PyXDG Documentation

Release 0.23

Sergey Kuleshov, Heinrich Wendel, Thomas Kluyver

Mar 23, 2017

Contents

PyXDG is a Python library supporting various freedesktop standards.

Contents:

CHAPTER 1

Base Directories

The XDG Base Directory specification provides standard locations to store application data, configuration, and cached data.

Data directories

xdg.BaseDirectory.**save_data_path**(**resource*)
: Ensure `$XDG_DATA_HOME/<resource>/` exists, and return its path. 'resource' should normally be the name of your application or a shared resource. Use this when saving or updating application data.

xdg.BaseDirectory.**load_data_paths**(**resource*)
: Returns an iterator which gives each directory named 'resource' in the application data search path. Information provided by earlier directories should take precedence over later ones.

xdg.BaseDirectory.**xdg_data_home**
: $XDG_DATA_HOME or the default, `~/.local/share`

xdg.BaseDirectory.**xdg_data_dirs**
: A list of directory paths in which application data may be stored, in preference order.

Configuration directories

xdg.BaseDirectory.**save_config_path**(**resource*)
: Ensure `$XDG_CONFIG_HOME/<resource>/` exists, and return its path. 'resource' should normally be the name of your application. Use this when saving configuration settings.

xdg.BaseDirectory.**load_config_paths**(**resource*)
: Returns an iterator which gives each directory named 'resource' in the configuration search path. Information provided by earlier directories should take precedence over later ones, and the user-specific config dir comes first.

xdg.BaseDirectory.**load_first_config**(**resource*)
Returns the first result from load_config_paths, or None if there is nothing to load.

xdg.BaseDirectory.**xdg_config_home**
$XDG_CONFIG_HOME or the default, `~/.config`

xdg.BaseDirectory.**xdg_config_dirs**
A list of directory paths in which configuration may be stored, in preference order.

Cache directory

xdg.BaseDirectory.**save_cache_path**(**resource*)
Ensure `$XDG_CACHE_HOME/<resource>/` exists, and return its path. 'resource' should normally be the name of your application or a shared resource.

xdg.BaseDirectory.**xdg_cache_home**
$XDG_CACHE_HOME or the default, `~/.cache`

Runtime directory

xdg.BaseDirectory.**get_runtime_dir**(*strict=True*)
Returns the value of $XDG_RUNTIME_DIR, a directory path.

This directory is intended for 'user-specific non-essential runtime files and other file objects (such as sockets, named pipes, ...)', and 'communication and synchronization purposes'.

As of late 2012, only quite new systems set $XDG_RUNTIME_DIR. If it is not set, with `strict=True` (the default), a KeyError is raised. With `strict=False`, PyXDG will create a fallback under /tmp for the current user. This fallback does *not* provide the same guarantees as the specification requires for the runtime directory.

The strict default is deliberately conservative, so that application developers can make a conscious decision to allow the fallback.

New in version 0.25.

CHAPTER 2

Desktop Entries

XDG Desktop Entry Specification

class xdg.DesktopEntry.**DesktopEntry**(*filename=None*)
Class to parse and validate Desktop Entries

__init__(*filename=None*)
Create a new DesktopEntry.

If filename exists, it will be parsed as a desktop entry file. If not, or if filename is None, a blank DesktopEntry is created.

new(*filename*)
Make this instance into a new, blank desktop entry.

If filename has a .desktop extension, Type is set to Application. If it has a .directory extension, Type is Directory. Other extensions will cause *ParsingError* to be raised.

parse(*file*)
Parse a desktop entry file.

This can raise *ParsingError*, *DuplicateGroupError* or *DuplicateKeyError*.

validate(*report='All'*)
Validate the contents, raising *ValidationError* if there is anything amiss.

report can be 'All' / 'Warnings' / 'Errors'

findTryExec()
Looks in the PATH for the executable given in the TryExec field.

Returns the full path to the executable if it is found, None if not. Raises *NoKeyError* if TryExec is not present.

New in version 0.26.

getCategories()
getComment()
getExec()
getGenericName()

getHidden()
getIcon()
getMimeTypes()
getMiniIcon()
getName()
getNoDisplay()
getNotShowIn()
getOnlyShowIn()
getPath()
getProtocols()
getStartupNotify()
getStartupWMClass()
getTerminal()
getTerminalOptions()
getTryExec()
getType()
getURL()
getVersionString()

Convenience methods to get the values of specific fields. If the field is missing, these will simply return an empty or zero value. There are similar methods for deprecated and KDE specific keys, but these are not listed here.

