Jinja Api Documentation Documentation

Release 0.5.0

David Zuber

Apr 14, 2019

Contents

Sphinx API Doc with Jinja2 templates

CHAPTER 1

Documentation

The full documentation is at http://jinjaapidoc.rtfd.org.

CHAPTER 2

Features

- jinjaapidoc is a tool for automatic generation of Sphinx sources that, using the jinja2 template rendering engine, document a whole package in the style of other automatic API documentation tools. It is based of the builtin sphinx apidoc

CHAPTER 3

Demo

This documentation itself makes use of the jinjaapidoc package. Check *Reference* to see the outcome of the default templates. All pages in the reference are automatically generated.

CHAPTER 4

Contents:

4.1 Installation

At the command line either via easy_install or pip:

```
$ easy_install jinjaapidoc
$ pip install jinjaapidoc
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv jinjaapidoc
$ pip install jinjaapidoc
```

4.2 Usage

4.2.1 Quickstart

Apply these changes to your `conf.py`:

1. Add `jinjaapidoc` to the extensions. Preferably before autosummary
2. Set `jinjaapidoc_srcdir` to the location of your python source code.
3. Optional - Set `jinjaapidoc_outpudir` to the directory for all generated files.
4. Set `autosummary_generate` to `True`.
5. Include the top-level packages/modules into your documents.

4.2.2 Enable Extension

To use Jinja Api Documentation in a project, add it to the extensions in your sphinx `conf.py`:

```
extensions = [
  'jinjaapidoc',
]
```

Important: jinjaapidoc will generate files. Other extensions, such as autosummary might want to process them, so add these extensions afterwards. The autosummary extension will automatically be enabled by jinjaapidoc.

4.2.3 Configuration

There are a few config values you can set. The only necessary one is the srcdir:

jinjaapi_srcdir **REQUIRED!** the path to the source directory of your python code.

jinjaapi_outputdir directory for generated files. Defaults to the documenation source directory (ussually the directory of conf.py).

jinjaapi_nodelete `bool` - If False, delete the output directory first. Defaults to True.

jinjaapi_exclude_paths `list` - A list of paths to exclude.

jinjaapi_force `bool` - If True, overwrite existing files. Defaults to True.

jinjaapi_followlinks `bool` - If True, follow symbolic links. Defaults to True.

jinjaapi_dryrun `bool` - If True, do not create any files. Defaults to False.

jinjaapi_includeprivate `bool` - If True, include private modules. Defaults to True.

jinjaapi_addsummarytemplate `bool` - If True, add autosummary template for classes. Defaults to True.

jinjaapi_include_from_all `bool` - If True, include members of a module or package that are listed in `__all__`. Defaults to True.

4.2.4 Documenter

The jinjaapidoc package provides an additional `sphinx.ext.autodoc.ModuleDocumenter`: *jinjaapidoc.ext.ModDocstringDocumenter*. The documenter will only insert the docstring of the module but will not create any index. Use him like this (replace `<<package.module>>`):

```
.. automoddoconly:: <<package.module>>
```

4.2.5 Templates

You can use your own templates for rendering the rst files. Add the directory with the templates to `templates_path` in the `conf.py`. You can provide a *jinjaapidoc.gendoc.MODULE_TEMPLATE_NAME* and *jinjaapidoc.gendoc.PACKAGE_TEMPLATE_NAME* template.

The context for the templates is generated by *jinjaapidoc.gendoc.get_context()*. Variables you can use are:

- **package** The top package
- **module** the module
- **fullname** package.module

- **subpkgs** packages beneath module
- **submods** modules beneath module
- **classes** public classes in module
- **allclasses** public and private classes in module
- **exceptions** public exceptions in module
- **allexceptions** public and private exceptions in module
- **functions** public functions in module
- **allfunctions** public and private functions in module
- **data** public data in module
- **alldata** public and private data in module
- **members** dir(module)

The default template looks like this:

```
{% block header -%}
:mod:`{{ fullname }}`
=======
{% for c in fullname %}={% endfor %}
{%- endblock %}

{% block subpackages %}{% if subpkgs -%}
Subpackages
-----------
.. toctree::
   :maxdepth: 3
{% for p in subpkgs %}
   {{ fullname }}.{{ p }}
{%- endfor %}{% endif %}{% endblock %}

{% block submodules %}{% if submods -%}
Submodules
----------
.. toctree::
   :maxdepth: 1
{% for m in submods %}
   {{ fullname }}.{{ m }}
{%- endfor %}{% endif %}{% endblock %}

{% block contents %}{% if ispkg -%}
Module contents
---------------
{%- endif %}

.. automoddoconly:: {{ fullname }}

.. currentmodule:: {{ fullname }}

{% block classsummary %}{% if classes -%}
Classes
~~~~~~~

.. autosummary::
   :toctree: {{ fullname }}
```

(continues on next page)

(continued from previous page)

