dtactions Documentation

Release 1.1.0

Quintijn Hoogenboom

Sep 20, 2021

CONTENTS:

This branch has not been published yet via pip, is still in development.

CHAPTER

ONE

MODULES

This section includes the available modules in DT actions.

1.1 natlinkclipboard

The `natlinkclipboard` module offers easy access to and manipulation of the Windows clipboard. The `Clipboard` class forms the core of this module. Each instance of this class is a container with a structure similar to the system clipboard, mapping content formats to content data.

1.1.1 Clipboard class

class Clipboard(*contents=None*, *text=None*, *from_system=False*, *save_clear=False*, *debug=None*)

Clipboard class, manages getting and setting the windows clipboard

classmethod Get_clipboard_formats()

returns a list of format types of current clipboard

This is mainly meant for debugging purposes.

classmethod Get_folderinfo(*waiting_time=0.05*)

returns a tuple of file/folder info of selected files or folders

As alias use Get_folderinfo, Get_hdrop or get_system_hdrop

win32con.CF_HDROP is the parameter for calling this type of clipboard data

classmethod Get_hdrop(*waiting_time=0.05*)

returns a tuple of file/folder info of selected files or folders

As alias use Get_folderinfo, Get_hdrop or get_system_hdrop

win32con.CF_HDROP is the parameter for calling this type of clipboard data

classmethod Get_text()

get text from the clipboard

from natlinkclipboard import Clipboard

simply call: text = Clipboard.get_system_text()

as alias also Get_text can be used, so: text = Clipboard.Get_text()

classmethod Set_text(*content*)

set text to the clipboard

First the clipboard is emptied.

This method fails when not in elevated mode.

As alias, you can also call: Clipboard.Set_text("abacadabra")

Set_text_and_paste(*t*)
a one shot function to past text back into the application

clear_clipboard()
Empty the clipboard and clear the internal clipboard data

assume the clipboard is open

this will be done at init phase with save_clear == True

copy_from_system(*formats=None*, *save_clear=False*, *waiting_interval=None*, *waiting_iterations=None*)
Copy the Windows system clipboard contents into this instance.

Arguments:

- *formats* (iterable, default: None) – if not None, only the given content formats will be retrieved. If None, all available formats will be retrieved.
- *save_clear* (boolean, default: False) – if true, the Windows system clipboard will be saved in self._backup, and cleared after its contents have been retrieved. Will be restored from self._backup when the instance is destroyed. If false contents are retrieved in self._contents

copy_to_system(*data=None*, *clear=True*)
Copy the contents of this instance to the Windows clipboard

Arguments: - data: text or dict of clipboard items (format, content) pairs

- *clear* (boolean, default: True) – if true, the Windows system clipboard will be cleared before this instance's contents are transferred.

get_folderinfo(*waiting_interval=None*, *waiting_iterations=None*)
Retrieve this instance's folderinfo (also hdrop)

do a copy_from_system automatically

This should be a tuple of valid paths. The paths are not checked.

If no valid info, return None

get_format(*format*)
Retrieved this instance's content for the given *format*.

Arguments:

- *format* (int) – the clipboard format to retrieve.

If the given *format* is not available, a *ValueError* is raised.

get_hdrop(*waiting_interval=None*, *waiting_iterations=None*)
Retrieve this instance's folderinfo (also hdrop)

do a copy_from_system automatically

This should be a tuple of valid paths. The paths are not checked.

If no valid info, return None

classmethod get_system_folderinfo(*waiting_time=0.05*)
returns a tuple of file/folder info of selected files or folders

As alias use Get_folderinfo, Get_hdrop or get_system_hdrop

win32con.CF_HDROP is the parameter for calling this type of clipboard data

classmethod get_system_hdrop(*waiting_time=0.05*)
returns a tuple of file/folder info of selected files or folders

As alias use Get_folderinfo, Get_hdrop or get_system_hdrop

win32con.CF_HDROP is the parameter for calling this type of clipboard data

classmethod get_system_text()
get text from the clipboard

from natlinkclipboard import Clipboard

simply call: text = Clipboard.get_system_text()

as alias also Get_text can be used, so: text = Clipboard.Get_text()

get_text(*waiting_interval=None*, *waiting_iterations=None*, *replaceNullChar=True*)
get the text (mostly unicode) contents of the clipboard

This method first does a copy from system.

If no text content available, return “”

has_format(*format*)
Determine whether this instance has content for the given format

Arguments:

- *format* (int) – the clipboard format to look for.

has_text(*waiting_interval=None*, *waiting_iterations=None*)
Determine whether this instance has text content.

restore()
restore the _backup to the system clipboard

save_sequence_number()
get the Clipboard Sequence Number and store in instance

It is set in self.current_sequence_number, no return

classmethod set_system_text(*content*)
set text to the clipboard

First the clipboard is emptied.

This method fails when not in elevated mode.

As alias, you can also call: Clipboard.Set_text(“abacadabra”)

1.1.2 Utility functions

OpenClipboardCautious(*nToTry=4*, *waiting_time=0.1*)
sometimes, wait a little before you can open the clipboard…

1.2 sendkeys

The `sendkeys` function sends keystrokes to the foreground window.

The format that is used in Unimacro and Vocola is translated into Dragonfly format, and the `dragonfly.actions.action_key.Key` class performs the actions.

This replaces the `natlink.playString` function of Natlink and the `SendInput` of Vocola.

At top of module insert:

```
from dtactions.sendkeys import sendkeys
```

And then in the appropriate place in the code:

```
sendkeys("keystrokes")
```

1.2.1 sendkeys module

1.3 monitorfunctions

The `monitorfunctions` module provides various functions related to monitors and windows. Windows can be positioned and resized on a monitor, or be moved to another monitor.

Windows only.

1.4 messagefunctions

The `messagefunctions` module provides various functions that can identify the foreground hndle of an open application.

Windows only.

CHAPTER

TWO

PROJECT

2.1 Documentation

The documentation for DT Actions is written in reStructuredText format. ReStructuredText is similar to the Markdown format. If you are unfamiliar with the format, the reStructuredText primer might be a good starting point.

The Sphinx documentation engine and Read the Docs are used to generate documentation from the *.rst* files in the *documentation/* folder. Docstrings in the source code are included in a semi-automatic way through use of the sphinx.ext.autodoc extension.

To build the documentation locally, install Sphinx and any other documentation requirements:

```
$ cd documentation
$ pip install -r requirements.txt
```

Then run the following command on Windows to build the documentation:

```
$ make.bat html
```

Or use the Makefile on other systems:

```
$ make html
```

If there were no errors during the build process, open the *_build/html/index.html* file in a web browser. Make changes, rebuild the documentation and reload the doc page(s) in your browser as you go.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX