UI Automation and Selectors

martedì 21 maggio 2024 11:54

UI or User Interface interactions are the actions performed by a user to interact with the system at the user interface level.

While working with applications like Microsoft Word, Excel, websites and CRM you interact with different buttons to perform actions like changing font types, size, saving documents, inserting data and many more.

These interactions are performed on UI elements which are graphical user interface pieces that constitute an application such as:

- •) windows
- •) text fields
- •) check boxes
- •) Drop-down lists

There are two types of UI interaction <
- Input actions
- Output actions

1) Input Actions :

are those actions through which the user sends data to an application by clicking, typing, selecting or doing any other operation to produce an outcome. Every time the user clicks on a UI element, information is sent to the application and there's an output.

These actions include: clicks, text typing, keyboard shortcuts, mouse hover, clipboard actions like paste

2) Output Actions :

are the actions through which the user takes out data from an application.

These actions include: getting texts, finding elements, identifying images, clipboard actions like copy, saving file from an application.

Input Actions and Methods

In Studio there are specific activities to perform these actions on User Interface. The activities are: click, type into, send hotkeys

1) Click :

it performs the action of clicking on a specified UI element.

These activity has certain properties to customize the action performed.

Properties:

- •) ClickType = specifies the type of mouse click action to be used when simulating the click event. The mouse click action can be single click, double click, click up or click down.
- •) Mouse Button = specifies the mouse button like the left, right or middle button used for the click action.
- •) Timeout = specifies the amount of time in milliseconds to wait for the activity to run before an exception is thrown.

•) Key Modifiers = This property enables the users to add a key modifier. Available options are " Alt, Control, Shift, Window " keys

2) Type Into :

It sends keystrokes to a UI element for example, entering text in an input box or a word file. This activity also certain properties to customize the action performed by it.

Properties :

•) Activate = Selecting this option brings the specified UI element to the foreground and activates it before typing the text. This is useful while the actions are performed on multiple applications.

•) Click Before Typing = Selecting this option the specified UI element is clicked before the text is written.

•) Delay Between Keys = This property specifies the delay time in milliseconds between consecutive key strokes.

•) Empty Field = Selecting this option all existing content in the UI element is erased before your text is typed in.

3) Send Hotkey

It sends keyboard shortcuts to a UI element. This activity also has certain properties to customize the action performed by it.

Properties :

•) Activate = Selecting this option will bring the specified UI element to the foreground and activate it before writing.

•) Click Before Typing = Selecting this checkbox, the specified UI element is clicked before the text is written.

•) Delay Between Keys = This specifies the delay time in milliseconds between two key strokes.

•) Empty Field = Selecting this option all existing content in the UI element is erased before your text is typed in.

All the input actions share some common properties :

•) Delay After, Delay Before = Sets a delay after or before the click or typing any text in an input field.

- Wait For Ready = it waits the target to be ready by verifying certain application tag.

Input Methods

To replicate the input actions, Studio supports 3 input methods for performing input actions:

- Default = captures the input given while interacting with the UI element of the screen.
- Send Window messages = captures actions performed using keyboard and mouse
- Simulate Type / Click = is much faster, it does not capture input given by the keyboard

---

Containers

A container is a box that holds the User Interaction activities performed on the same application.

(-) It specifies the top-level UI window where the elements belong
(-) It gives details of the button or field to be used
(-) It helps define the scope of a variable
(-) It can be modified by the user

When recording actions like web and desktop recorder are used, containers are automatically generated but changes can be made to them from the properties panel.
Alternatively, the user can drag-and-drop the required container activities like open application from the activities panel.

Types of container activities in Studio

1) Open Application : it launches a specified application and performs actions within it. It can also pass a list of arguments to the application.

2) Attach Browser : it enables the user to attach to an open browser and perform actions within it.
This activity is automatically generated when we use the web recorder.

3) Open Browser : it enables the user to open a browser with a specified URL and execute multiple activities within it.

[illegible] it enables the user to attach to an already open

4) Attach Window : It enables the user to [illegible]
window and perform actions within it.
This activity is automatically generated when using
the desktop recorder.

5) Get Active Window : It retrieves the current active window and
enables the user to perform actions within it.

Recording

Recording is a technique that helps capture the manual actions performed
by the user on the screen and translate them into sequences.

- It captures the step-by-step activity of the user.
- It generates the series of all the workflows needed
  for automation.
- It helps in identifying the elements on the screen,
  reading data from display.
- It allows you to save a lot of time when automating the processes

You can use the recording feature by clicking the recording button
in the recording wizard ●
after selecting the type of recording required and do the
necessary automation steps.

All user interface elements are highlighted while recording to ensure
that correct buttons, fields or menus are selected.

After finishing the recording steps, you need to save the recording
↳ UiPath Studio will create the workflow according to the
actions you recorded.

