context$_{menu}$

Release 1.0.0

Aug 21, 2023

Contents:

1 context_menu **3**

1.1 context_menu package 3

1.1.1 Submodules 3

1.1.2 context_menu.example module 3

1.1.3 context_menu.linux_menus module 3

1.1.4 context_menu.menus module 5

1.1.5 context_menu.windows_menus module 6

1.1.6 Module contents 7

Python Module Index **9**

Index **11**

The most important information is on how **menus.py** works. Basic usage is as follows:

1. Import the **ContextCommand** and **ContextMenu** Class from menus.py
2. Create your menu and add sub_items using the **add_items()** command
   - You can nest this functionality (see example below)
   - For commands, either pass a command in a string or a function without the "()"
3. Compile the model.

```
def foo2(filenames):
    print('foo2')
    print(filenames)
    input()

def foo3(filenames):
    print('foo3')
    print(filenames)
    input()


if __name__ == '__main__':
    from context_menu import menus

    cm = menus.ContextMenu('Foo menu', type='DIRECTORY_BACKGROUND')
    cm.add_items([
        menus.ContextCommand('Foo One', command='echo hello > example.txt'),
        menus.ContextCommand('Foo Two', python=foo2),
        menus.ContextCommand('Foo Three', python=foo3)
    ])
    cm.compile()
```

A **ContextCommand** is the item that is selectable on a menu, so the entry that actually runs a command. A **ContextMenu** simply holds the commands. Menus can be nested, so you can create subsubsubmenus if you really wanted to. Simply add them to a menu as you would a command. See examples for more information.

You'll probably find some useful methods in the windows sub package, such as deleting keys and all their subkeys from the registry.

CHAPTER 1

context_menu

1.1 context_menu package

1.1.1 Submodules

1.1.2 context_menu.example module

1.1.3 context_menu.linux_menus module

class context_menu.linux_menus.**CodeBuilder**(*name: str, body_commands: list[str], script_dirs: list[str], funcs: list[str], imports: list[str], type: ActivationType | str*)

Bases: object

The CodeBuilder class is used for generating the final python file for the Linux menus.

build_imports() → str
Creates the header of necessary imports.

Handled automatically by compile.

build_script_dirs() → str
Creates the header of necessary path configurations.

Adds all the 'sys.path.appends' in order to immport the classes and functions.

Handled automatically by compile.

compile() → str
Creates the code file.

class context_menu.linux_menus.**ExistingCode**
Bases: enum.Enum

Preset values important for metaprogramming.

BACKGROUND_ITEMS = '\tdef get_background_items(self, *args):\n\t\tfiles = args[-1]'

CLASS_TEMPLATE = '\nclass {}MenuProvider(GObject.GObject, Nautilus.MenuProvider):\n\td

CODE_HEAD = "\nimport gi\nimport sys\nimport os\n\ntry:\n\tgi.require_version('Nautilu

COMMAND_HANDLER_TEMPLATE = "\n\tdef {}(self, menu, files):\n\t\tfilepath = [unquote(su

FILE_ITEMS = '\tdef get_file_items(self, *args):\n\t\tfiles = args[-1]'

MENU_ITEM = 'menuitem{} = Nautilus.MenuItem(name = "ExampleMenuProvider::{}", label="{

METHOD_HANDLER_TEMPLATE = '\n\tdef {}(self, menu, files):\n\t\tfilenames = [unquote(su

SUB_MENU = 'submenu{} = Nautilus.Menu()'

class context_menu.linux_menus.**NautilusMenu**(*name: str, sub_items: list[ItemType], type: ActivationType | str*)

Bases: object

append_item(*menu: str*, *item: str*) → str
Creates a necessary body_command.

build_script() → str
Finishes and returns the full code.

build_script_body(*name: str, items: list[ItemType]*) → None
Builds the body commands of the script.

compile() → None
Creates the code, creates a file, and moves it to the correct location.

connect(*item: str*, *func: str*) → str
Creates a necessary body_command.

create_path(*path: str*, *dir: str*) → str
Creates a path to directory. Creates all sub-directories

generate_command_func(*command: str*) → context_menu.linux_menus.Variable
Generates a command attached to a python function

generate_item(*name: str*) → context_menu.linux_menus.Variable
Generates a nautilus command variable.

generate_menu() → context_menu.linux_menus.Variable
Generates a nautilus menu variable.

generate_mod_command_func(*command: str, command_vars: list[CommandVar]*) → Variable
Generates a command attached to a python function that allows special variables.

generate_python_func(*class_origin: str*, *class_func: str*, *params: str*) → context_menu.linux_menus.Variable
Generates a command attached to a python function

get_next_item() → str
Very niche, required in other methods.

set_submenu(*item: str*, *menu: str*) → str
Creates a necessary body_command.

class context_menu.linux_menus.**Variable**(*name: str*, *code: str*)

Bases: object

Very simple class with no methods to help simplify the code.

context_menu.linux_menus.**command_var_format**(*item: str*) → str
Converts a python string to a value for a command

context_menu.linux_menus.**remove_linux_menu**(*name*) → None

1.1.4 context_menu.menus module

class context_menu.menus.**ContextCommand**(*name: str*, *command: str | None = None*, *python: FunctionType | None = None*, *params: str = ''*, *command_vars: list[CommandVar] | None = None*)

Bases: object

The general command class.

