mainwindow. If required can get the subwindow instance and get the required UI elements.

**Element Finding Mechanisms in FlaUI**

* Finding the right control is crucial.
* 
* If search does not follow any of the above and need manual search- We can have a customised search.
* Children- refers to the direct relationship between the parent and children.
* May share direct or indirect relationship for the parent.



Note:

C3-1 is child of C3

C2 and C3 are siblings, so C3-1 is not the descendant of C2.

* When to use what?
  + Direct relationship – child
  + Indirect relationship – descendant
* Use child if possible, avoiding the depth search in each child, which in return will have higher performance.
* FindAllChildren – returns all children
* FindFirstChildren – returns 1st children
* FindChildAt – returns nth child – index starts from 0
* FindAllDescendants – returns all descendants
* FindFirstDescendant – returns 1st descendant
* Find element in TreeStructure is XPath finding.
* FindAllByXPath -
  + Absolute XPath find the children (start with /);
  + Relative XPath find descendants (start with //).
* Customising the search: - pass the tree scope and condition base
  + FindAll
  + FindAllNested
  + FindAt
  + FindFirst
  + FindFirstNested

FindAll:

* Tree Scope – Children/SubTree/...
* PropertyCondition instance is passed for the ConditionBase
  + Requires the passing the Property and its value

Example: Controltype – Button

Name - “Eight”

**Condition Factory with multiple conditions**

* Doing the Logically operations like OR/AND on the XPath is one way of doing.
* Doing the FalUI way is:

.And()

.Or()

.Not() in the FindMethod();

**Handle Menu and Context Menu with FlaUI**

* Menu is displayed on top of the window, horizontally.
* Context Menu is displayed when right clicked inside the window anywhere. It is displayed vertically.
* Every Application has one menu.
* Menu items can be fetched by the name or index.
* Context menu is can also be fetched directly like mainWindow.ContextMenu

**Mouse actions**

* Use Mouse class to perform Left Click, Right Click
* For specific position, define points. Use the Point class from FlaUI.Core.Shapes class.
* Use Where is my Cursor application to know the cursor position and set the point according ly and click in the mouse for actions.

Mouse.RightClick(point);

* Mouse.MiddleClick() - No METHOD, Use the generic Click and specify the button in it like this:
* Mouse.Click(MouseButton.Middle, point);
* Instead of clicking the mouse, if just need to move the mouse, Use – MoveBy() and MoveTo()
  + MoveBy(x, y)- increments the current x and y coordinates by the given x,y.
  + MoveTo(x, y) - pass the point to the given point directly.
* Instead of moving, If I want to set the mouse position such that its movement is not visible, Mouse.Position(x, y) - target location points.
* Scrolling: (Consider Mathematical co-ordinate system)
  + Mouse.Scroll(10); - Vertical Scroll
    - +ve value - Scroll up
    - -ve Scroll down
  + Mouse.HorizontalScroll(10);
    - +ve value – Scroll right
    - -ve value – Scroll left

**Keyboard Actions**

* Keyboard static class can be used.
* Actions in keyboard:
  + Press
  + Release
  + Type
* To get focusing, do focus it.
* Keyboard.Type() has 3 overloads:
  + Single character. Example: ‘A’
  + String input. Example “Hello”
  + VirtualKeyShort.KEY\_{key}. Example VirtualKeyShort.KEY\_A, Can be provided as an array inside the Type().
* VirtualKeyShort is an enum.
* Press is of two usages:
  + Press and Release:
    - Press()- Pass the VirtualKeyShort.BACK
  + Pressing and hold:
    - Pressing()-
      * Keyboard.Pressing(VirtualKeyShort.CONTROL);
      * Keyboard.Press(“A”);
      * Keyboard.Release(VirtualKeyShort.CONTROL);
    - TypeSimultaneously()
      * Keyboard.TypeSimultaneously(VirtualKeyShort.CONTROL, VirtualKeyShort.KEY\_A);
    - Perform some logic with typing
      * Using statement encloses the pressing statement. The key will be released automatically after executing the block.

Example: Shift+letter – inverts the casing.

**Capture the screenshots**

* Use Capture class – has methods to capture the class.
* Capturing can be done in 3 three ways:
  + Full Screen – Capture.Screen() will return the CaptureImage. Save that as a file using the ToFile() method.
  + Capturing on automation element – No need to convert the Automation element to specific requirements. - Capture.Element(automation element); - returns the CaptureImage which is saved as a file.
  + User defined area – Capture.Rectangle(new Rectangle(100, 200, 200, 500)); - x, y, width, height.

**Accessing Standard UIA Properties using FlaUI**

* To identify a specific UI element, the identification parameters are passed to the condition factory. Those parameters are called properties of that element.
* In case of accessing the properties, there is no need for the converting the generic UI element to the specific UI element.
* Such conversions are only required if trying to perform some actions on it.
* Some properties may not be available directly on the UI element, in that case use the Properties.ControlType