CHAPTER 3

Desktop Menu

XDG Desktop Menu specification

xdg.Menu.**parse**(*filename=None*, *debug=False*)
Helper function. Equivalent to calling xdg.Menu.XMLMenuBuilder().parse(filename)

class xdg.Menu.**Menu**
Menu containing sub menus under menu.Entries

Contains both Menu and MenuEntry items.

getComment()
Returns the menu's comment text.

getEntries(*show_hidden=False*)
Interator for a list of Entries visible to the user.

getGenericName()
Returns the menu's generic name.

getIcon()
Returns the menu's icon, filename or simple name

getMenu(*path*)
Searches for a Menu with a given path.

getMenuEntry(*desktopfileid*, *deep=False*)
Searches for a MenuEntry with a given DesktopFileID.

getName()
Returns the menu's localised name.

getPath(*org=False*, *toplevel=False*)
Returns this menu's path in the menu structure.

class xdg.Menu.**MenuEntry**(*filename*, *dir=''*, *prefix=''*)
Wrapper for 'Menu Style' Desktop Entries

DesktopEntry
The *xdg.DesktopEntry.DesktopEntry* instance holding the data for this entry.

getDir()
Return the directory containing the desktop entry file.

getType()
Return the type of MenuEntry, System/User/Both

save()
Save any changes to the desktop entry.

CHAPTER 4

Icon Themes

XDG Icon Theme specification

xdg.IconTheme.**getIconPath**(*iconname, size=None, theme=None, extensions=['png', 'svg', 'xpm']*)
Get the path to a specified icon.

size : Icon size in pixels. Defaults to `xdg.Config.icon_size`.

theme : Icon theme name. Defaults to `xdg.Config.icon_theme`. If the icon isn't found in the specified theme, it will be looked up in the basic 'hicolor' theme.

extensions : List of preferred file extensions.

Example:

```
>>> getIconPath("inkscape", 32)
'/usr/share/icons/hicolor/32x32/apps/inkscape.png'
```

xdg.IconTheme.**getIconData**(*path*)
Retrieve the data from the .icon file corresponding to the given file. If there is no .icon file, it returns None.

Example:

```
getIconData("/usr/share/icons/Tango/scalable/places/folder.svg")
```

class xdg.IconTheme.**IconData**
Class to parse and validate IconData Files

getAttachPoints()
Retrieve the anchor points for overlays & emblems from the icon data, as a list of co-ordinate pairs, if they are specified.

getDisplayName()
Retrieve the display name from the icon data, if one is specified.

getEmbeddedTextRectangle()
Retrieve the embedded text rectangle from the icon data as a list of numbers (x0, y0, x1, y1), if it is specified.

CHAPTER 5

Shared MIME-info Database

XDG Shared MIME-info Database specification

Finding a file's Mime type

Example:

```
>>> Mime.get_type2('/path/to/foo.zip')
MIMEtype('application', 'zip')
```

xdg.Mime.**get_type2**(*path*, *follow=True*)
Find the MIMEtype of a file using the XDG recommended checking order.

This first checks the filename, then uses file contents if the name doesn't give an unambiguous MIMEtype. It can also handle special filesystem objects like directories and sockets.

Parameters

- **path** – file path to examine (need not exist)
- **follow** – whether to follow symlinks

Return type *MIMEtype*

New in version 1.0.

xdg.Mime.**get_type_by_name**(*path*)
Returns type of file by its name, or None if not known

xdg.Mime.**get_type_by_contents**(*path*, *max_pri=100*, *min_pri=0*)
Returns type of file by its contents, or None if not known

xdg.Mime.**get_type_by_data**(*data*, *max_pri=100*, *min_pri=0*)
Returns type of the data, which should be bytes.

xdg.Mime.**get_type**(*path*, *follow=True*, *name_pri=100*)
Returns type of file indicated by path.

This function is *deprecated* - *get_type2()* is more accurate.

Parameters

- **path** – pathname to check (need not exist)
- **follow** – when reading file, follow symbolic links
- **name_pri** – Priority to do name matches. 100=override magic

This tries to use the contents of the file, and falls back to the name. It can also handle special filesystem objects like directories and sockets.