```
{% for c in classes %}
     {{ c }}
{%- endfor %}{% endif %}{% endblock %}

{% block exceptionssummary %}{% if exceptions -%}
Exceptions
~~~~~~~~~~

.. autosummary::
   :toctree: {{ fullname }}
{% for e in exceptions %}
     {{ e }}
{%- endfor %}{% endif %}{% endblock %}

{% block functionsssummary %}{% if functions -%}
Functions
~~~~~~~~~

.. autosummary::
{% for f in functions %}
     {{ f }}
{%- endfor %}{% endif %}{% endblock %}

{% block datasummary %}{% if data -%}
Data
~~~~

.. autosummary::
{% for d in data %}
     {{ d }}
{%- endfor %}{% endif %}{% endblock %}

{% block functionsdoc -%}
{% for f in functions %}
.. autofunction:: {{ f }}
{%- endfor %}{% endblock %}

{% block datadoc -%}
{% for d in data %}
.. autodata:: {{ d}}
{%- endfor %}{% endblock %}{% endblock %}
```

4.2.6 Autosummary

The default templates use autosummary. Thats why autosummary will be setup automatically. If you already added it to your extensions, make sure it is behind jinjaapidoc. That way, autosummary will also consider the new generated files. Set *autosummary_generate* to True in your *conf.py*

By default, custom autosummary templates are added. Right now, there is one for classes. You can set *jinjaapi_addsummarytemplate* in *conf.py* to False to avoid that and fall back to the default one. The template looks like this:

```
{{ fullname }}
{{ underline }}
```

(continues on next page)

(continued from previous page)

```
.. currentmodule:: {{ module }}

.. autoclass:: {{ objname }}
   :members:
   :undoc-members:
   :show-inheritance:

   {% block methods -%}
   .. automethod:: __init__

   {% if methods -%}
   .. rubric:: **Methods**

   .. autosummary::
   {% for item in methods %}
      ~{{ name }}.{{ item }}
   {%- endfor %}
   {%- endif %}
   {%- endblock %}

   {% block attributes -%}
   {%- if attributes -%}
   .. rubric:: **Attributes**

   .. autosummary::
   {% for item in attributes %}
      ~{{ name }}.{{ item }}
   {%- endfor %}
   {%- endif %}
   {%- endblock %}
```

4.3 Reference

Automatic generated Documenation by apidoc and autodoc.

4.3.1 jinjaapidoc

Submodules

jinjaapidoc.ext

This module contains content related to sphinx extensions.

Classes

ModDocstringDocumenter(*args)	A documenter for modules which only inserts the docstring of the module.

jinjaapidoc.ext.ModDocstringDocumenter

class jinjaapidoc.ext.**ModDocstringDocumenter**(**args*)

Bases: sphinx.ext.autodoc.ModuleDocumenter

A documenter for modules which only inserts the docstring of the module.

__init__(**args*)

x.__init__(...) initializes x; see help(type(x)) for signature

Methods

__init__(*args)	x.__init__(...) initializes x; see help(type(x)) for signature
add_content(more_content[, no_docstring])	Add content from docstrings, attribute documentation and user.
add_directive_header(sig)	Add the directive header and options to the generated content.
add_line(line, source, *lineno)	Append one line of generated reST to the output.
can_document_member(member, membername, ...)	Called to see if a member can be documented by this documenter.
check_module()	Check if *self.object* is really defined in the module given by *self.modname*.
document_members([all_members])	Generate reST for member documentation.
filter_members(members, want_all)	Filter the given member list.
format_args()	Format the argument signature of *self.object*.
format_name()	Format the name of *self.object*.
format_signature()	Format the signature (arguments and return annotation) of the object.
generate([more_content, real_modname, ...])	Generate reST for the object given by *self.name*, and possibly for its members.
get_attr(obj, name, *defargs)	getattr() override for types such as Zope interfaces.
get_doc([encoding, ignore])	Decode and return lines of the docstring(s) for the object.
get_object_members(want_all)	Return *(members_check_module, members)* where *members* is a list of *(membername, member)* pairs of the members of *self.object*.
get_real_modname()	Get the real module name of an object to document.
get_sourcename()	
import_object()	Import the object given by *self.modname* and *self.objpath* and set it as *self.object*.
parse_name()	Determine what module to import and what attribute to document.
process_doc(docstrings)	Let the user process the docstrings before adding them.
resolve_name(modname, parents, path, base)	Resolve the module and name of the object to document given by the arguments and the current module/class.

Attributes

content_indent	
documenters	Returns registered Documenter classes
member_order	
objtype	
option_spec	
priority	
titles_allowed	

objtype = 'moddoconly'

content_indent = ''

add_directive_header(*sig*)
Add the directive header and options to the generated content.

document_members(*all_members=False*)
Generate reST for member documentation.

If *all_members* is True, do all members, else those given by *self.options.members*.

jinjaapidoc.gendoc

This is a modification of sphinx.apidoc by David.Zuber. It uses jinja templates to render the rst files.

Parses a directory tree looking for Python modules and packages and creates ReST files appropriately to create code documentation with Sphinx.

This is derived form the "sphinx-apidoc" script, which is:

Copyright 2007-2014 by the Sphinx team, see http://sphinx-doc.org/latest/authors.html.

