pyewmh Documentation

Release 0.1.6

parcouss

December 21, 2016

Contents

Contents:

CHAPTER 1

The ewmh python module

This module intends to provide an implementation of Extended Window Manager Hints, based on the Xlib modules for python.

See the freedesktop.org specification for more information.

1.1 EWMH class

class ewmh.ewmh.**EWMH**(*_display=None*, *root=None*)

This class provides the ability to get and set properties defined by the EWMH spec.

Each property can be accessed in two ways. For example, to get the active window:

```
win = ewmh.getActiveWindow()
# or: win = ewmh.getProperty('_NET_ACTIVE_WINDOW')
```

Similarly, to set the active window:

```
ewmh.setActiveWindow(myWindow)
# or: ewmh.setProperty('_NET_ACTIVE_WINDOW', myWindow)
```

When a property is written, don't forget to really send the notification by flushing requests:

```
ewmh.display.flush()
```

Parameters

- **_display** – the display to use. If not given, Xlib.display.Display() is used.
- **root** – the root window to use. If not given, self.display.screen().root is used.

NET_WM_ACTIONS = ('_NET_WM_ACTION_MOVE', '_NET_WM_ACTION_RESIZE', '_NET_WM_ACTION_M

List of strings representing all known window actions.

NET_WM_STATES = ('_NET_WM_STATE_MODAL', '_NET_WM_STATE_STICKY', '_NET_WM_STATE_MAXI

List of strings representing all known window states.

NET_WM_WINDOW_TYPES = ('_NET_WM_WINDOW_TYPE_DESKTOP', '_NET_WM_WINDOW_TYPE_DOCK

List of strings representing all known window types.

getActiveWindow()

Get the current active (toplevel) window or None (property _NET_ACTIVE_WINDOW)

Returns Window object or None

getClientList()
Get the list of windows maintained by the window manager for the property _NET_CLIENT_LIST.
Returns list of Window objects

getClientListStacking()
Get the list of windows maintained by the window manager for the property _NET_CLIENT_LIST_STACKING.
Returns list of Window objects

getCurrentDesktop()
Get the current desktop number (property _NET_CURRENT_DESKTOP)
Returns int

getDesktopGeometry()
Get the desktop geometry (property _NET_DESKTOP_GEOMETRY) as an array of two integers [width, height].
Returns [int, int]

getDesktopViewPort()
Get the current viewports of each desktop as a list of [x, y] representing the top left corner (property _NET_DESKTOP_VIEWPORT).
Returns list of [int, int]

getNumberOfDesktops()
Get the number of desktops (property _NET_NUMBER_OF_DESKTOPS).
Returns int

getProperty(*prop*, **args*, ***kwargs*)
Get the value of a property. See the corresponding method for the required arguments. For example, for the property _NET_WM_STATE, look for *getWmState()*

getReadableProperties()
Get all the readable properties' names

getShowingDesktop()
Get the value of "showing the desktop" mode of the window manager (property _NET_SHOWING_DESKTOP). 1 means the mode is activated, and 0 means deactivated.
Returns int

getWmAllowedActions(*win*, *str=False*)
Get the list of allowed actions for the given window (property _NET_WM_ALLOWED_ACTIONS).
Parameters
- **win** – the window object
- **str** – True to get a list of string allowed actions instead of int

Returns list of (int|str)

getWmDesktop(*win*)
Get the current desktop number of the given window (property _NET_WM_DESKTOP).
Parameters **win** – the window object
Returns int

getWmName(*win*)
Get the property _NET_WM_NAME for the given window as a string.
Parameters **win** – the window object
Returns str

getWmPid(*win*)
Get the pid of the application associated to the given window (property _NET_WM_PID)

Parameters **win** – the window object

getWmState(*win*, *str=False*)
Get the list of states of the given window (property _NET_WM_STATE).
Parameters
- **win** – the window object
- **str** – True to get a list of string states instead of int

Returns list of (int|str)

getWmVisibleName(*win*)
Get the property _NET_WM_VISIBLE_NAME for the given window as a string.
Parameters **win** – the window object
Returns str

getWmWindowType(*win*, *str=False*)
Get the list of window types of the given window (property _NET_WM_WINDOW_TYPE).
Parameters
- **win** – the window object
- **str** – True to get a list of string types instead of int

