
Qt Dev Helper Documentation

Release 0.0.4

Sebastian Weigand

Jul 08, 2023

CONTENTS:

1	Qt Dev Helper	1
1.1	Installation	1
1.2	Features	1
1.3	Planned features	2
1.4	FAQ	2
1.5	Contributors	2
2	Installation	3
2.1	Stable release	3
2.2	From sources	3
3	Usage	5
4	Command-line Interface	7
4.1	qt-dev-helper	7
5	Configuration	11
6	Inner workings	19
6.1	qt_dev_helper	19
7	Contributing	51
7.1	Types of Contributions	51
7.2	Get Started!	52
7.3	Pull Request Guidelines	53
7.4	Tips	53
7.5	Deploying	53
8	Changelog	55
8.1	0.0.4 (2023-07-08)	55
8.2	0.0.3 (2022-09-05)	55
8.3	0.0.2 (2022-09-05)	55
8.4	0.0.1 (2022-09-04)	55
9	Indices and tables	57
	Python Module Index	59
	Index	61

QT DEV HELPER

Toolbox to help develop Qt applications, improving the usability of the existing tooling.

1.1 Installation

```
pip install qt-dev-helper
```

OR

```
conda install -c conda-forge qt-dev-helper
```

1.2 Features

- Usable as Library and/or CLI tool
- Compatible with [PEP517](#) build system ([see test case](#))
- CLI auto completion
- Project wide configuration in `pyproject.toml`
- Recursive asset compiler for Qt projects (using `uic` and `rcc`):
 - `*.ui -> *.py`
 - `*.qrc -> *.py`
 - `*.ui -> *.h`
 - `*.qrc -> *.h`
 - `*.scss -> *.qss`
- Support for multiple Qt tooling suppliers
 - PySide6-Essentials
 - qt6-applications
 - qt5-applications

- Ability to open all files in a folder in QtDesigner

1.3 Planned features

- Stand alone executable for each release (Windows)
- File watch mode
- qss injection into *.ui files
- `pre-commit` hooks

1.4 FAQ

- Q: Why is PyQt5 not supported?

A: PyQt5 only ships a python specific version of `uic` and `rcc` breaking the tool API and compatibility with cpp projects. Use the matching version of `qt5-applications` as Qt tooling supplier.

1.5 Contributors

Thanks goes out to these wonderful people ([emoji key](#)):

This project follows the [all-contributors](#) specification. Contributions of any kind are welcome!

INSTALLATION

2.1 Stable release

To install Qt Dev Helper, run this command in your terminal:

```
$ pip install qt_dev_helper
```

This is the preferred method to install Qt Dev Helper, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for Qt Dev Helper can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/s-weigand/qt-dev-helper
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/s-weigand/qt-dev-helper/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```


USAGE

To use Qt Dev Helper in a project:

```
import qt_dev_helper
```


COMMAND-LINE INTERFACE

4.1 qt-dev-helper

Collection of CLI commands to improve workflows when developing Qt GUI Applications.

```
qt-dev-helper [OPTIONS] COMMAND [ARGS]...
```

Options

--install-completion <install_completion>

Install completion for the specified shell.

Options

bash | zsh | fish | powershell | pwsh

--show-completion <show_completion>

Show completion for the specified shell, to copy it or customize the installation.

Options

bash | zsh | fish | powershell | pwsh

4.1.1 build

Build production assets from input files.

```
qt-dev-helper build [OPTIONS] [BASE_PATH]
```

Options

-c, --config <config>

Path to a config file.

-r, --recurse-folder

Recurse directories searching for files.

Default

False

-g, --generator <generator>

Code generator used to compile ui and resource files.

Options

python | cpp

--flatten-folder-structure, --no-flatten-folder-structure

Whether or not to flatten the folder structure of the ui and resource files.

--ui, --no-ui

Whether or not to build ui files from '*.ui' files.

Default

True

--ui-files-folder <ui_files_folder>

Root folder containing *.ui files.

--generated-ui-code-folder <generated_ui_code_folder>

Root folder to save code generated from *.ui files to.

--uic-args <uic_args>

Additional arguments for the uic executable, as comma separated list.

--rc, --no-rc

Whether or not to build resource files from '*.qrc' files.

Default

True

--resource-folder <resource_folder>

Root folder containing *.qrc files.

--generated-rc-code-folder <generated_rc_code_folder>

Root folder to save code generated from *.qrc files to.

--form-import, --no-form-import

Python: generate imports relative to '.'

--rcc-args <rcc_args>

Additional arguments for the rcc executable, as comma separated list.

--qss, --no-qss

Whether or not to build qss files from '*.scss' files.

Default

True

--root-sass-file <root_sass_file>

Scss stylesheet with the style for the whole application.

--root-qss-file <root_qss_file>

Qss stylesheet with the style for the whole application, generated from 'root_sass_file'.

Arguments

BASE_PATH

Optional argument

4.1.2 designer

Open *.ui files in qt-designer.

```
qt-dev-helper designer [OPTIONS] [FILES]...
```

Options

--recurse-folder, --no-recurse-folder

Whether or not to recurse directories searching for files.

Default

True

--open-files, --no-open-files

Whether or not to open files.

Default

True

Arguments

FILES

Optional argument(s)

CONFIGURATION

pydantic settings Config

Project configuration.

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
{
  "title": "Config",
  "description": "Project configuration.",
  "type": "object",
  "properties": {
    "base_path": {
      "title": "Base Path",
      "description": "Directory the config was loaded from, used to resolve_
↳relative paths.",
      "env_names": "'base_path'",
      "type": "string",
      "format": "path"
    },
    "root_sass_file": {
      "title": "Root Sass File",
      "description": "Scss stylesheet with the style for the whole application.",
      "env_names": "'root_sass_file'",
      "type": "string"
    },
    "root_qss_file": {
      "title": "Root Qss File",
      "description": "Qss stylesheet with the style for the whole application,
↳generated from 'root_sass_file'.",
      "env_names": "'root_qss_file'",
      "type": "string"
    },
    "generator": {
      "description": "Code generator used to compile ui and resource files.",
      "default": "python",
      "env_names": "'generator'",
      "allOf": [
        {
          "$ref": "#/definitions/CodeGenerators"
        }
      ]
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    ]
  },
  "flatten_folder_structure": {
    "title": "Flatten Folder Structure",
    "description": "Whether to keep the original folder structure or flatten_
→ it.",
    "default": true,
    "env_names": "'flatten_folder_structure'",
    "type": "boolean"
  },
  "ui_files_folder": {
    "title": "Ui Files Folder",
    "description": "Root folder containing *.ui files.",
    "env_names": "'ui_files_folder'",
    "type": "string"
  },
  "generated_ui_code_folder": {
    "title": "Generated Ui Code Folder",
    "description": "Root folder to save code generated from *.ui files to.",
    "env_names": "'generated_ui_code_folder'",
    "type": "string"
  },
  "uic_args": {
    "title": "Uic Args",
    "description": "Additional arguments for the uic executable.",
    "env_names": "'uic_args'",
    "type": "array",
    "items": {
      "type": "string"
    }
  },
  "form_import": {
    "title": "Form Import",
    "description": "Python: generate imports relative to '.'",
    "default": true,
    "env_names": "'form_import'",
    "type": "boolean"
  },
  "resource_folder": {
    "title": "Resource Folder",
    "description": "Root folder containing *.qrc files.",
    "env_names": "'resource_folder'",
    "type": "string"
  },
  "generated_rc_code_folder": {
    "title": "Generated Rc Code Folder",
    "description": "Root folder to save code generated from *.qrc files to.",
    "env_names": "'generated_rc_code_folder'",
    "type": "string"
  },
  "rcc_args": {
    "title": "Rcc Args",

```

(continues on next page)

(continued from previous page)