In Studio there are several user actions that can be recorded

- Text typing
- Drop-down lists
- Mouse clicks
- Check boxes

UiPath - Recorda...

Types of Recorders

(1) Basic Recording = used for automating desktop applications
It is used when you only want to work with a single window

[illegible]

(2) Desktop Recording = captures data from desktop applications when multiple actions are performed in multiple windows.

(3) Web Recording = used to automate user actions performed on websites. You can do web recording using any 3 popular web browsers: Edge, Chrome, Firefox

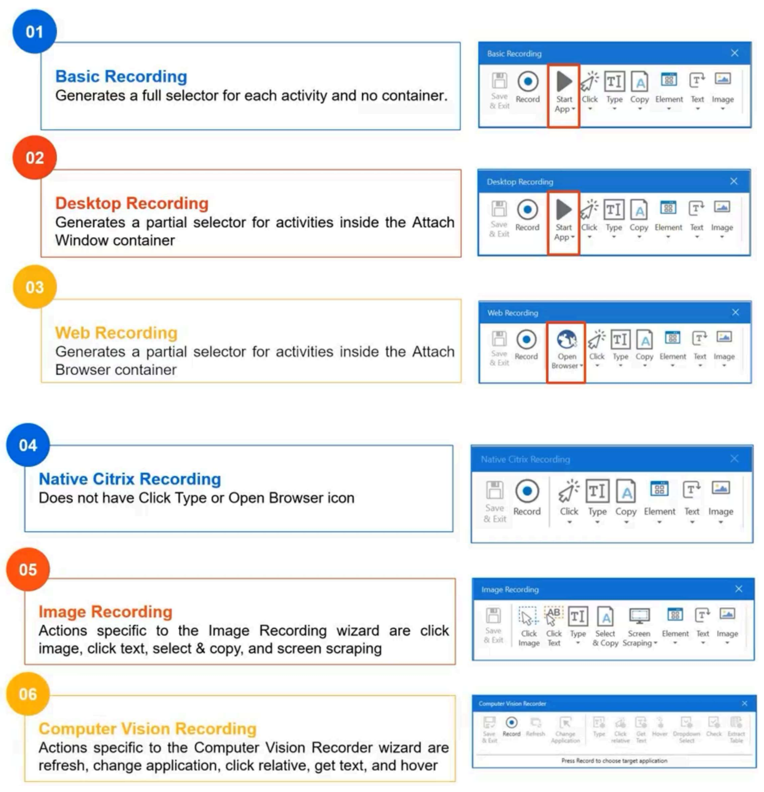
(4) Image Recording = used to capture images for all virtual environments. The automation cannot identify all the applications and it relies on image recognition.
In this type of recording, the precise position of data is required.

(5) Native Citrix Recording = used for Citrix environments that are configured to support UiPath Native Citrix.
Once the setup is made, this recorder will work seamlessly on virtual environments. Just like the desktop recorder works on a regular desktop.

(6) Computer Vision Recording = used in a virtual desktop environment such as Citrix machines. It uses a computer vision neural network to identify UI elements such as buttons, text input fields or checkboxes without the use of selectors.

Recording Wizard

It opens when you select any of the 6 recording tools.



01 **Basic Recording**
Generates a full selector for each activity and no container.

02 **Desktop Recording**
Generates a partial selector for activities inside the Attach Window container

03 **Web Recording**
Generates a partial selector for activities inside the Attach Browser container

04 **Native Citrix Recording**
Does not have Click Type or Open Browser icon

05 **Image Recording**
Actions specific to the Image Recording wizard are click image, click text, select & copy, and screen scraping

06 **Computer Vision Recording**
Actions specific to the Computer Vision Recorder wizard are refresh, change application, click relative, get text, and hover

Selectors

Selectors are a fundamental part of Studio as they are used to recognize the UI elements of the user screen.

Types of Selectors

1) Full Selectors: when automation is created through Basic recorder, full selectors are generated. They start with a window or XML identifier, and they contain all the information of the UI element. They are suitable when the workflow has to switch through multiple windows, and the use of containers would add unnecessary complexity.

2) Partial Selectors: when automation is created through desktop recorder, partial selectors are generated. They do not contain the top-level window information, they are more suitable when the robot has to perform multiple actions in the same window or application. In these cases, the activities will be placed inside a container, such as Attach Browser or Attach window.

Selector Editor Panel

It displays the selector for the specified UI object and enables you to customize it by checking and unchecking nodes.

The selector editor can be accessed in two ways.

↳ (a) The first way is to access it through the Properties Panel. Go to the designer panel and click on the activity whose selector is to be edited. On the property panel of the activity click the ellipse button of the selector property.

(b) The second way is through the designer panel. Click on the ≡ button on the activity and select the "Edit Selector" option from the context menu.

