pywinauto_recorder

David Pratmarty

Jun 13, 2021

CONTENTS:

"Pywinauto recorder" records user interface actions and saves them in a Python script. The saved Python script can be run to replay the user interface actions previously recorded.

"Pywinauto recorder" is a unique inspect/record/replay tool in the open source field because it generates reliable scripts without hard-coded coordinates thanks to Pywinauto.

Pywinauto is a library that uses accessibility technologies allowing you to automate almost any type of GUI: MFC, VB6, VCL, simple WinForms controls and most of the old legacy apps, WinForms, WPF, Store apps, Qt, and browsers.

The functions of the generated Python script return Pywinauto wrappers so it can be enhanced with Pywinauto methods.

CHAPTER

ONE

PYWINAUTO_RECORDER FOR WINDOWS

- "pywinauto_recorder.exe" is a standalone application, it's the compiled version of "pywinauto_recorder.py" for 64-bit Windows.

- "pywinauto_recorder.py" is the main source, you will find the Pyhon code downloading the github clone.

1.1 Usage

- Double click on "pywinauto_recorder.exe" or run "python.exe pywinauto_recorder.py" to start the recorder.
- The recorder is started in display information mode, a tray icon is added in the right-side of the Windows Taskbar.
- Press CTRL+SHIFT+f to copy the code that finds the element colored green or orange to the clipboard.
- Press CTRL+ALT+r to switch to "Record" mode.
- If the element below the mouse cursor can be uniquely identified, it will turn green or orange.
- You can then click or perform another action on the user interface and it will be recorded in the generated Python script.
- Repeat this process performing a few actions on the user interface and when you're done press CTRL+ALT+r to end recording.
- The generated Python script is saved in the "Pywinauto recorder" folder in your home folder and copied in the clipboard.
- Click on "Quit" in the tray menu.
- To replay a Python script, you can drag and drop it to "pywinauto_recorder.exe"

Warning: Pywinauto_recorder_exe does not work on all PCs. If it doesn't work use the Python version. Help me to find a solution by filling this form.

1.2 Icons

Some transparent icons are displayed at the top left of the screen:

- an icon corresponds to Record/Stop mode. Press CTRL+ALT+r to switch.
- a magnifiying glass icon cooresponds to Display information mode.
- a bulb icon corresponds to Smart mode. Press CTRL+ALT+S to activate it.
- another icon displays a green bar at each iteration of the loop. It allows you to see how fast the loop is running.

1.2.1 More explanations

The main of "Pywinauto recorder" is an infinite loop where at each iteration it:

(1) finds the path of the element under the mouse cursor. The path is formed by the window_text and control_type pair of the element and all its ancestors.

(2) searches for an unambiguous path, if found, it colors the element region green or orange.

(3) records a user action in a file involving the last recognized unique path.

Note: To reflect the position of the mouse cursor as closely as possible, an offset is added to the user actions recorded in the generated Python script. This offset is proportional to the size of the element and relative to the center of the element.

If the path of the element under the mouse cursor is not ambiguous, the region of the element is colored green. Otherwise two st

(1) All elements having the same path are ordered in a 2D array. The path of the element region under the mouse cursor is disambiguated by adding a row index and a column index so that it is colored orange. The other element regions are colored red

(2) When Smart mode is enabled, an element whose path is unambiguous is searched on the same line on the left, if found its region is colored blue and the element under the mouse cursor is colored orange.

CHAPTER

TWO

PYWINAUTO RECORDER API

2.1 Setup

If you want to replay the actions of a script generated by "pywinauto_recorder.exe" in your Pyhton interpreter you must install the pywinauto_recorder package:

```
pip install -U pywinauto_recorder
```

2.2 Functions

Detailed API documentation

pywinauto_recorder.player	This module contains functions to replay a sequence of user actions automatically.

2.2.1 pywinauto_recorder.player

This module contains functions to replay a sequence of user actions automatically.