```

        "description": "Additional arguments for the rcc executable.",
        "env_names": "'rcc_args'",
        "type": "array",
        "items": {
            "type": "string"
        }
    },
    "required": [
        "base_path"
    ],
    "additionalProperties": false,
    "definitions": {
        "CodeGenerators": {
            "title": "CodeGenerators",
            "description": "Valid code generator values.",
            "enum": [
                "python",
                "cpp"
            ],
            "type": "string"
        }
    }
}

```

Fields

- *base_path* (*pathlib.Path*)
- *flatten_folder_structure* (*bool*)
- *form_import* (*bool*)
- *generated_rc_code_folder* (*str | None*)
- *generated_ui_code_folder* (*str | None*)
- *generator* (*qt_dev_helper.config.CodeGenerators*)
- *rcc_args* (*List[str]*)
- *resource_folder* (*str | None*)
- *root_qss_file* (*str | None*)
- *root_sass_file* (*str | None*)
- *ui_files_folder* (*str | None*)
- *uic_args* (*List[str]*)

Validators

- *_validate_rc_input_path* » *resource_folder*
- *_validate_rc_io* » all fields
- *_validate_style_input_path* » *root_sass_file*
- *_validate_styles_io* » all fields

- `_validate_ui_input_path` » *ui_files_folder*
- `_validate_ui_io` » all fields

field `base_path`: `Path` [Required]

Directory the config was loaded from, used to resolve relative paths.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `flatten_folder_structure`: `bool` = `True`

Whether to keep the original folder structure or flatten it.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `form_import`: `bool` = `True`

Python: generate imports relative to ‘.’

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `generated_rc_code_folder`: `str` | `None` = `None`

Root folder to save code generated from *.qrc files to.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `generated_ui_code_folder`: `str` | `None` = `None`

Root folder to save code generated from *.ui files to.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `generator`: *CodeGenerators* = `CodeGenerators.python`

Code generator used to compile ui and resource files.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`

- `_validate_ui_io`

field `rcc_args`: `List[str]` [Optional]

Additional arguments for the rcc executable.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `resource_folder`: `str` | `None` = `None`

Root folder containing *.qrc files.

Validated by

- `_validate_rc_input_path`
- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `root_qss_file`: `str` | `None` = `None`

Qss stylesheet with the style for the whole application, generated from 'root_sass_file'.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

field `root_sass_file`: `str` | `None` = `None`

Scss stylesheet with the style for the whole application.

Validated by

- `_validate_rc_io`
- `_validate_style_input_path`
- `_validate_styles_io`
- `_validate_ui_io`

field `ui_files_folder`: `str` | `None` = `None`

Root folder containing *.ui files.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_input_path`
- `_validate_ui_io`

field `uic_args`: `List[str]` [Optional]

Additional arguments for the uic executable.

Validated by

- `_validate_rc_io`
- `_validate_styles_io`
- `_validate_ui_io`

deactivate_resource_build() → None

Deactivate resource building with `build_all_assets()`.

deactivate_style_build() → None

Deactivate style building with `build_all_assets()`.

deactivate_ui_build() → None

Deactivate ui building with `build_all_assets()`.

rc_folder_paths() → Tuple[Path, Path]

Resolve paths to root style files.

Returns

Paths to `resource_folder` and `generated_rc_code_folder`.

Return type

Tuple[Path, Path]

rcc_kwargs() → *RccKwargs*

Extract keyword arguments to be used with `compile_resource_file`.

Returns

Keyword arguments for `compile_resource_file`.

Return type

RccKwargs

root_style_paths() → Tuple[Path, Path]

Resolve paths to root style files.

Returns

Paths to `root_sass_file` and `root_qss_file`.

Return type

Tuple[Path, Path]

ui_folder_paths() → Tuple[Path, Path]

Resolve paths to root style files.

Returns

Paths to `ui_files_folder` and `generated_ui_code_folder`.

Return type

Tuple[Path, Path]

uic_kwargs() → *UicKwargs*

Extract keyword arguments to be used with `compile_ui_file`.

Returns

Keyword arguments for `compile_ui_file`.

Return type

UicKwargs

update(*update_dict: Dict[str, Any], filter_none: bool = True*) → None

Update config values.

Parameters

- **update_dict** (*Dict[str, Any]*) – Dict containing updated values.
- **filter_none** (*bool*) – Whether or not to filter None values before updating. Defaults to True

INNER WORKINGS

This is the detailed documentation of the inner workings of `qt_dev_helper`.

<code>qt_dev_helper</code>	Top-level package for Qt Dev Helper.
----------------------------	--------------------------------------

6.1 qt_dev_helper

Top-level package for Qt Dev Helper.

Modules

<code>qt_dev_helper.__main__</code>	Entry point module when calling python with -m option.
<code>qt_dev_helper.cli</code>	Command Line Interface package.
<code>qt_dev_helper.config</code>	Configuration module.
<code>qt_dev_helper.qt_tools</code>	Qt implementation cross compatibility module.
<code>qt_dev_helper.transpiler</code>	Module containing functionality to transpile resources <code>*.scss</code> , <code>*.ui</code> and <code>*.qrc</code> .
<code>qt_dev_helper.utils</code>	Module containing utility functionality.

6.1.1 __main__

Entry point module when calling python with -m option.

6.1.2 cli

Command Line Interface package.

Modules

<code>qt_dev_helper.cli._cli_docs</code>	Module used to to autogenerate CLI documentation with sphinx_click.
<code>qt_dev_helper.cli.commands</code>	CLI commands package.
<code>qt_dev_helper.cli.main_app</code>	Main CLI application definition.
<code>qt_dev_helper.cli.utils</code>	CLI utility functionality.

`_cli_docs`

Module used to to autogenerate CLI documentation with sphinx_click.

commands

CLI commands package.

Modules

<code>qt_dev_helper.cli.commands.build</code>	Module containing the CLI build command implementations.
<code>qt_dev_helper.cli.commands.designer</code>	Module containing the CLI designer command implementations.

build

Module containing the CLI build command implementations.

Functions

Summary

<code>build</code>	Build production assets from input files.
--------------------	---

build


```
build(base_path: ~pathlib.Path | None = <typer.models.ArgumentInfo object>, config: ~pathlib.Path |
None = <typer.models.OptionInfo object>, recurse_folder: bool = <typer.models.OptionInfo
object>, generator: ~qt_dev_helper.config.CodeGenerators | None = <typer.models.OptionInfo
object>, flatten_folder_structure: bool | None = <typer.models.OptionInfo object>, ui: bool =
<typer.models.OptionInfo object>, ui_files_folder: ~pathlib.Path | None =
<typer.models.OptionInfo object>, generated_ui_code_folder: ~pathlib.Path | None =
<typer.models.OptionInfo object>, uic_args: str | None = <typer.models.OptionInfo object>, rc:
bool = <typer.models.OptionInfo object>, resource_folder: ~pathlib.Path | None =
<typer.models.OptionInfo object>, generated_rc_code_folder: ~pathlib.Path | None =
<typer.models.OptionInfo object>, form_import: bool | None = <typer.models.OptionInfo
object>, rcc_args: str | None = <typer.models.OptionInfo object>, qss: bool =
<typer.models.OptionInfo object>, root_sass_file: ~pathlib.Path | None =
<typer.models.OptionInfo object>, root_qss_file: ~pathlib.Path | None =
<typer.models.OptionInfo object>) → None
```

Build production assets from input files.

designer

Module containing the CLI designer command implementations.

Functions

Summary

<i>designer</i>	Open *.ui files in qt-designer.
-----------------	---------------------------------

designer

```
designer(files: ~typing.List[~pathlib.Path] = <typer.models.ArgumentInfo object>, recurse_folder:
bool = <typer.models.OptionInfo object>, open_files: bool = <typer.models.OptionInfo
object>) → None
```

Open *.ui files in qt-designer.

main_app

Main CLI application definition.

utils

CLI utility functionality.

Functions

Summary

<code>parse_optional_args_string</code>	Parse optional args string as comma separated list.
---	---

`parse_optional_args_string`

`parse_optional_args_string`(*optional_args_string*: *str* | *None*) → list[str] | None

Parse optional args string as comma separated list.

Parameters

`optional_args_string` (*str* | *None*) – Optional argument string with coma separated arguments if not None.

Returns

None or list of arguments

Return type

list[str] | None

6.1.3 config

Configuration module.

Functions

Summary

<code>_check_input_exists</code>	Check that the input path <code>base_path / input_var</code> exists.
<code>_check_symmetric_io_definition</code>	Check that <code>input_var_name</code> and <code>output_var_name</code> are both None or both not None.
<code>_str_list_factory</code>	Ensure that a list of arbitrary argument is List[str].
<code>expand_io_paths</code>	Expand relative io paths with <code>base_path</code> from config.
<code>find_config</code>	Find config file based on its name and start path, by traversing parent paths.
<code>load_config</code>	Load config from file.
<code>load_toml_config</code>	Load config from toml config file.

`_check_input_exists`

`_check_input_exists(input_var: str, input_var_name: str, base_path: Path, is_file: bool = False)`
`→ str | None`

Check that the input path `base_path / input_var` exists.

Parameters

- **input_var** (*str*) – Value of the input which should be checked for existence.
- **input_var_name** (*str*) – Variable name, used to format the error message.
- **base_path** (*Path*) – Base path the input var is relative to.
- **is_file** (*bool*) – Whether to check if the path is a valid file or folder. Defaults to `False`

Returns

Value of `input_var`

Return type

`Optional[str]`

Raises

- **ValueError** – If `is_file` is `True` and the path is not a file.
- **ValueError** – If `is_file` is `False` and the path is not a folder.

`_check_symmetric_io_definition`

`_check_symmetric_io_definition(values: Dict[str, Any], input_var_name: str, output_var_name: str) → Dict[str, Any]`

Check that `input_var_name` and `output_var_name` are both `None` or both not `None`.

Parameters

- **values** (*Dict[str, Any]*) – Dict representation of the Config.
- **input_var_name** (*str*) – Name of the input path variable.
- **output_var_name** (*str*) – Name of the output path variable.

Returns

Value of `values`

Return type

`Dict[str, Any]`

Raises

ValueError – If only one value of `input_var_name` and `output_var_name` is `None`.

`_str_list_factory`

`_str_list_factory(*args: Any) → List[str]`

Ensure that a list of arbitrary argument is List[str].

Parameters

args (*Any*) – Arbitrary arguments.

Returns

List of args cast to string.

Return type

List[str]

`expand_io_paths`

`expand_io_paths(config: Config, input_var_name: str, output_var_name: str) → Tuple[Path, Path]`

Expand relative io paths with `base_path` from config.

Parameters

- **config** ([Config](#)) – Config instance, needed to determine the base path.
- **input_var_name** (*str*) – Name of the variable holding the input path string.
- **output_var_name** (*str*) – Name of the variable holding the input path string.

Returns

Expanded input path and expanded output path.

Return type

Tuple[Path, Path]

Raises

[QtDevHelperConfigError](#) – If any of the io paths is None.

`find_config`

`find_config(start_path: Path | str | None = None, config_file_name: str = 'pyproject.toml') → Path`

Find config file based on its name and start path, by traversing parent paths.

Parameters

- **start_path** (*Optional[Union[Path, str]]*) – Path to start looking for the config file. Defaults to None which means the current dir will be used
- **config_file_name** (*str*) – Name of the config file. Defaults to “pyproject.toml”

Returns

Path of the found config file

Return type

Path

Raises

[ConfigNotFoundError](#) – If no config file could be found.

load_config

load_config(start_path: Path | str | None = None) → Config

Load config from file.

Parameters

start_path (Optional[Union[Path, str]]) – Path to start looking for the config file. Defaults to None which means the current dir will be used

Returns

Configuration instance generated from file.

Return type

Config

Raises

ConfigNotFoundError – If no config file containing ‘qt-dev-helper’ config could be found.

load_toml_config

load_toml_config(path: Path) → Config

Load config from toml config file.

Parameters

path (Path) – Path to the toml config file.

Returns

Configuration instance generate from toml definition.

Return type

Config

Raises

ConfigNotFoundError – If no config file does not contain ‘qt-dev-helper’ config.

Classes

Summary

<i>CodeGenerators</i>	Valid code generator values.
<i>RccKwargs</i>	Keyword arguments to be used with compile_resource_file.
<i>UicKwargs</i>	Keyword arguments to be used with compile_ui_file.

CodeGenerators

class `CodeGenerators`(*value*)

Valid code generator values.

Attributes Summary

<code>python</code>	
<code>cpp</code>	
<code>__module__</code>	
<code>__dict__</code>	
<code>__weakref__</code>	list of weak references to the object (if defined)
<code>_member_names_</code>	
<code>_member_map_</code>	
<code>_value2member_map_</code>	
<code>__doc__</code>	
<code>__members__</code>	

Methods Summary

<code>_generate_next_value_</code>

`_generate_next_value_`

`CodeGenerators._generate_next_value_(start, count, last_values)`

RccKwargs

class `RccKwargs`(*args, **kwargs)

Keyword arguments to be used with `compile_resource_file`.

Attributes Summary

<code>__annotations__</code>	
<code>__dict__</code>	
<code>__doc__</code>	
<code>__hash__</code>	
<code>__module__</code>	
<code>__total__</code>	
<code>__weakref__</code>	list of weak references to the object (if defined)

Methods Summary

<code>__contains__</code>	True if the dictionary has the specified key, else False.
<code>__delattr__</code>	Implement <code>delattr(self, name)</code> .
<code>__delitem__</code>	Delete <code>self[key]</code> .
<code>__dir__</code>	Default <code>dir()</code> implementation.
<code>__eq__</code>	Return <code>self==value</code> .
<code>__format__</code>	Default object formatter.
<code>__ge__</code>	Return <code>self>=value</code> .
<code>__getattribute__</code>	Return <code>getattr(self, name)</code> .
<code>__getitem__</code>	<code>x.__getitem__(y) <==> x[y]</code>
<code>__gt__</code>	Return <code>self>value</code> .
<code>__init__</code>	
<code>__init_subclass__</code>	This method is called when a class is subclassed.
<code>__iter__</code>	Implement <code>iter(self)</code> .
<code>__le__</code>	Return <code>self<=value</code> .
<code>__len__</code>	Return <code>len(self)</code> .
<code>__lt__</code>	Return <code>self<value</code> .
<code>__ne__</code>	Return <code>self!=value</code> .
<code>__new__</code>	
<code>__reduce__</code>	Helper for pickle.
<code>__reduce_ex__</code>	Helper for pickle.
<code>__repr__</code>	Return <code>repr(self)</code> .
<code>__reversed__</code>	Return a reverse iterator over the dict keys.
<code>__setattr__</code>	Implement <code>setattr(self, name, value)</code> .
<code>__setitem__</code>	Set <code>self[key]</code> to value.
<code>__sizeof__</code>	
<code>__str__</code>	Return <code>str(self)</code> .

continues on next page

Table 1 – continued from previous page

<code>__subclasshook__</code>	Abstract classes can override this to customize <code>issubclass()</code> .
<code>clear</code>	
<code>copy</code>	
<code>fromkeys</code>	Create a new dictionary with keys from iterable and values set to value.
<code>get</code>	Return the value for key if key is in the dictionary, else default.
<code>items</code>	
<code>keys</code>	
<code>pop</code>	If key is not found, d is returned if given, otherwise <code>KeyError</code> is raised
<code>popitem</code>	Remove and return a (key, value) pair as a 2-tuple.
<code>setdefault</code>	Insert key with a value of default if key is not in the dictionary.
<code>update</code>	If E is present and has a <code>.keys()</code> method, then does: for k in E: D[k] = E[k] If E is present and lacks a <code>.keys()</code> method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
<code>values</code>	

`__contains__`

`RccKwargs.__contains__(key, /)`

True if the dictionary has the specified key, else False.

`__delattr__`

`RccKwargs.__delattr__(name, /)`

Implement `delattr(self, name)`.

`__delitem__`

`RccKwargs.__delitem__(key, /)`

Delete `self[key]`.

__dir__

RccKwargs.__dir__()

Default dir() implementation.

__eq__

RccKwargs.__eq__(value, /)

Return self==value.

__format__

RccKwargs.__format__(format_spec, /)

Default object formatter.

__ge__

RccKwargs.__ge__(value, /)

Return self>=value.

__getattribute__

RccKwargs.__getattribute__(name, /)

Return getattr(self, name).

__getitem__

RccKwargs.__getitem__()

x.__getitem__(y) <==> x[y]

__gt__

RccKwargs.__gt__(value, /)

Return self>value.

__init__

RccKwargs.__init__(*args, **kwargs)

`__init_subclass__`

`RccKwargs.__init_subclass__()`

This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

`__iter__`

`RccKwargs.__iter__()`

Implement `iter(self)`.

`__le__`

`RccKwargs.__le__(value, /)`

Return `self<=value`.

`__len__`

`RccKwargs.__len__()`

Return `len(self)`.

`__lt__`

`RccKwargs.__lt__(value, /)`

Return `self<value`.

`__ne__`

`RccKwargs.__ne__(value, /)`

Return `self!=value`.

`__new__`

static `RccKwargs.__new__(cls, /, *args, **kwargs)`

`__reduce__`

`RccKwargs.__reduce__()`

Helper for pickle.

`__reduce_ex__`

`RccKwargs.__reduce_ex__(protocol, /)`
Helper for pickle.

`__repr__`

`RccKwargs.__repr__()`
Return `repr(self)`.

`__reversed__`

`RccKwargs.__reversed__()`
Return a reverse iterator over the dict keys.

`__setattr__`

`RccKwargs.__setattr__(name, value, /)`
Implement `setattr(self, name, value)`.

`__setitem__`

`RccKwargs.__setitem__(key, value, /)`
Set `self[key]` to value.

`__sizeof__`

`RccKwargs.__sizeof__()` → size of D in memory, in bytes

`__str__`

`RccKwargs.__str__()`
Return `str(self)`.

`__subclasshook__`

`RccKwargs.__subclasshook__()`
Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

clear

`RccKwargs.clear()` → None. Remove all items from D.

copy

`RccKwargs.copy()` → a shallow copy of D

fromkeys

`RccKwargs.fromkeys(value=None, /)`

Create a new dictionary with keys from iterable and values set to value.

get

`RccKwargs.get(key, default=None, /)`

Return the value for key if key is in the dictionary, else default.

items

`RccKwargs.items()` → a set-like object providing a view on D's items

keys

`RccKwargs.keys()` → a set-like object providing a view on D's keys

pop

`RccKwargs.pop(k[, d])` → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

popitem

`RccKwargs.popitem()`

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

setdefault

`RccKwargs.setdefault(key, default=None, /)`

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update

`RccKwargs.update([E], **F) → None`. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for k in E: D[k] = E[k] If E is present and lacks a `.keys()` method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values

`RccKwargs.values()` → an object providing a view on D's values

Methods Documentation

`__contains__(key, /)`

True if the dictionary has the specified key, else False.

`__delattr__(name, /)`

Implement `delattr(self, name)`.

`__delitem__(key, /)`

Delete `self[key]`.

`__dir__()`

Default `dir()` implementation.

`__eq__(value, /)`

Return `self==value`.

`__format__(format_spec, /)`

Default object formatter.

`__ge__(value, /)`

Return `self>=value`.

`__getattr__(name, /)`

Return `getattr(self, name)`.

`__getitem__()`

`x.__getitem__(y) <==> x[y]`

`__gt__(value, /)`

Return `self>value`.

`__init__(*args, **kwargs)`

__init_subclass__()

This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__iter__()

Implement iter(self).

__le__(value, /)

Return self<=value.

__len__()

Return len(self).

__lt__(value, /)

Return self<value.

__ne__(value, /)

Return self!=value.

static __new__(cls, /, *args, **kwargs)

__reduce__()

Helper for pickle.

__reduce_ex__(protocol, /)

Helper for pickle.

__repr__()

Return repr(self).

__reversed__()

Return a reverse iterator over the dict keys.

__setattr__(name, value, /)

Implement setattr(self, name, value).

__setitem__(key, value, /)

Set self[key] to value.

__sizeof__() → size of D in memory, in bytes

__str__()

Return str(self).

__subclasshook__()

Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

clear() → None. Remove all items from D.

copy() → a shallow copy of D

fromkeys(value=None, /)

Create a new dictionary with keys from iterable and values set to value.

get(*key*, *default=None*, /)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

pop(*k*, [*d*]) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise `KeyError` is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

setdefault(*key*, *default=None*, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update([*E*], ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for *k* in E: *D*[*k*] = *E*[*k*] If E is present and lacks a `.keys()` method, then does: for *k*, *v* in E: *D*[*k*] = *v* In either case, this is followed by: for *k* in F: *D*[*k*] = *F*[*k*]

values() → an object providing a view on D's values

UicKwargs

class `UicKwargs(*args, **kwargs)`

Keyword arguments to be used with `compile_ui_file`.

Attributes Summary

<code>__annotations__</code>	
<code>__dict__</code>	
<code>__doc__</code>	
<code>__hash__</code>	
<code>__module__</code>	
<code>__total__</code>	
<code>__weakref__</code>	list of weak references to the object (if defined)

Methods Summary

<code>__contains__</code>	True if the dictionary has the specified key, else False.
<code>__delattr__</code>	Implement <code>delattr(self, name)</code> .
<code>__delitem__</code>	Delete <code>self[key]</code> .
<code>__dir__</code>	Default <code>dir()</code> implementation.
<code>__eq__</code>	Return <code>self==value</code> .
<code>__format__</code>	Default object formatter.
<code>__ge__</code>	Return <code>self>=value</code> .
<code>__getattribute__</code>	Return <code>getattr(self, name)</code> .
<code>__getitem__</code>	<code>x.__getitem__(y) <==> x[y]</code>
<code>__gt__</code>	Return <code>self>value</code> .
<code>__init__</code>	
<code>__init_subclass__</code>	This method is called when a class is subclassed.
<code>__iter__</code>	Implement <code>iter(self)</code> .
<code>__le__</code>	Return <code>self<=value</code> .
<code>__len__</code>	Return <code>len(self)</code> .
<code>__lt__</code>	Return <code>self<value</code> .
<code>__ne__</code>	Return <code>self!=value</code> .
<code>__new__</code>	
<code>__reduce__</code>	Helper for pickle.
<code>__reduce_ex__</code>	Helper for pickle.
<code>__repr__</code>	Return <code>repr(self)</code> .
<code>__reversed__</code>	Return a reverse iterator over the dict keys.
<code>__setattr__</code>	Implement <code>setattr(self, name, value)</code> .
<code>__setitem__</code>	Set <code>self[key]</code> to value.
<code>__sizeof__</code>	
<code>__str__</code>	Return <code>str(self)</code> .
<code>__subclasshook__</code>	Abstract classes can override this to customize <code>issubclass()</code> .
<code>clear</code>	
<code>copy</code>	
<code>fromkeys</code>	Create a new dictionary with keys from iterable and values set to value.
<code>get</code>	Return the value for key if key is in the dictionary, else default.
<code>items</code>	
<code>keys</code>	
<code>pop</code>	If key is not found, d is returned if given, otherwise <code>KeyError</code> is raised
<code>popitem</code>	Remove and return a (key, value) pair as a 2-tuple.

continues on next page

Table 2 – continued from previous page

<i>setdefault</i>	Insert key with a value of default if key is not in the dictionary.
<i>update</i>	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
<i>values</i>	

__contains__

UicKwargs.__contains__(key, /)

True if the dictionary has the specified key, else False.

__delattr__

UicKwargs.__delattr__(name, /)

Implement delattr(self, name).

__delitem__

UicKwargs.__delitem__(key, /)

Delete self[key].

__dir__

UicKwargs.__dir__()

Default dir() implementation.

__eq__

UicKwargs.__eq__(value, /)

Return self==value.

__format__

UicKwargs.__format__(format_spec, /)

Default object formatter.

`__ge__`

UicKwargs.`__ge__`(*value*, /)
Return self>=value.

`__getattr__`

UicKwargs.`__getattr__`(*name*, /)
Return getattr(self, name).

`__getitem__`

UicKwargs.`__getitem__`()
x.`__getitem__`(y) <==> x[y]

`__gt__`

UicKwargs.`__gt__`(*value*, /)
Return self>value.

`__init__`

UicKwargs.`__init__`(*args, **kwargs)

`__init_subclass__`

UicKwargs.`__init_subclass__`()
This method is called when a class is subclassed.
The default implementation does nothing. It may be overridden to extend subclasses.

`__iter__`

UicKwargs.`__iter__`()
Implement iter(self).

`__le__`

UicKwargs.`__le__`(*value*, /)
Return self<=value.

__len__

UicKwargs.__len__()
 Return len(self).

__lt__

UicKwargs.__lt__(value, /)
 Return self<value.

__ne__

UicKwargs.__ne__(value, /)
 Return self!=value.

__new__

static UicKwargs.__new__(cls, /, *args, **kwargs)

__reduce__

UicKwargs.__reduce__()
 Helper for pickle.

__reduce_ex__

UicKwargs.__reduce_ex__(protocol, /)
 Helper for pickle.

__repr__

UicKwargs.__repr__()
 Return repr(self).

__reversed__

UicKwargs.__reversed__()
 Return a reverse iterator over the dict keys.

`__setattr__`

`UicKwargs.__setattr__(name, value, /)`
Implement `setattr(self, name, value)`.

`__setitem__`

`UicKwargs.__setitem__(key, value, /)`
Set `self[key]` to `value`.

`__sizeof__`

`UicKwargs.__sizeof__()` → size of `D` in memory, in bytes

`__str__`

`UicKwargs.__str__()`
Return `str(self)`.

`__subclasshook__`

`UicKwargs.__subclasshook__()`
Abstract classes can override this to customize `issubclass()`.

This is invoked early on by `abc.ABCMeta.__subclasscheck__()`. It should return `True`, `False` or `NotImplemented`. If it returns `NotImplemented`, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

`clear`

`UicKwargs.clear()` → `None`. Remove all items from `D`.

`copy`

`UicKwargs.copy()` → a shallow copy of `D`

`fromkeys`

`UicKwargs.fromkeys(value=None, /)`
Create a new dictionary with keys from iterable and values set to `value`.

get

UicKwargs.**get**(*key*, *default=None*, /)

Return the value for key if key is in the dictionary, else default.

items

UicKwargs.**items**() → a set-like object providing a view on D's items

keys

UicKwargs.**keys**() → a set-like object providing a view on D's keys

pop

UicKwargs.**pop**(*k*[, *d*]) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise `KeyError` is raised

popitem

UicKwargs.**popitem**()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises `KeyError` if the dict is empty.

setdefault

UicKwargs.**setdefault**(*key*, *default=None*, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update

UicKwargs.**update**([*E*], ***F*) → `None`. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for *k* in E: *D*[*k*] = *E*[*k*] If E is present and lacks a `.keys()` method, then does: for *k*, *v* in E: *D*[*k*] = *v* In either case, this is followed by: for *k* in F: *D*[*k*] = *F*[*k*]

values

UicKwargs.**values**() → an object providing a view on D's values

Methods Documentation

__contains__(*key*, /)

True if the dictionary has the specified key, else False.

__delattr__(*name*, /)

Implement delattr(self, name).

__delitem__(*key*, /)

Delete self[key].

__dir__()

Default dir() implementation.

__eq__(*value*, /)

Return self==value.

__format__(*format_spec*, /)

Default object formatter.

__ge__(*value*, /)

Return self>=value.

__getattr__(*name*, /)

Return getattr(self, name).

__getitem__()

x.__getitem__(y) <==> x[y]

__gt__(*value*, /)

Return self>value.

__init__(*args, **kwargs)

__init_subclass__()

This method is called when a class is subclassed.

The default implementation does nothing. It may be overridden to extend subclasses.

__iter__()

Implement iter(self).

__le__(*value*, /)

Return self<=value.

__len__()

Return len(self).

__lt__(*value*, /)

Return self<value.

__ne__(*value*, /)

Return self!=value.

static __new__(cls, /, *args, **kwargs)

__reduce__()

Helper for pickle.

__reduce_ex__(protocol, /)

Helper for pickle.

__repr__()

Return repr(self).

__reversed__()

Return a reverse iterator over the dict keys.

__setattr__(name, value, /)

Implement setattr(self, name, value).

__setitem__(key, value, /)

Set self[key] to value.

__sizeof__() → size of D in memory, in bytes

__str__()

Return str(self).

__subclasshook__()

Abstract classes can override this to customize issubclass().

This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

clear() → None. Remove all items from D.

copy() → a shallow copy of D

fromkeys(value=None, /)

Create a new dictionary with keys from iterable and values set to value.

get(key, default=None, /)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

pop(k[, d]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised

popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

setdefault(key, default=None, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

update($[E]$, $**F$) \rightarrow None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values() \rightarrow an object providing a view on D's values

Exceptions

Exception Summary

<i>ConfigNotFoundError</i>	Error thrown when the config file could not be found.
<i>QtDevHelperConfigError</i>	Error thrown when accessing functionality with insufficient config.

ConfigNotFoundError

exception ConfigNotFoundError

Error thrown when the config file could not be found.

QtDevHelperConfigError

exception QtDevHelperConfigError

Error thrown when accessing functionality with insufficient config.

6.1.4 qt_tools

Qt implementation cross compatibility module.

Functions

Summary

<i>call_qt_tool</i>	Call qt tools in a generic way.
<i>extend_qt_tool_path</i>	Prepend path variable with package dirs to qt-tools if present.
<i>find_qt_tool</i>	Find path to Qt tool executable like rcc, uic or designer.

call_qt_tool

call_qt_tool(*tool_name*: str, *, *arguments*: Sequence[str] = (), *no_wait*: bool = False) → None

Call qt tools in a generic way.

Parameters

- **tool_name** (str) – Name of the Qt tool to use (e.g. rcc, uic or designer)
- **arguments** (Sequence[str]) – Additional arguments for options for the tool. Defaults to ()
- **no_wait** (bool) – Whether or not to wait for the process to finish (used for CLI not to wait for designer application to close). Defaults to False

Raises

- **ValueError** – If arguments is not of type Sequence[str]
- **QtToolExecutionError** – If the tool returns a non-zero exit code.

extend_qt_tool_path

extend_qt_tool_path() → str

Prepend path variable with package dirs to qt-tools if present.

Returns

Path extended with library executable paths.

Return type

str

find_qt_tool

find_qt_tool(*tool_name*: str) → str

Find path to Qt tool executable like rcc, uic or designer.

Parameters

tool_name (str) – Name of the Qt tool to look up.

Returns

Path to the tool executable.

Return type

str

Raises

QtToolNotFoundError – If the tool could not be found in the path.

Exceptions

Exception Summary

<code>QtToolExecutionError</code>	Error thrown when a Qt tool returns a non-zero exit status code.
<code>QtToolNotFoundError</code>	Error thrown when a Qt tool can't be found.

QtToolExecutionError

exception `QtToolExecutionError`(*returncode: int, cmd: str, stdout: bytes, stderr: bytes*)

Error thrown when a Qt tool returns a non-zero exit status code.

See also:

`call_qt_tool`

QtToolNotFoundError

exception `QtToolNotFoundError`(*tool_name: str*)

Error thrown when a Qt tool can't be found.

See also:

`find_qt_tool`

6.1.5 transpiler

Module containing functionality to transpile resources *.scss, *.ui and *.qrc.

Functions

Summary

<code>build_all_assets</code>	Build all assets based on the provided configuration.
<code>build_resources</code>	Compile qrc files by iterating over all qrc files in a folder.
<code>build_uis</code>	Compile ui files by iterating over all ui files in a folder.
<code>compile_resource_file</code>	Call 'Qt Resource Compiler' to create code from a resource file.
<code>compile_ui_file</code>	Call 'Qt User Interface Compiler' to create code from a ui file.
<code>transpile_sass</code>	Transpile scss file to qss.

build_all_assets

build_all_assets(*config*: *Config* | *str* | *Path*, *log_function*: *Callable*[(*Ellipsis*, *None*)] = <function *print*>, *recurse_folder*: *bool* = *True*) → list[*Path*]

Build all assets based on the provided configuration.

This can be used in the built script when using source distributions.

Parameters

- **config** (*Config*) – Configuration to use for building assets. If a path is passed it will try to find the config.
- **log_function** (*Callable*[..., *None*]) – Function used to print log messages. Defaults to rich.print
- **recurse_folder** (*bool*) – Whether or not to recurse directories searching for files. Defaults to True

Returns

List of generated files.

Return type

list[*Path*]

See also:

[*load_config*](#)

build_resources

build_resources(*resource_folder*: *Path*, *generated_rc_code_folder*: *Path*, *, *flatten_path*: *bool* = *True*, *rcc_kwargs*: *RccKwargs* | *None* = *None*, *log_function*: *Callable*[(*Ellipsis*, *None*)] = <function *print*>, *recurse_folder*: *bool* = *True*) → list[*Path*]

Compile qrc files by iterating over all qrc files in a folder.

Parameters

- **resource_folder** (*Path*) – Base path containing the input qrc files.
- **generated_rc_code_folder** (*Path*) – Base path to save generated code from qrc files to.
- **flatten_path** (*bool*) – Whether or not to flatten the folder structure of the ui files (For the ‘python’ generator this should be True in order for imports to resolve). Defaults to True
- **rcc_kwargs** (*RccKwargs* | *None*) – Keyword arguments passed to the rcc executable. Defaults to None
- **log_function** (*Callable*[..., *None*]) – Function used to print log messages. Defaults to rich.print
- **recurse_folder** (*bool*) – Whether or not to recurse directories searching for files. Defaults to True

Returns

List of generated files.

Return type

list[*Path*]

build_uis

build_uis(*ui_files_folder*: Path, *generated_ui_code_folder*: Path, *, *flatten_path*: bool = True, *uic_kwargs*: UicKwargs | None = None, *log_function*: Callable[(Ellipsis, None)] = <function print>, *recurse_folder*: bool = True) → list[Path]

Compile ui files by iterating over all ui files in a folder.

Parameters

- **ui_files_folder** (Path) – Base path containing the input ui files.
- **generated_ui_code_folder** (Path) – Base path to save generated code from ui files to.
- **flatten_path** (bool) – Whether or not to flatten the folder structure of the ui files (For the ‘python’ generator this should be True in order for imports to resolve). Defaults to True
- **uic_kwargs** (UicKwargs / None) – Keyword arguments passed to the uic executable. Defaults to None
- **log_function** (Callable[... , None]) – Function used to print log messages. Defaults to rich.print
- **recurse_folder** (bool) – Whether or not to recurse directories searching for files. Defaults to True

Returns

List of generated files.

Return type

list[Path]

See also:

[`compile_ui_file`](#)

compile_resource_file

compile_resource_file(*qrc_file*: str | Path, *output_path*: str | Path, *, *generator*: Literal['python', 'cpp'] = 'python', *rcc_args*: Sequence[str] = ()) → Path

Call ‘Qt Resource Compiler’ to create code from a resource file.

Parameters

- **qrc_file** (str / Path) – Path to the resource file.
- **output_path** (str / Path) – Path the output file should be saved to.
- **generator** (Literal["python", "cpp"]) – Language to generate code for. Defaults to “python”
- **rcc_args** (Sequence[str]) – Additional args for ‘rcc’ (use ‘–help’ for details). Defaults to ()

Returns

Path of the compiled file

Return type

Path

compile_ui_file

compile_ui_file(*ui_file*: *str* | *Path*, *output_path*: *str* | *Path*, *, *generator*: *Literal*['python', 'cpp'] = 'python', *form_import*: *bool* = *True*, *uic_args*: *Sequence*[*str*] = ()) → *Path*

Call ‘Qt User Interface Compiler’ to create code from a ui file.

Parameters

- **ui_file** (*str* | *Path*) – Path to the ui file.
- **output_path** (*str* | *Path*) – Path the output file should be saved to.
- **generator** (*Literal*["python", "cpp"]) – Language to generate code for. Defaults to “python”
- **form_import** (*bool*) – Sets the ‘–from-imports’ flag when used with python. Defaults to *True*
- **uic_args** (*Sequence*[*str*]) – Additional args for ‘uic’ (use ‘–help’ for details). Defaults to ()

Returns

Path of the compiled file

Return type

Path

transpile_sass

transpile_sass(*sass_file*: *str* | *Path*, *qss_file*: *str* | *Path*) → *Path*

Transpile scss file to qss.

This function differs from `qtsass.compile_filename` in that it ensures that the output file is utf8 encoded.

Parameters

- **sass_file** (*str* | *Path*) – Path to the sass input file.
- **qss_file** (*str* | *Path*) – Path to output the compiled qss file to.

Returns

Absolute path to the compiled qss file.

Return type

Path

6.1.6 utils

Module containing utility functionality.

Functions

Summary

<code>find_matching_files</code>	Search for files matching <code>file_pattern</code> .
<code>format_rel_output_path</code>	Format <code>file_path</code> to a relative path for output files.

`find_matching_files`

find_matching_files(*files: Sequence[Path], file_pattern: str, *, recurse_folder: bool = True*) → tuple[str, ...]

Search for files matching `file_pattern`.

Parameters

- **files** (*Sequence[Path]*) – List of paths (files or folders) to check for matching files.
- **file_pattern** (*str*) – Pattern to match files, this a Unix shell-style pattern and nto an regex.
- **recurse_folder** (*bool*) – Whether or not to recurse directories searching for files. Defaults to True

Returns

Tuple of posix conform string paths to files matching `file_pattern`.

Return type

tuple[str,...]

`format_rel_output_path`

format_rel_output_path(*root_folder: Path, file_path: Path, format_string: str, *, flatten_path: bool = True*) → Path

Format `file_path` to a relative path for output files.

Parameters

- **root_folder** (*Path*) – Root folder of the file resides in.
- **file_path** (*Path*) – Path to the file to be formatted.
- **format_string** (*str*) – String with format instruction ‘file_stem’ (e.g. ‘Ui_{file_stem}.py’).
- **flatten_path** (*bool*) – Whether or not to persist the original folder structure. Defaults to True

Returns

Relative path in respect to `root_folder` for the formatted file.

Return type

Path

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at <https://github.com/s-weigand/qt-dev-helper/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

7.1.4 Write Documentation

Qt Dev Helper could always use more documentation, whether as part of the official Qt Dev Helper docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/s-weigand/qt-dev-helper/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up `qt_dev_helper` for local development.

1. Fork the `qt-dev-helper` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/qt_dev_helper.git
```