Installing Mime data

xdg.Mime.**install_mime_info**(*application*, *package_file*)
Copy 'package_file' as `~/.local/share/mime/packages/<application>.xml`. If package_file is None, install `<app_dir>/<application>.xml`. If already installed, does nothing. May overwrite an existing file with the same name (if the contents are different)

MIMEtype objects

class xdg.Mime.**MIMEtype**
Class holding data about a MIME type.

Calling the class will return a cached instance, so there is only one instance for each MIME type. The name can either be passed as one part ('text/plain'), or as two ('text', 'plain').

Changed in version 1.0: The class now takes care of caching; call *lookup()* in earlier versions.

media
e.g. 'text'

subtype
e.g. 'plain'

get_comment()
Returns comment for current language, loading it if needed.

New in version 0.25: *MIMEtype.canonical()* and *MIMEtype.inherits_from()*.

canonical()
Returns the canonical MimeType object if this is an alias.

inherits_from()
Returns a set of Mime types which this inherits from.

xdg.Mime.**lookup**(*media*, *subtype=None*)
Get the MIMEtype object for the given type.

This remains for backwards compatibility; calling MIMEtype now does the same thing.

The name can either be passed as one part ('text/plain'), or as two ('text', 'plain').

Miscellaneous

xdg.Mime.**get_extensions**(*mimetype*)
Retrieve the set of filename extensions matching a given MIMEtype.

Extensions are returned without a leading dot, e.g. 'py'. If no extensions are registered for the MIMEtype, returns an empty set.

The extensions are stored in a cache the first time this is called.

New in version 1.0.

xdg.Mime.**is_text_file**(*path*)
Guess whether a file contains text or binary data.

Heuristic: binary if the first 32 bytes include ASCII control characters. This rule may change in future versions.

New in version 1.0.

CHAPTER 6

Recently used files

XDG recent file storage specification

class xdg.RecentFiles.**RecentFiles**

addFile(*item*, *mimetype*, *groups=None*, *private=False*)
Add a recently used file.

item should be the URI of the file, typically starting with `file:///`.

deleteFile(*item*)
Remove a recently used file, by URI, from the list.

getFiles(*mimetypes=None*, *groups=None*, *limit=0*)
Get a list of recently used files.

The parameters can be used to filter by mime types, by group, or to limit the number of items returned. By default, the entire list is returned, except for items marked private.

parse(*filename=None*)
Parse a list of recently used files.

filename defaults to `~/.recently-used`.

write(*filename=None*)
Write the list of recently used files to disk.

If the instance is already associated with a file, filename can be omitted to save it there again.

class xdg.RecentFiles.**RecentFile**

URI
The URI of the file; typically starts with `file:///`.

MimeType
A Mime type, such as 'text/plain'.

Timestamp
Unix timestamp of when the file was added to the list.

Private
Boolean indicating whether the entry is private, meaning that it will only be shown if it is in a selected group.

Groups
A list of groups this entry belongs to.

CHAPTER 7

Exceptions

Exception Classes for the xdg package

exception xdg.Exceptions.**DuplicateGroupError**(*group*, *file*)
Raised when the same key occurs twice in an INI-style file.

Attributes are .group and .file.

exception xdg.Exceptions.**DuplicateKeyError**(*key*, *group*, *file*)
Raised when the same key occurs twice in an INI-style file.

Attributes are .key, .group and .file.

exception xdg.Exceptions.**Error**(*msg*)
Base class for exceptions defined here.

exception xdg.Exceptions.**NoGroupError**(*group*, *file*)
Raised when trying to access a nonexistant group in an INI-style file.

Attributes are .group and .file.

exception xdg.Exceptions.**NoKeyError**(*key*, *group*, *file*)
Raised when trying to access a nonexistant key in an INI-style file.

Attributes are .key, .group and .file.

exception xdg.Exceptions.**NoThemeError**(*theme*)
Raised when trying to access a nonexistant icon theme.

The name of the theme is the .theme attribute.

exception xdg.Exceptions.**ParsingError**(*msg*, *file*)
Raised when a file cannot be parsed.

The filename is the .file attribute.

exception xdg.Exceptions.**ValidationError**(*msg*, *file*)
Raised when a file fails to validate.

The filename is the .file attribute.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

x

Index

Symbols

A

C

D

E

F

G