Returns list of (int|str)

getWorkArea()
Get the work area for each desktop (property _NET_WORKAREA) as a list of [x, y, width, height]
Returns a list of [int, int, int, int]

getWritableProperties()
Get all the writable properties names

setActiveWindow(*win*)
Set the given window active (property _NET_ACTIVE_WINDOW)
Parameters **win** – the window object

setCloseWindow(*win*)
Close the given window (property _NET_CLOSE_WINDOW)
Parameters **win** – the window object

setCurrentDesktop(*i*)
Set the current desktop (property _NET_CURRENT_DESKTOP).
Parameters **i** – the desired desktop number

setDesktopGeometry(*w*, *h*)
Set the desktop geometry (property _NET_DESKTOP_GEOMETRY)
Parameters
- **w** – desktop width
- **h** – desktop height

setDesktopViewport(*w*, *h*)
Set the viewport size of the current desktop (property _NET_DESKTOP_VIEWPORT)
Parameters
- **w** – desktop width
- **h** – desktop height

setMoveResizeWindow(*win*, *gravity=0*, *x=None*, *y=None*, *w=None*, *h=None*)
Set the property _NET_MOVERESIZE_WINDOW to move or resize the given window. Flags are automatically calculated if x, y, w or h are defined.
Parameters
- **win** – the window object
- **gravity** – gravity (one of the Xlib.X.*Gravity constant or 0)

- **x** – int or None
- **y** – int or None
- **w** – int or None
- **h** – int or None

setNumberOfDesktops(*nb*)
Set the number of desktops (property _NET_NUMBER_OF_DESKTOPS).
Parameters **nb** – the number of desired desktops

setProperty(*prop*, **args*, ***kwargs*)
Set the value of a property by sending an event on the root window. See the corresponding method for the required arguments. For example, for the property _NET_WM_STATE, look for *setWmState()*

setShowingDesktop(*show*)
Set/unset the mode Showing desktop (property _NET_SHOWING_DESKTOP)
Parameters **show** – 1 to set the desktop mode, else 0

setWmDesktop(*win*, *i*)
Move the window to the desired desktop by changing the property _NET_WM_DESKTOP.
Parameters
- **win** – the window object
- **i** – desired desktop number

setWmName(*win*, *name*)
Set the property _NET_WM_NAME
Parameters
- **win** – the window object
- **name** – desired name

setWmState(*win*, *action*, *state*, *state2=0*)
Set/unset one or two state(s) for the given window (property _NET_WM_STATE).
Parameters
- **win** – the window object
- **action** – 0 to remove, 1 to add or 2 to toggle state(s)
- **state** (int or str (see *NET_WM_STATES*)) – a state
- **state2** (int or str (see *NET_WM_STATES*)) – a state or 0

setWmVisibleName(*win*, *name*)
Set the property _NET_WM_VISIBLE_NAME
Parameters
- **win** – the window object
- **name** – desired visible name

1.2 Examples

These examples are tested on gnome.

Exemple to set the active window in fullscreen mode:

```
from ewmh import EWMH
ewmh = EWMH()

# get the active window
win = ewmh.getActiveWindow()

# list all possible names states:
```

```
# print EWMH.NET_WM_STATES

# set the state on win
ewmh.setWmState(win, 1, '_NET_WM_STATE_FULLSCREEN')

# flush request
ewmh.display.flush()
```

Exemple to move every iceweasel windows on desktop 2:

```
from ewmh import EWMH
ewmh = EWMH()

# get every displayed windows
wins = ewmh.getClientList()

# get every iceweasel windows, by looking their class name:
icewins = filter(lambda w: w.get_wm_class()[1] == 'Iceweasel', wins)

# move them to desktop 2 (desktop numbering starts from 0):
for w in icewins:
  ewmh.setWmDesktop(w, 1)

# flush requests
ewmh.display.flush()
```

Example trying to close every windows on desktop 2:

```
from ewmh import EWMH
ewmh = EWMH()

# get every displayed windows on desktop 2:
wins = filter(lambda w: ewmh.getWmDesktop(w) == 1, ewmh.getClientList())

# trying to close them:
for w in wins:
  ewmh.setCloseWindow(w)

# flush requests
ewmh.display.flush()
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

e

E

G

N

S