3. Install your local copy into a virtualenv. Assuming you have `virtualenvwrapper` installed, this is how you set up your fork for local development:

```
$ mkvirtualenv qt_dev_helper
$ cd qt-dev-helper/
$ pip install -e .
```

4. install the `pre-commit` and `pre-push` hooks:

```
$ pre-commit install && pre-commit install -t pre-push
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

6. When you're done making changes, check that your changes pass `flake8` and the tests, including testing other Python versions with `tox`:

```
$ tox
```

To get `flake8` and `tox`, just `pip` install them into your virtualenv.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
3. The pull request should work for Python 3.8, 3.9 and 3.10. Check <https://github.com/s-weigand/qt-dev-helper/actions> and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

```
$ pytest tests.test_qt_dev_helper
```

7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch  
$ git push --follow-tags
```

Travis will then deploy to PyPI if tests pass.

CHANGELOG

8.1 0.0.4 (2023-07-08)

- Fix missing CLI docs on RTD by @s-weigand in (#63)
- Add py3.11 CI tests and autoupdate pre-commit config by @s-weigand in (#69)
- Improve readme by @s-weigand in (#70)
- Pin pydantic to version <2 by @s-weigand in (#115)

8.2 0.0.3 (2022-09-05)

- Removed qtsass310 dependency in favor of qtsass>=0.3.1 (#54)

8.3 0.0.2 (2022-09-05)

- Use extended path when calling Qt tools and add folder containing lib files (#52)

8.4 0.0.1 (2022-09-04)

- First release on PyPI.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

q

- `qt_dev_helper`, 19
- `qt_dev_helper.__main__`, 19
- `qt_dev_helper.cli`, 19
 - `_cli_docs`, 20
 - `commands`, 20
 - `build`, 20
 - `designer`, 21
 - `main_app`, 21
 - `utils`, 21
- `qt_dev_helper.config`, 22
- `qt_dev_helper.qt_tools`, 44
- `qt_dev_helper.transpiler`, 46
- `qt_dev_helper.utils`, 49

Symbols

`__contains__()` (*RccKwargs method*), 33
`__contains__()` (*UicKwargs method*), 42
`__delattr__()` (*RccKwargs method*), 33
`__delattr__()` (*UicKwargs method*), 42
`__delitem__()` (*RccKwargs method*), 33
`__delitem__()` (*UicKwargs method*), 42
`__dir__()` (*RccKwargs method*), 33
`__dir__()` (*UicKwargs method*), 42
`__eq__()` (*RccKwargs method*), 33
`__eq__()` (*UicKwargs method*), 42
`__format__()` (*RccKwargs method*), 33
`__format__()` (*UicKwargs method*), 42
`__ge__()` (*RccKwargs method*), 33
`__ge__()` (*UicKwargs method*), 42
`__getattribute__()` (*RccKwargs method*), 33
`__getattribute__()` (*UicKwargs method*), 42
`__getitem__()` (*RccKwargs method*), 33
`__getitem__()` (*UicKwargs method*), 42
`__gt__()` (*RccKwargs method*), 33
`__gt__()` (*UicKwargs method*), 42
`__init__()` (*RccKwargs method*), 33
`__init__()` (*UicKwargs method*), 42
`__init_subclass__()` (*RccKwargs method*), 33
`__init_subclass__()` (*UicKwargs method*), 42
`__iter__()` (*RccKwargs method*), 34
`__iter__()` (*UicKwargs method*), 42
`__le__()` (*RccKwargs method*), 34
`__le__()` (*UicKwargs method*), 42
`__len__()` (*RccKwargs method*), 34
`__len__()` (*UicKwargs method*), 42
`__lt__()` (*RccKwargs method*), 34
`__lt__()` (*UicKwargs method*), 42
`__ne__()` (*RccKwargs method*), 34
`__ne__()` (*UicKwargs method*), 42
`__new__()` (*RccKwargs static method*), 34
`__new__()` (*UicKwargs static method*), 42
`__reduce__()` (*RccKwargs method*), 34
`__reduce__()` (*UicKwargs method*), 43
`__reduce_ex__()` (*RccKwargs method*), 34
`__reduce_ex__()` (*UicKwargs method*), 43
`__repr__()` (*RccKwargs method*), 34
`__repr__()` (*UicKwargs method*), 43
`__reversed__()` (*RccKwargs method*), 34
`__reversed__()` (*UicKwargs method*), 43
`__setattr__()` (*RccKwargs method*), 34
`__setattr__()` (*UicKwargs method*), 43
`__setitem__()` (*RccKwargs method*), 34
`__setitem__()` (*UicKwargs method*), 43
`__sizeof__()` (*RccKwargs method*), 34
`__sizeof__()` (*UicKwargs method*), 43
`__str__()` (*RccKwargs method*), 34
`__str__()` (*UicKwargs method*), 43
`__subclasshook__()` (*RccKwargs method*), 34
`__subclasshook__()` (*UicKwargs method*), 43
`__check_input_exists()` (in module *qt_dev_helper.config*), 23
`__check_symmetric_io_definition()` (in module *qt_dev_helper.config*), 23
`__str_list_factory()` (in module *qt_dev_helper.config*), 24
`--config`
 qt-dev-helper-build command line option,
 7
`--flatten-folder-structure`
 qt-dev-helper-build command line option,
 8
`--form-import`
 qt-dev-helper-build command line option,
 8
`--generated-rc-code-folder`
 qt-dev-helper-build command line option,
 8
`--generated-ui-code-folder`
 qt-dev-helper-build command line option,
 8
`--generator`
 qt-dev-helper-build command line option,
 7
`--install-completion`
 qt-dev-helper command line option, 7
`--no-flatten-folder-structure`
 qt-dev-helper-build command line option,
 8

