UI Spy

using System.Windows.Forms;

using System.Windows.Automation;

using System.Threading;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

System.Diagnostics.Process p;

AutomationElement AppRoot;

AutomationElement App;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

AppRoot = AutomationElement.RootElement;

//Launch Calculator application

p = System.Diagnostics.Process.Start(@"C:\Windows\System32\calc.exe");

int numWaits = 0;

do

{

App = AppRoot.FindFirst(TreeScope.Children,

new PropertyCondition(AutomationElement.NameProperty, "Calculator"));

numWaits++;

Thread.Sleep(100);

}

while (App == null && numWaits < 5);

if (App == null)

{

return;

}

// In spy++ Controlids are represented hex (00000087) - convert to decimal (135)

ClickBt("00000087"); // 5 135

ClickBt("0000005D"); // + 93

ClickBt("00000087"); // 5 135

ClickBt("00000079"); // = 121

Thread.Sleep(2000);

// Exit Calculator

p.CloseMainWindow();

}

public void ClickBt(string valor)

{

string elemento = System.Convert.ToInt32(valor, 16).ToString();

AutomationElement Ref1;

Ref1 = App.FindFirst(TreeScope.Descendants, new

PropertyCondition(AutomationElement.AutomationIdProperty, elemento));

InvokePattern Click1 = (InvokePattern)Ref1.GetCurrentPattern(InvokePattern.Pattern);

Click1.Invoke();

}

}

}

// get to ROW X (here it's row #1 name is always "Row X")

AutomationElement row1 = dataGrid.FindFirst(TreeScope.Children, new PropertyCondition(AutomationElement.NameProperty, "Row 1"));

// get row header

AutomationElement row1Header = row1.FindFirst(TreeScope.Children, new PropertyCondition(AutomationElement.ControlTypeProperty, ControlType.Header));

// invoke it (select the whole line)

((InvokePattern)row1Header.GetCurrentPattern(InvokePattern.Pattern)).Invoke();

Num lines

var loginLines = datagrid.FindAll(TreeScope.Children, new PropertyCondition(AutomationElement.ControlTypeProperty, ControlType.Custom));

AutomationElement AppRoot;

AutomationElement App;

Process p;

public MakeASum()

{

'Set reference to the root ae element - the desktop

AppRoot = AutomationElement.RootElement

'Launch Calculator application

p = Process.Start("C:\Windows\System32\calc.exe")

'\*\*\*\*\*\*\*

int numWaits = 0;

do

{

App = AppRoot.FindFirst(TreeScope.Children,

new PropertyCondition(AutomationElement.NameProperty, "Calculator"));

numwaits ++;

Thread.Sleep(100);

}

while (aeCalculater==null && numWaits< 5)

if (App == null)

{

Return;

}

// In spy++ Controlids are represented hex (00000087) - convert to decimal (135)

ClickBt("00000087"); // 5 135

ClickBt("0000005D"); // + 93

ClickBt("00000087"); // 5 135

ClickBt("00000079"); // = 121

System.Convert.ToInt32(("00000087"), 16).ToString();

Thread.Sleep(2000)

// Exit Calculator

p.CloseMainWindow()

}

public void ClickBt(string valor)

{

string elemento = System.Convert.ToInt32(valor, 16).ToString();

AutomationElement Ref1;

Ref1 = App.FindFirst(TreeScope.Descendants, new

PropertyCondition(AutomationElement.AutomationIdProperty, **elemento**));

Click1 = (InvokePattern)Ref1.GetCurrentPattern(InvokePattern.Pattern);

InvokePattern ipCLick1;

}

AutomationElement AppRoot;

  private void StartFirst()  
        {

AppRoot = AutomationElement.RootElement;

// GET WINDOW FRAME

           Automation.Condition WindowName = new PropertyCondition(AutomationElement.NameProperty, "Demo Window For Csharp Automation");

           AutomationElement WinRef = AppRoot.FindFirst(TreeScope.Children, WindowName);

// REF. BUTTON HELLO AND CLICK

           Automation.Condition btnHello = new PropertyCondition(AutomationElement.AutomationIdProperty, "btnHello");

           AutomationElement btnStart = WinRef.FindFirst(TreeScope.Descendants, btnHello);  
  
           InvokePattern btn1 = btnStart.GetCurrentPattern(InvokePattern.Pattern) as InvokePattern;

           btn1.Invoke();  
        }             
  
  
        private void btnStartAutomation\_Click(object sender, RoutedEventArgs e)  
        {  
            Thread automateThread = new Thread(new ThreadStart(StartFirst));  
            automateThread.Start();              
        }

USING REGEX

AutomationElement element = FindFirstDescendant(

AutomationElement.FromHandle(windows\_hWnd),

(ele)=>Regex.IsMatch( ele.Current.Name, pattern)

);

//The generic method to find a descendant element:

public static AutomationElement FindFirstDescendant(AutomationElement element, Func<AutomationElement, bool> condition) {

var walker = TreeWalker.ControlViewWalker;

element = walker.GetFirstChild(element);

while (element != null) {

if (condition(element))

return element;

var subElement = FindFirstDescendant(element, condition);

if (subElement != null)

return subElement;

element = walker.GetNextSibling(element);

}

return null;

}

target.delay(1);

UIATarget.localTarget().frontMostApp().mainWindow().textFields()["username"].tap();

target.delay(1);

UIATarget.localTarget().frontMostApp().mainWindow().textFields()["username"].setValue("test");

var processStartInfo = new ProcessStartInfo(@"tdesktop\Program.exe");

var proc = Process.Start(processStartInfo);

Thread.Sleep(3000);

AutomationElement mainWin = AutomationElement.RootElement.FindChildByProcessId(proc.Id);

List<AutomationElement> elmList= GetChildren(mainWin);

//MessageBox.Show(elmList.Count.ToString());

if (elmList.Count == 7)

{

List<AutomationElement> menubar= GetChildren(elmList[6]);

AutomationElement elementNode = menubar[1];

double x = elementNode.GetClickablePoint().X;

double y = elementNode.GetClickablePoint().Y;

win32 w = new win32();

w.move\_left\_click((UInt32)x, (UInt32)y);

}

A menu bar doesn't expose the [InvokePattern](https://msdn.microsoft.com/en-us/library/system.windows.automation.invokepattern.aspx) (see [UI Automation Support for the MenuBar Control Type](https://msdn.microsoft.com/en-us/library/ms752322.aspx)).

However, a menu item can be [Invoke](https://msdn.microsoft.com/en-us/library/ms601607.aspx)d (see [UI Automation Support for the MenuItem Control Type](https://msdn.microsoft.com/en-us/library/ms746680.aspx)).

The following code illustrates how to generate a list of menu items:

AutomationElementCollection items = menubar.FindAll(

TreeScope.Children,

new PropertyCondition(AutomationElement.ControlTypeProperty, ControlType.MenuItem));