Functions

click([element_path, duration, mode, ...])	Clicks on element
double_left_click(element_path[, duration, ...])	Double left clicks on element
drag_and_drop(element_path1, element_path2)	Drags and drop with left button pressed from element_path1 to element_path2.
exists(element_path[, timeout])	Tests if en UI_Element exists.
find([element_path, timeout])	Finds an element
full_definition(str_shortcut)	Returns the full element path associated to the shortcut defined in the previously loaded dictionary
left_click(element_path[, duration, mode, ...])	Left clicks on element
load_dictionary(filename_key, filename_def)	Loads a dictionary
menu_click(element_path, menu_path[, ...])	Clicks on menu item.
middle_drag_and_drop(element_path1, ...[, ...])	Drags and drop with middle button pressed from element_path1 to element_path2.

continues on next page

Table 2 – continued from previous page

mouse_wheel(steps[, pause])	Turns the mouse wheel up or down.
move(element_path[, duration, mode, timeout])	Moves on element
right_click(element_path[, duration, mode, ...])	Right clicks on element
right_drag_and_drop(element_path1, element_path2)	Drags and drop with right button pressed from element_path1 to element_path2.
select_file(element_path, full_path[, ...])	Selects a file in an already opened file dialog.
send_keys(str_keys[, pause, with_spaces, ...])	Parses the keys and type them You can use any Unicode characters (on Windows) and some special keys.
set_combobox(element_path, value[, ...])	Sets the value of a combobox.
set_text(element_path, value[, duration, ...])	Sets the value of a text field.
shortcut(str_shortcut)	Returns the shortcut path associated to the shortcut defined in the previously loaded dictionary
triple_left_click(element_path[, duration, ...])	Triple left clicks on element
unescape_special_char(string)	
wait_is_ready_try1(wrapper[, timeout])	Waits until element is ready (wait while greyed, not enabled, not visible, not ready, ...) : So far, I didn't find better than wait_cpu_usage_lower when greyed but must be enhanced

pywinauto_recorder.player.click

pywinauto_recorder.player.**click**(*element_path=None*, *duration=None*, *mode=<MoveMode.linear: 0>*, *button='left'*, *timeout=None*, *wait_ready=True*)

Clicks on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>)), None]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move, it just sends WM_CLICK window message, useful for minimized or non-active window).
- **mode** (Enum) – move mouse mode
- **button** (str) – mouse button: 'left','double_left', 'triple_left', 'right'
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element
- **wait_ready** (bool) – if True waits until the element is ready

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.double_left_click

pywinauto_recorder.player.**double_left_click**(*element_path*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*, *wait_ready=True*)

Double left clicks on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element
- **wait_ready** (bool) – if True waits until the element is ready

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.drag_and_drop

pywinauto_recorder.player.**drag_and_drop**(*element_path1*, *element_path2*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*)

Drags and drop with left button pressed from element_path1 to element_path2.

Parameters

- **element_path1** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **element_path2** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper with element_path2

pywinauto_recorder.player.exists

pywinauto_recorder.player.**exists**(*element_path*, *timeout=None*)

Tests if en UI_Element exists.

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of the found element or None

pywinauto_recorder.player.find

pywinauto_recorder.player.**find**(*element_path=None*, *timeout=None*)

Finds an element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>)), None]) – element path
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.full_definition

pywinauto_recorder.player.**full_definition**(*str_shortcut*)

Returns the full element path associated to the shortcut defined in the previously loaded dictionary

Parameters **str_shortcut** (str) – shortcut

Return type str

pywinauto_recorder.player.left_click

pywinauto_recorder.player.**left_click**(*element_path*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*, *wait_ready=True*)

Left clicks on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)

- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element
- **wait_ready** (bool) – if True waits until the element is ready

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.load_dictionary

pywinauto_recorder.player.**load_dictionary**(*filename_key*, *filename_def*, *encoding='utf8'*)

Loads a dictionary

Parameters

- **filename_key** (str) – filename of the key file
- **filename_def** (str) – filename of the definition file
- **encoding** (str) – encoding of the dictionary file

Return type None

pywinauto_recorder.player.menu_click

pywinauto_recorder.player.**menu_click**(*element_path*, *menu_path*, *duration=None*, *mode=<MoveMode.linear: 0>*, *menu_type='QT'*, *timeout=None*)

Clicks on menu item.

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **menu_path** (str) – menu path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **menu_type** (str) – menu type ('QT', 'NPP')
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of the clicked item

pywinauto_recorder.player.middle_drag_and_drop

pywinauto_recorder.player.**middle_drag_and_drop**(*element_path1*, *element_path2*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*)

Drags and drop with middle button pressed from element_path1 to element_path2.

Parameters

- **element_path1** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **element_path2** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper with element_path2

pywinauto_recorder.player.mouse_wheel

pywinauto_recorder.player.**mouse_wheel**(*steps*, *pause=0*)

Turns the mouse wheel up or down.

Parameters

- **steps** (int) – number of wheel steps, if positive the mouse wheel is turned up else it is turned down
- **pause** (float) – pause in seconds between each wheel step

Return type None

pywinauto_recorder.player.move

pywinauto_recorder.player.**move**(*element_path*, *duration=0.5*, *mode=<MoveMode.linear: 0>*, *timeout=120*)

Moves on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (float) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.right_click

pywinauto_recorder.player.**right_click**(*element_path*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*, *wait_ready=True*)

Right clicks on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element
- **wait_ready** (bool) – if True waits until the element is ready

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.right_drag_and_drop

pywinauto_recorder.player.**right_drag_and_drop**(*element_path1*, *element_path2*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*)

Drags and drop with right button pressed from element_path1 to element_path2.

Parameters

- **element_path1** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **element_path2** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper with element_path2

pywinauto_recorder.player.select_file

pywinauto_recorder.player.**select_file**(*element_path*, *full_path*, *force_slow_path_typing=False*)

Selects a file in an already opened file dialog.

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **full_path** (str) – the full path of the file to select
- **force_slow_path_typing** (bool) – if True it will type the path even if the current path of the dialog box is the same

than the file to select

Return type None

pywinauto_recorder.player.send_keys

pywinauto_recorder.player.**send_keys**(*str_keys*, *pause=0.1*, *with_spaces=True*, *with_tabs=True*, *with_newlines=True*, *turn_off_numlock=True*, *vk_packet=True*)

Parses the keys and type them You can use any Unicode characters (on Windows) and some special keys. See https://pywinauto.readthedocs.io/en/latest/code/pywinauto.keyboard.html

Parameters

- **str_keys** (str) – string representing the keys to be typed
- **pause** (float) – pause in seconds between each typed key
- **with_spaces** (bool) – if False spaces are not taken into account
- **with_tabs** (bool) – if False tabs are not taken into account
- **with_newlines** (bool) – if False newlines are not taken into account
- **turn_off_numlock** (bool) – if True numlock is turned off
- **vk_packet** (bool) – For Windows only, pywinauto defaults to sending a virtual key packet (VK_PACKET) for textual input

Return type None

pywinauto_recorder.player.set_combobox

pywinauto_recorder.player.**set_combobox**(*element_path*, *value*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*)

Sets the value of a combobox.

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **value** (str) – value of the combobox
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)

- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

Return type None

pywinauto_recorder.player.set_text

pywinauto_recorder.player.**set_text**(*element_path*, *value*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*, *pause=0.1*)

Sets the value of a text field.

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **value** (str) – value of the combobox
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element
- **pause** (float) – pause in seconds between each typed key

Return type None

pywinauto_recorder.player.shortcut

pywinauto_recorder.player.**shortcut**(*str_shortcut*)

Returns the shortcut path associated to the shortcut defined in the previously loaded dictionary

Parameters **str_shortcut** (str) – shortcut

Return type str

pywinauto_recorder.player.triple_left_click

pywinauto_recorder.player.**triple_left_click**(*element_path*, *duration=None*, *mode=<MoveMode.linear: 0>*, *timeout=None*, *wait_ready=True*)

Triple left clicks on element

Parameters

- **element_path** (Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]) – element path
- **duration** (Optional[float]) – duration in seconds of the mouse move (if duration is -1 the mouse cursor doesn't move)
- **mode** (Enum) – move mouse mode
- **timeout** (Optional[float]) – period of time in seconds that will be allowed to find the element

- **wait_ready** (bool) – if True waits until the element is ready

Return type Union[str, UIAWrapper, NewType()(UI_Coordinates, (<class 'float'>, <class 'float'>))]