`--no-form-import`
 `qt-dev-helper-build` command line option, 8

`--no-open-files`
 `qt-dev-helper-designer` command line option, 9

`--no-qss`
 `qt-dev-helper-build` command line option, 8

`--no-rc`
 `qt-dev-helper-build` command line option, 8

`--no-recurse-folder`
 `qt-dev-helper-designer` command line option, 9

`--no-ui`
 `qt-dev-helper-build` command line option, 8

`--open-files`
 `qt-dev-helper-designer` command line option, 9

`--qss`
 `qt-dev-helper-build` command line option, 8

`--rc`
 `qt-dev-helper-build` command line option, 8

`--rcc-args`
 `qt-dev-helper-build` command line option, 8

`--recurse-folder`
 `qt-dev-helper-build` command line option, 7
 `qt-dev-helper-designer` command line option, 9

`--resource-folder`
 `qt-dev-helper-build` command line option, 8

`--root-qss-file`
 `qt-dev-helper-build` command line option, 8

`--root-sass-file`
 `qt-dev-helper-build` command line option, 8

`--show-completion`
 `qt-dev-helper` command line option, 7

`--ui`
 `qt-dev-helper-build` command line option, 8

`--ui-files-folder`
 `qt-dev-helper-build` command line option, 8

`--uic-args`
 `qt-dev-helper-build` command line option, 8

`-c`
 `qt-dev-helper-build` command line option, 7

`-g`
 `qt-dev-helper-build` command line option, 7

`-r`
 `qt-dev-helper-build` command line option, 7

B

`BASE_PATH`
 `qt-dev-helper-build` command line option, 9

`base_path` (*Config attribute*), 14

`build()` (*in module `qt_dev_helper.cli.commands.build`*), 20

`build_all_assets()` (*in module `qt_dev_helper.transpiler`*), 47

`build_resources()` (*in module `qt_dev_helper.transpiler`*), 47

`build_uis()` (*in module `qt_dev_helper.transpiler`*), 48

C

`call_qt_tool()` (*in module `qt_dev_helper.qt_tools`*), 45

`clear()` (*RccKwargs method*), 34

`clear()` (*UicKwargs method*), 43

`CodeGenerators` (*class in `qt_dev_helper.config`*), 26

`compile_resource_file()` (*in module `qt_dev_helper.transpiler`*), 48

`compile_ui_file()` (*in module `qt_dev_helper.transpiler`*), 49

`ConfigNotFoundError`, 44

`copy()` (*RccKwargs method*), 34

`copy()` (*UicKwargs method*), 43

D

`deactivate_resource_build()` (*Config method*), 16

`deactivate_style_build()` (*Config method*), 16

`deactivate_ui_build()` (*Config method*), 16

`designer()` (*in module `qt_dev_helper.cli.commands.designer`*), 21

E

`expand_io_paths()` (*in module `qt_dev_helper.config`*), 24

`extend_qt_tool_path()` (*in module `qt_dev_helper.qt_tools`*), 45

F

`FILES`
 `qt-dev-helper-designer` command line option, 9

find_config() (in module *qt_dev_helper.config*), 24
 find_matching_files() (in module *qt_dev_helper.utils*), 50
 find_qt_tool() (in module *qt_dev_helper.qt_tools*), 45
 flatten_folder_structure (Config attribute), 14
 form_import (Config attribute), 14
 format_rel_output_path() (in module *qt_dev_helper.utils*), 50
 fromkeys() (RccKwargs method), 34
 fromkeys() (UicKwargs method), 43

G

generated_rc_code_folder (Config attribute), 14
 generated_ui_code_folder (Config attribute), 14
 generator (Config attribute), 14
 get() (RccKwargs method), 34
 get() (UicKwargs method), 43

I

items() (RccKwargs method), 35
 items() (UicKwargs method), 43

K

keys() (RccKwargs method), 35
 keys() (UicKwargs method), 43

L

load_config() (in module *qt_dev_helper.config*), 25
 load_toml_config() (in module *qt_dev_helper.config*), 25

M

module
 qt_dev_helper, 19
 qt_dev_helper.__main__, 19
 qt_dev_helper.cli, 19
 qt_dev_helper.cli._cli_docs, 20
 qt_dev_helper.cli.commands, 20
 qt_dev_helper.cli.commands.build, 20
 qt_dev_helper.cli.commands.designer, 21
 qt_dev_helper.cli.main_app, 21
 qt_dev_helper.cli.utils, 21
 qt_dev_helper.config, 22
 qt_dev_helper.qt_tools, 44
 qt_dev_helper.transpiler, 46
 qt_dev_helper.utils, 49

P

parse_optional_args_string() (in module *qt_dev_helper.cli.utils*), 22
 pop() (RccKwargs method), 35
 pop() (UicKwargs method), 43
 popitem() (RccKwargs method), 35

popitem() (UicKwargs method), 43

Q

qt_dev_helper
 module, 19
qt_dev_helper.__main__
 module, 19
qt_dev_helper.cli
 module, 19
qt_dev_helper.cli._cli_docs
 module, 20
qt_dev_helper.cli.commands
 module, 20
qt_dev_helper.cli.commands.build
 module, 20
qt_dev_helper.cli.commands.designer
 module, 21
qt_dev_helper.cli.main_app
 module, 21
qt_dev_helper.cli.utils
 module, 21
qt_dev_helper.config
 module, 22
qt_dev_helper.qt_tools
 module, 44
qt_dev_helper.transpiler
 module, 46
qt_dev_helper.utils
 module, 49
 qt-dev-helper command line option
 --install-completion, 7
 --show-completion, 7
 qt-dev-helper-build command line option
 --config, 7
 --flatten-folder-structure, 8
 --form-import, 8
 --generated-rc-code-folder, 8
 --generated-ui-code-folder, 8
 --generator, 7
 --no-flatten-folder-structure, 8
 --no-form-import, 8
 --no-qss, 8
 --no-rc, 8
 --no-ui, 8
 --qss, 8
 --rc, 8
 --rcc-args, 8
 --recurse-folder, 7
 --resource-folder, 8
 --root-qss-file, 8
 --root-sass-file, 8
 --ui, 8
 --ui-files-folder, 8
 --uic-args, 8

- c, 7
- g, 7
- r, 7
- BASE_PATH, 9
- qt-dev-helper-designer command line option
 - no-open-files, 9
 - no-recurse-folder, 9
 - open-files, 9
 - recurse-folder, 9
- FILES, 9
- QtDevHelperConfigError, 44
- QtToolExecutionError, 46
- QtToolNotFoundError, 46

R

- rc_folder_paths() (*Config method*), 16
- rcc_args (*Config attribute*), 15
- rcc_kwargs() (*Config method*), 16
- RccKwargs (*class in qt_dev_helper.config*), 26
- resource_folder (*Config attribute*), 15
- root_qss_file (*Config attribute*), 15
- root_sass_file (*Config attribute*), 15
- root_style_paths() (*Config method*), 16

S

- setdefault() (*RccKwargs method*), 35
- setdefault() (*UicKwargs method*), 43

T

- transpile_sass() (in *module qt_dev_helper.transpiler*), 49

U

- ui_files_folder (*Config attribute*), 15
- ui_folder_paths() (*Config method*), 16
- uic_args (*Config attribute*), 15
- uic_kwargs() (*Config method*), 16
- UicKwargs (*class in qt_dev_helper.config*), 35
- update() (*Config method*), 16
- update() (*RccKwargs method*), 35
- update() (*UicKwargs method*), 43

V

- values() (*RccKwargs method*), 35
- values() (*UicKwargs method*), 44