Text change Event Handler

AutomationElement Ref1 = AutomationElement.RootElement.FindFirst(TreeScope.Descendants,

new PropertyCondition(AutomationElement.AutomationIdProperty, "t1"));

Automation.AddAutomationPropertyChangedEventHandler(

Ref1,

TreeScope.Element,

(o, e) => { MessageBox.Show("Textbox value property event"); },

ValuePattern.ValueProperty

);

OnFocus

Global variable

string control1 = "";

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

{

Automation.AddAutomationFocusChangedEventHandler(new

AutomationFocusChangedEventHandler(OnFocusChange));

}

private void OnFocusChange(object src, AutomationFocusChangedEventArgs e)

{

Try

{

AutomationElement elemento = (AutomationElement)src;

string control = elemento.Current.AutomationId;

// boton that starts automation

if (control == "btnStart") return;

// while the process doesn’t get the first field, avoid

if (control != "t1" && control=="") return;

// avoid garbage and duplicates

if (control1 !=control && control != "WpfTextView" && control!="control" && control!="")

{

control1 = control;

}

// MessageBox.Show("Got focus " + control);

}

catch (Exception es)

{ }

}

Todos los botones

Automation.AddAutomationEventHandler(InvokePattern.InvokedEvent,

AutomationElement.RootElement,

TreeScope.Descendants,

new AutomationEventHandler(trigger1) );

private static void Trigger1(object sender, AutomationEventArgs e)

{

Console.WriteLine("Evento Ocurrio");

}

= = = = =

AutomationElement Ref1 = AutomationElement.RootElement.FindFirst(TreeScope.Descendants,

new PropertyCondition(AutomationElement.AutomationIdProperty, "button2"));

Automation.AddAutomationEventHandler(InvokePattern.InvokedEvent,

Ref1,

TreeScope.Element,

new AutomationEventHandler(trigger1));

private static void Trigger1(object sender, AutomationEventArgs e)

{

Console.WriteLine("Evento Ocurrio");

}

https://msdn.microsoft.com/en-us/library/system.windows.automation.automationeventhandler(v=vs.110).aspx

AutomationEventHandler UIAeventHandler;

AutomationElement Ref1 = AutomationElement.RootElement.FindFirst(TreeScope.Descendants,

new PropertyCondition(AutomationElement.AutomationIdProperty, "button2"));

Automation.AddAutomationEventHandler(InvokePattern.InvokedEvent,

Ref1,

TreeScope.Element,

UIAeventHandler = new AutomationEventHandler(trigger1));

InvokePattern pattern = element.GetCurrentPattern(InvokePattern.Pattern) as InvokePattern;

Automation.AddAutomationEventHandler(InvokePattern.InvokedEvent,

element,

TreeScope.Element,

new AutomationEventHandler(OnUIAutomationEvent));

pattern.Invoke();

TreeScope Enumeration

Members

|  |  |  |
| --- | --- | --- |
|  | **Member name** | **Description** |
|  | Children | Search include the element's immediate children. |
|  | Descendants | Search include the element's descendants, including children. |
|  | Element | Search include the element itself. |
|  | Subtree | Search include the root of the search and all descendants. |

Automation.AddAutomationEventHandler(event,

element,

scope,

new AutomationEventHandler(MyAutomationEventHandler));

My problem is the following:

 - if I use TreeScope.Element: the handler works perfectly

 - if I use TreeScope.Children: nothing happens, the handler method is never called for any children

- if I use TreeScope.Subtree: nothing happens, the handler method is never called for any element of the subtree.

But if I add another handler at this time for an element (that is in the former subtree) with TreeScope.Element, then the handler method will be called twice each time when the event occurs for the specific element.   
  
I don't know, what the problem is.