Returns Pywinauto wrapper of clicked element

pywinauto_recorder.player.unescape_special_char

pywinauto_recorder.player.**unescape_special_char**(*string*)

pywinauto_recorder.player.wait_is_ready_try1

pywinauto_recorder.player.**wait_is_ready_try1**(*wrapper*, *timeout=120*)

Waits until element is ready (wait while greyed, not enabled, not visible, not ready, ...) : So far, I didn't find better than wait_cpu_usage_lower when greyed but must be enhanced

Classes

MoveMode(value)	An enumeration.
PlayerSettings()	
Region([relative_path, regex_title])	
Window	alias of *pywinauto_recorder.player.Region*

pywinauto_recorder.player.MoveMode

class pywinauto_recorder.player.**MoveMode**(*value*)

An enumeration.

pywinauto_recorder.player.PlayerSettings

class pywinauto_recorder.player.**PlayerSettings**

pywinauto_recorder.player.Region

class pywinauto_recorder.player.**Region**(*relative_path=None*, *regex_title=False*)

pywinauto_recorder.player.Window

pywinauto_recorder.player.**Window**
alias of *pywinauto_recorder.player.Region*

CHAPTER

THREE

CONTRIBUTING TO PYWINAUTO RECORDER

You are here to help on Pywinauto recorder? Awesome, feel welcome and read the following sections in order to know how to ask questions and how to work on something.

All members of our community are expected to follow our *Code of Conduct*. Please make sure you are welcoming and friendly in all of our spaces.

3.1 Get in touch

- Ask usage questions ("How do I?") on StackOverflow.
- Report bugs, suggest features or view the source code on GitHub.

3.2 Contributing to development

If you want to deep dive and help out with development on Pywinauto recorder, then first get the project installed locally. After that is done we suggest you have a look at tickets in our issue tracker that are labelled Good First Issue. These are meant to be a great way to get a smooth start and won't put you in front of the most complex parts of the system.

If you are up to more challenging tasks with a bigger scope, then there are a set of tickets with a Feature or Improvement tag. These tickets have a general overview and description of the work required to finish. If you want to start somewhere, this would be a good place to start (make sure that the issue also have the Accepted label). That said, these aren't necessarily the easiest tickets. They are simply things that are explained. If you still didn't find something to work on, search for the Sprintable label. Those tickets are meant to be standalone and can be worked on ad-hoc.

You can read all of our /development/index to understand more the development of Pywinauto recorder. When contributing code, then please follow the standard Contribution Guidelines set forth at contribution-guide.org.

3.3 Contributing to documentation

Documentation for Pywinauto recorder itself is hosted at https://pywinauto-recorder.readthedocs.io.

At https://docs.readthedocs.io there are guidelines around writing and formatting documentation for the project.

CHAPTER

FOUR

CODE OF CONDUCT

Like the technical community as a whole, the Pywinauto recorder team and community is made up of a mixture of professionals and volunteers from all over the world, working on every aspect of the mission - including mentorship, teaching, and connecting people.

Diversity is one of our huge strengths, but it can also lead to communication issues and unhappiness. To that end, we have a few ground rules that we ask people to adhere to. This code applies equally to founders, mentors and those seeking help and guidance.

This isn't an exhaustive list of things that you can't do. Rather, take it in the spirit in which it's intended - a guide to make it easier to enrich all of us and the technical communities in which we participate.

This code of conduct applies to all spaces managed by the Pywinauto recorder project. This includes IRC, the mailing lists, the issue tracker, and any other forums created by the project team which the community uses for communication. In addition, violations of this code outside these spaces may affect a person's ability to participate within them.

- **Be friendly and patient.**
- **Be welcoming.** We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to members of any race, ethnicity, culture, national origin, colour, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.
- **Be considerate.** Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.
- **Be respectful.** Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It's important to remember that a community where people feel uncomfortable or threatened is not a productive one. Members of the Pywinauto recorder community should be respectful when dealing with other members as well as with people outside the Pywinauto recorder community.
- **Be careful in the words that you choose.** We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren't acceptable. This includes, but is not limited to:
  - Violent threats or language directed against another person.
  - Discriminatory jokes and language.
  - Posting sexually explicit or violent material.
  - Posting (or threatening to post) other people's personally identifying information ("doxing").
  - Personal insults, especially those using racist or sexist terms.
  - Unwelcome sexual attention.