A command is an executable entry in a context-menu, where menus hold other commands.

Name = Name of the command Command = command to be ran from the shell python = function to be ran params = any other parameters to be passed command_vars = to help with the command

get_method_info() → MethodInfo
Extremely important for making shell commands to run python code.

Returns a tuple (function name, function file name, path to function directory)

get_platform_command()
Will be used in future changes.

class context_menu.menus.**ContextMenu**(*name: str*, *type: ActivationType | str | None = None*)

Bases: object

The general menu class. This class generalizes the menus and eventually passes the correct values to the platform-specifically menus.

add_items(*items: list[ItemType]*) → None
Adds a list of items to the current menu.

compile() → None
Recognizes the current platform and passes information to the respective menu. Creates the actual menu.

class context_menu.menus.**FastCommand**(*name: str*, *type: ActivationType | str*, *command: str | None = None*, *python: FunctionType | None = None*, *params: str = ''*, *command_vars: list[CommandVar] | None = None*)

Bases: object

Used for fast creation of a command. Good if you don't want to get too involved and just jump start a program.

Extremely similar methods to other classes, only slightly modified. View the documentation of the above classes for info on these methods.

compile() → None

get_method_info() → MethodInfo

context_menu.menus.**removeMenu**(*name: str*, *type: ActivationType | str*) → None
Removes a menu/command entry from a context menu.

Requires the name of the menu and type of the menu

1.1.5 context_menu.windows_menus module

class context_menu.windows_menus.**FastRegistryCommand**(*name: str, type: ActivationType | str, command: str, python: FunctionType, params: str, command_vars: list[CommandVar]*)

Bases: object

Fast command class.

Everything is identical to either the RegistryMenu class or code in the menus file

compile() → None

get_method_info() → MethodInfo

class context_menu.windows_menus.**RegistryMenu**(*name: str, sub_items: list[ItemType], type: str*)

Bases: object

Class to convert the general menu from menus.py to a Windows-specific menu.

compile(*items: list[ItemType] | None = None*, *path: str | None = None*) → None
Used to create the menu. Recursively iterates through each element in the top level menu.

create_command(*name: str*, *path: str*, *command: str*) → None
Creates a key with a command subkey with the 'name' and 'command', at path 'path'.

create_menu(*name: str*, *path: str*) → str
Creates a menu with the given name and path.

Used in the compile method.

context_menu.windows_menus.**command_preset_format**(*item: str*) → str
Converts a python string to an executable location.

For example, 'python' -> sys.executable

context_menu.windows_menus.**command_var_format**(*item: str*) → str
Converts a python string to a value for a command

context_menu.windows_menus.**context_registry_format**(*item: str*) → str
Converts a verbose type into a registry path.

For example, 'FILES' -> 'SoftwareClasses*shell'

context_menu.windows_menus.**create_directory_background_command**(*func_name: str*, *func_file_name: str*, *func_dir_path: str*, *params: str*) → str
Creates a registry valid command to link a context menu entry to a funtion, specifically for backgrounds(DIRECTORY_BACKGROUND, DESKTOP_BACKGROUND).

Requires the name of the function, the name of the file, and the path to the directory of the file.

context_menu.windows_menus.**create_file_select_command**(*func_name: str*, *func_file_name: str*, *func_dir_path: str*, *params: str*) → str
Creates a registry valid command to link a context menu entry to a funtion, specifically for file selection(FILES, DIRECTORY, DRIVE).

Requires the name of the function, the name of the file, and the path to the directory of the file.

context_menu.windows_menus.**create_key**(*path: str*, *hive: int = 0*) → None
Creates a key at the desired path.

context_menu.windows_menus.**create_shell_command**(*command: str*, *command_vars: list[CommandVar]*) → str
Creates a shell command and replaces '?' with the command_vars list

context_menu.windows_menus.**delete_key**(*path: str*, *hive: int = 0*) → None
Deletes the desired key and all other subkeys at the given path.

context_menu.windows_menus.**get_key_value**(*key_path: str*, *subkey_name: str*, *hive: int = 0*) → Any
Gets the value of a subkey.

context_menu.windows_menus.**join_keys**(**keys*) → str
Joins parts of a registry path.

This joins the parts with unlike os.path.join that would use / on Linux and break tests.

Parameters **keys** – parts of the registry path

Returns complete registry path

context_menu.windows_menus.**list_keys**(*path: str*, *hive: int = 0*) → list[str]
Returns a list of all the keys at a given registry path.

context_menu.windows_menus.**remove_windows_menu**(*name: str*, *type: ActivationType | str*) → None
Removes a context menu from the windows registry.

context_menu.windows_menus.**set_key_value**(*key_path: str*, *subkey_name: str*, *value: str | int*, *hive: int = 0*) → None
Changes the value of a subkey. Creates the subkey if it doesn't exist.