Selector Editor Properties

(a) Validate = this button shows the status of the selector by checking the validity of the selector definition and the visibility of the target element on the screen.

(b) Indicate = this property lets you indicate a new UI element to replace the previous one.

(c) Repair = enables the user to re-indicate the same target UI element and repair the selector.

The button is available only when the selector is invalid.

(•) Highlight = It brings the target element in the foreground
The button stays on until it's switched off and it is enabled only if the selector is valid.

Dynamic Selectors

It finds the address of a UI element dynamically and identifies the attributes of the element across windows.

It uses a variable or an argument as a property for the attribute of the target tag. This allows the selector to easily identify a target element based on the value of the variable or argument.
The variable or argument can be changed to interact with a different element without changing the selector itself.
This is best-suited situations in which the target element constantly changes its value

---

UI Explorer

UI Explorer is an advanced tool for customizing the selector.
It enables the user to create a custom selector for a specific UI element

To open the UI explorer window, click the UI Explorer button in the design tab.

(•) The "validate" button shows whether a selector is valid

(•) "Indicate element" button is used to indicate a new UI element to replace the previous one

(•) "Indicate anchor" button is used to choose an anchor relative to the target UI element.

(•) "Repair button" enables you to re-indicate the same target UI element and repair the selector

(•) "Highlight button" brings the target element in the foreground

(–) "Visual tree" panel displays a tree of the user interface hierarchy and enables you to navigate through various options.
When you double click a UI element from the visual tree, you can access "selector editor", "selector attributes" and "property explorer".

The bottom part of the panel displays the actual XML fragment that you have to use in a project.
The top part of the panel enables you to view all the nodes in a selector and eliminates the ones that are unnecessary by clearing the check-box in front of them.
↳ selecting a node here displays its attributes in the select attributes and property explorer panels.

⟹ "Selector attributes" panel displays all the available attributes of a selected node. You can add or eliminate some node attributes by selecting or clearing the check-box.

⟹ "Property Explorer" panel displays all the attributes that specified UI object can have, including the ones that do not appear in the selector.

UI Framework defines the technology that determines UI elements and their selectors. The user can switch between 3 UI frameworks from the UI Explorer designated button:

1) Default: this is the proprietary method that usually works correctly with all types of user interfaces.

2) Active Accessibility: this represents an earlier solution from Microsoft that makes apps accessible. It is recommended when using legacy software if the default framework does not work as expected

3) UI Automation: This is the improved accessibility model from Microsoft, which is recommended when using legacy software if the default framework does not worked as expected.
The selectors generated may be very different from one to another

Anchors

[illegible] when a reliable selector for a UI Element is not available.

Anchors are used [illegible]

If a selector keeps changing after every iteration, the user defines an anchor that can locate the selector every time it changes.

(•) Anchor Base activity searches for a UI element by using other UI elements as anchors.
The activity locates the anchor and its relative UI element on the screen.

Fine-Tuning Selectors

In most cases the automatically generated selectors are not reliable enough.
This issue is solved by Fine-Tuning.

Fine-tuning is the process of refining selectors to have the workflow correctly executed in situations in which the generated selector is unreliable, too specific or too sensitive with regards to system changes.

It mainly consists of changes such as:

(•) Adding wild cards
(•) Using the repair function
(•) Using variables in selectors

The different tools to fine-tune selectors are:

(•) Anchor Base
useful in cases in which the attribute values are not reliable, but a stable UI Element exists on the screen.
In such case, the stable UI Element is linked to the target UI Element using the "Anchor Base" activity

(•) Relative Selector
A selector can be improved by using the "Indicate Anchor" option in UI Explorer. It incorporates the information about the anchor's selector in the selector of the target UI Element.
However the new selector will probably need additional editing as the part that made the first selector will still be in the new selector.
That part, like a dynamic ID, needs to be removed and the selector will stabilize using the anchor's selector.

(•) Visual Tree Hierarchy

It is useful when the target UI Element's selector is not reliable, but the selector of the UI Element right above in the hierarchy is reliable.

Here, the selector needs further editing and validation as the dynamic part needs to be removed. This can be done using the UI Explorer

c) Find Children

It identifies the children of a certain UI Element.

The output of the activity is the collection of children.

To make it work, the user needs to come up with a mechanism to identify only the target UI element using one of its attributes that makes it unique between the children, but would not be enough to identify it universally.

The Recommended Approach to fine-tune selectors are:

1) Use the Selector Editor to inspect the applications

2) Examine the elements from the Selector Editor path

3) Identify the element causing the issue

4) Change element's attributes through "Indicate Elements" and use UI Explorer to find various selectors using "Visual Tree"

5) Highlight the application after modifying attribute values

6) Test the functionality of the recently debugged element.