  - Advocating for, or encouraging, any of the above behavior.
  - Repeated harassment of others. In general, if someone asks you to stop, then stop.

- **When we disagree, try to understand why.** Disagreements, both social and technical, happen all the time and Pywinauto recorder is no exception. It is important that we resolve disagreements and differing views constructively. Remember that we're different. The strength of Pywinauto recorder comes from its varied community, people from a wide range of backgrounds. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn't mean that they're wrong. Don't forget that it is human to err and blaming each other doesn't get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

Original text courtesy of the Speak Up! project. This version was adopted from the Django Code of Conduct.

4.1 Description

"Pywinauto recorder" records user interface actions and saves them in a Python script. The saved Python script can be run to replay the user interface actions previously recorded.

"Pywinauto recorder" is a unique inspect/record/replay tool in the open source field because it generates reliable scripts without hard-coded coordinates thanks to Pywinauto.

Pywinauto is a library that uses accessibility technologies allowing you to automate almost any type of GUI: MFC, VB6, VCL, simple WinForms controls and most of the old legacy apps, WinForms, WPF, Store apps, Qt, and browsers.

The functions of the generated Python script return Pywinauto wrappers so it can be enhanced with Pywinauto methods.

4.2 Pywinauto_recorder for Windows

- "pywinauto_recorder.exe" is a standalone application, it's the compiled version of "pywinauto_recorder.py" for 64-bit Windows.

- "pywinauto_recorder.py" is the main source, you will find the Pyhon code downloading the github clone.

4.2.1 Usage

- Double click on "pywinauto_recorder.exe" or run "python.exe pywinauto_recorder.py" to start the recorder.
- The recorder is started in display information mode, a tray icon is added in the right-side of the Windows Taskbar.
- Press CTRL+SHIFT+f to copy the code that finds the element colored green or orange to the clipboard.
- Press CTRL+ALT+r to switch to "Record" mode.
- If the element below the mouse cursor can be uniquely identified, it will turn green or orange.
- You can then click or perform another action on the user interface and it will be recorded in the generated Python script.
- Repeat this process performing a few actions on the user interface and when you're done press CTRL+ALT+r to end recording.
- The generated Python script is saved in the "Pywinauto recorder" folder in your home folder and copied in the clipboard.
- Click on "Quit" in the tray menu.
- To replay a Python script, you can drag and drop it to "pywinauto_recorder.exe"

Warning: Pywinauto_recorder_exe does not work on all PCs. If it doesn't work use the Python version. Help me to find a solution by filling this form.

4.2.2 Icons

Some transparent icons are displayed at the top left of the screen:

- an icon corresponds to Record/Stop mode. Press CTRL+ALT+r to switch.
- a magnifiying glass icon cooresponds to Display information mode.
- a bulb icon corresponds to Smart mode. Press CTRL+ALT+S to activate it.
- another icon displays a green bar at each iteration of the loop. It allows you to see how fast the loop is running.

More explanations

The main of "Pywinauto recorder" is an infinite loop where at each iteration it:

(1) finds the path of the element under the mouse cursor. The path is formed by the window_text and control_type pair of the element and all its ancestors.

(2) searches for an unambiguous path, if found, it colors the element region green or orange.

(3) records a user action in a file involving the last recognized unique path.

Note: To reflect the position of the mouse cursor as closely as possible, an offset is added to the user actions recorded in the generated Python script. This offset is proportional to the size of the element and relative to the center of the element.

If the path of the element under the mouse cursor is not ambiguous, the region of the element is colored green. Otherwise two st

(1) All elements having the same path are ordered in a 2D array. The path of the element region under the mouse cursor is disambiguated by adding a row index and a column index so that it is colored orange. The other element regions are colored red

(2) When Smart mode is enabled, an element whose path is unambiguous is searched on the same line on the left, if found its region is colored blue and the element under the mouse cursor is colored orange.