1.1.6 Module contents

Python Module Index

C

context_menu, 7
context_menu.linux_menus, 3
context_menu.menus, 5
context_menu.windows_menus, 6

Index

A

add_items() (*context_menu.menus.ContextMenu method*), 5
append_item() (*context_menu.linux_menus.NautilusMenu method*), 4

B

BACKGROUND_ITEMS (*context_menu.linux_menus.ExistingCode attribute*), 3
build_imports() (*context_menu.linux_menus.CodeBuilder method*), 3
build_script() (*context_menu.linux_menus.NautilusMenu method*), 4
build_script_body() (*context_menu.linux_menus.NautilusMenu method*), 4
build_script_dirs() (*context_menu.linux_menus.CodeBuilder method*), 3

C

CLASS_TEMPLATE (*context_menu.linux_menus.ExistingCode attribute*), 3
CODE_HEAD (*context_menu.linux_menus.ExistingCode attribute*), 4
CodeBuilder (*class in context_menu.linux_menus*), 3
COMMAND_HANDLER_TEMPLATE (*context_menu.linux_menus.ExistingCode attribute*), 4
command_preset_format() (*in module context_menu.windows_menus*), 6
command_var_format() (*in module context_menu.linux_menus*), 4
command_var_format() (*in module context_menu.windows_menus*), 6
compile() (*context_menu.linux_menus.CodeBuilder method*), 3
compile() (*context_menu.linux_menus.NautilusMenu method*), 4
compile() (*context_menu.menus.ContextMenu method*), 5
compile() (*context_menu.menus.FastCommand method*), 5
compile() (*context_menu.windows_menus.FastRegistryCommand method*), 6
compile() (*context_menu.windows_menus.RegistryMenu method*), 6
connect() (*context_menu.linux_menus.NautilusMenu method*), 4
context_menu (*module*), 7
context_menu.linux_menus (*module*), 3
context_menu.menus (*module*), 5
context_menu.windows_menus (*module*), 6
context_registry_format() (*in module context_menu.windows_menus*), 6
ContextCommand (*class in context_menu.menus*), 5
ContextMenu (*class in context_menu.menus*), 5
create_command() (*context_menu.windows_menus.RegistryMenu method*), 6
create_directory_background_command() (*in module context_menu.windows_menus*), 6
create_file_select_command() (*in module context_menu.windows_menus*), 6
create_key() (*in module context_menu.windows_menus*), 7
create_menu() (*context_menu.windows_menus.RegistryMenu method*), 6
create_path() (*context_menu.linux_menus.NautilusMenu method*), 4
create_shell_command() (*in module context_menu.windows_menus*), 7

D

delete_key() (*in module context_menu.windows_menus*), 7

E

ExistingCode (*class in context_menu.linux_menus*), 3

F

FastCommand (*class in context_menu.menus*), 5
FastRegistryCommand (*class in context_menu.windows_menus*), 6
FILE_ITEMS (*context_menu.linux_menus.ExistingCode attribute*), 4

G

generate_command_func() (*context_menu.linux_menus.NautilusMenu method*), 4
generate_item() (*context_menu.linux_menus.NautilusMenu method*), 4
generate_menu() (*context_menu.linux_menus.NautilusMenu method*), 4
generate_mod_command_func() (*context_menu.linux_menus.NautilusMenu method*), 4
generate_python_func() (*context_menu.linux_menus.NautilusMenu method*), 4
get_key_value() (*in module context_menu.windows_menus*), 7
get_method_info() (*context_menu.menus.ContextCommand method*), 5
get_method_info() (*context_menu.menus.FastCommand method*), 5
get_method_info() (*context_menu.windows_menus.FastRegistryCommand method*), 6
get_next_item() (*context_menu.linux_menus.NautilusMenu method*), 4
get_platform_command() (*context_menu.menus.ContextCommand method*), 5

J

join_keys() (*in module context_menu.windows_menus*), 7

L

list_keys() (*in module context_menu.windows_menus*), 7

M

MENU_ITEM (*context_menu.linux_menus.ExistingCode attribute*), 4
METHOD_HANDLER_TEMPLATE (*context_menu.linux_menus.ExistingCode attribute*), 4

N

NautilusMenu (*class in context_menu.linux_menus*), 4

R

RegistryMenu (*class in context_menu.windows_menus*), 6
remove_linux_menu() (*in module context_menu.linux_menus*), 4
remove_windows_menu() (*in module context_menu.windows_menus*), 7
removeMenu() (*in module context_menu.menus*), 5

S

set_key_value() (*in module context_menu.windows_menus*), 7
set_submenu() (*context_menu.linux_menus.NautilusMenu method*), 4
SUB_MENU (*context_menu.linux_menus.ExistingCode attribute*), 4

V

Variable (*class in context_menu.linux_menus*), 4