4.3 Pywinauto recorder API

4.3.1 Setup

If you want to replay the actions of a script generated by "pywinauto_recorder.exe" in your Pyhton interpreter you must install the pywinauto_recorder package:

```
pip install -U pywinauto_recorder
```

4.3.2 Functions

Detailed API documentation

pywinauto_recorder.player	This module contains functions to replay a sequence of user actions automatically.

4.4 Contributing to Pywinauto recorder

You are here to help on Pywinauto recorder? Awesome, feel welcome and read the following sections in order to know how to ask questions and how to work on something.

All members of our community are expected to follow our *Code of Conduct*. Please make sure you are welcoming and friendly in all of our spaces.

4.4.1 Get in touch

- Ask usage questions ("How do I?") on StackOverflow.
- Report bugs, suggest features or view the source code on GitHub.

4.4.2 Contributing to development

If you want to deep dive and help out with development on Pywinauto recorder, then first get the project installed locally. After that is done we suggest you have a look at tickets in our issue tracker that are labelled Good First Issue. These are meant to be a great way to get a smooth start and won't put you in front of the most complex parts of the system.

If you are up to more challenging tasks with a bigger scope, then there are a set of tickets with a Feature or Improvement tag. These tickets have a general overview and description of the work required to finish. If you want to start somewhere, this would be a good place to start (make sure that the issue also have the Accepted label). That said, these aren't necessarily the easiest tickets. They are simply things that are explained. If you still didn't find something to work on, search for the Sprintable label. Those tickets are meant to be standalone and can be worked on ad-hoc.

You can read all of our /development/index to understand more the development of Pywinauto recorder. When contributing code, then please follow the standard Contribution Guidelines set forth at contribution-guide.org.

4.4.3 Contributing to documentation

Documentation for Pywinauto recorder itself is hosted at https://pywinauto-recorder.readthedocs.io.

At https://docs.readthedocs.io there are guidelines around writing and formatting documentation for the project.

4.5 Code of Conduct

Like the technical community as a whole, the Pywinauto recorder team and community is made up of a mixture of professionals and volunteers from all over the world, working on every aspect of the mission - including mentorship, teaching, and connecting people.

Diversity is one of our huge strengths, but it can also lead to communication issues and unhappiness. To that end, we have a few ground rules that we ask people to adhere to. This code applies equally to founders, mentors and those seeking help and guidance.

This isn't an exhaustive list of things that you can't do. Rather, take it in the spirit in which it's intended - a guide to make it easier to enrich all of us and the technical communities in which we participate.

This code of conduct applies to all spaces managed by the Pywinauto recorder project. This includes IRC, the mailing lists, the issue tracker, and any other forums created by the project team which the community uses for communication. In addition, violations of this code outside these spaces may affect a person's ability to participate within them.

- **Be friendly and patient.**
- **Be welcoming.** We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to members of any race, ethnicity, culture, national origin, colour, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.
- **Be considerate.** Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.

- **Be respectful.** Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It's important to remember that a community where people feel uncomfortable or threatened is not a productive one. Members of the Pywinauto recorder community should be respectful when dealing with other members as well as with people outside the Pywinauto recorder community.
- **Be careful in the words that you choose.** We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren't acceptable. This includes, but is not limited to:
  - Violent threats or language directed against another person.
  - Discriminatory jokes and language.
  - Posting sexually explicit or violent material.
  - Posting (or threatening to post) other people's personally identifying information ("doxing").
  - Personal insults, especially those using racist or sexist terms.
  - Unwelcome sexual attention.
  - Advocating for, or encouraging, any of the above behavior.
  - Repeated harassment of others. In general, if someone asks you to stop, then stop.
- **When we disagree, try to understand why.** Disagreements, both social and technical, happen all the time and Pywinauto recorder is no exception. It is important that we resolve disagreements and differing views constructively. Remember that we're different. The strength of Pywinauto recorder comes from its varied community, people from a wide range of backgrounds. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn't mean that they're wrong. Don't forget that it is human to err and blaming each other doesn't get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

Original text courtesy of the Speak Up! project. This version was adopted from the Django Code of Conduct.

PYTHON MODULE INDEX

p

INDEX