Functions

create_module_file(app, env, package, ...)	Build the text of the file and write the file.
create_package_file(app, env, root_package, ...)	Build the text of the file and write the file.
generate(app, src, dest[, exclude, ...])	Generage the rst files
get_context(app, package, module, fullname)	Return a dict for template rendering
get_members(app, mod, typ[, include_public])	Return the members of mod of the given type
get_submodules(app, module)	Get all submodules without packages for the given module/package
get_subpackages(app, module)	Get all subpackages for the given module/package
import_name(app, name)	Import the given name and return name, obj, parent, mod_name
is_excluded(root, excludes)	Check if the directory is in the exclude list.
main(app)	Parse the config of the app and initiate the generation process
make_environment(loader)	Return a new jinja2.Environment with the given loader
make_loader(template_dirs)	Return a new jinja2.FileSystemLoader that uses the template_dirs
makename(package, module)	Join package and module with a dot.

Continued on next page

Table 4 – continued from previous page

normalize_excludes(excludes)	Normalize the excluded directory list.
prepare_dir(app, directory[, delete])	Create apidoc dir, delete contents if delete is True.
recurse_tree(app, env, src, dest, excludes, ...)	Look for every file in the directory tree and create the corresponding ReST files.
shall_skip(app, module, private)	Check if we want to skip this module.
write_file(app, name, text, dest, suffix, ...)	Write the output file for module/package <name>.

Data

AUTOSUMMARYTEMPLATE_DIR	Templates for autosummary
INITPY	str(object='') -> string
MODULE_TEMPLATE_NAME	Name of the template that is used for rendering modules.
PACKAGE_TEMPLATE_NAME	Name of the template that is used for rendering packages.
PY_SUFFIXES	set() -> new empty set object set(iterable) -> new set object
TEMPLATE_DIR	Built-in template dir for jinjaapi rendering
logger	LoggerAdapter allowing `type` and `subtype` keywords.

jinjaapidoc.gendoc.**create_module_file**(*app*, *env*, *package*, *module*, *dest*, *suffix*, *dryrun*, *force*)

Build the text of the file and write the file.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **env** (jinja2.Environment) – the jinja environment for the templates
- **package** (str) – the package name
- **module** (str) – the module name
- **dest** (str) – the output directory
- **suffix** (str) – the file extension
- **dryrun** (bool) – If True, do not create any files, just log the potential location.
- **force** (bool) – Overwrite existing files

Returns None

Raises None

jinjaapidoc.gendoc.**create_package_file**(*app*, *env*, *root_package*, *sub_package*, *private*, *dest*, *suffix*, *dryrun*, *force*)

Build the text of the file and write the file.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **env** (jinja2.Environment) – the jinja environment for the templates
- **root_package** (str) – the parent package
- **sub_package** (str) – the package name without root

- **private** (bool) – Include "_private" modules
- **dest** (str) – the output directory
- **suffix** (str) – the file extension
- **dryrun** (bool) – If True, do not create any files, just log the potential location.
- **force** (bool) – Overwrite existing files

Returns None

Raises None

jinjaapidoc.gendoc.**generate**(*app*, *src*, *dest*, *exclude=[]*, *followlinks=False*, *force=False*, *dryrun=False*, *private=False*, *suffix='rst'*, *template_dirs=None*)

Generage the rst files

Raises an OSError if the source path is not a directory.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **src** (str) – path to python source files
- **dest** (str) – output directory
- **exclude** (list) – list of paths to exclude
- **followlinks** (bool) – follow symbolic links
- **force** (bool) – overwrite existing files
- **dryrun** (bool) – do not create any files
- **private** (bool) – include "_private" modules
- **suffix** (str) – file suffix
- **template_dirs** (None | list) – directories to search for user templates

Returns None

Return type None

Raises OSError

jinjaapidoc.gendoc.**get_context**(*app*, *package*, *module*, *fullname*)

Return a dict for template rendering

Variables:

- **package** The top package
- **module** the module
- **fullname** package.module
- **subpkgs** packages beneath module
- **submods** modules beneath module
- **classes** public classes in module
- **allclasses** public and private classes in module
- **exceptions** public exceptions in module
- **allexceptions** public and private exceptions in module

- **functions** public functions in module
- **allfunctions** public and private functions in module
- **data** public data in module
- **alldata** public and private data in module
- **members** dir(module)

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **package** (*str*) – the parent package name
- **module** (*str*) – the module name
- **fullname** (*str*) – package.module

Returns a dict with variables for template rendering

Return type dict

Raises None

jinjaapidoc.gendoc.**get_members**(*app*, *mod*, *typ*, *include_public=None*)
Return the members of mod of the given type

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **mod** (*module*) – the module with members
- **typ** (*str*) – the typ, 'class', 'function', 'exception', 'data', 'members'
- **include_public** (*list | None*) – list of private members to include to publics

Returns None

Return type None

Raises None

jinjaapidoc.gendoc.**get_submodules**(*app*, *module*)
Get all submodules without packages for the given module/package

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **module** (*module | str*) – the module to query or module path

Returns list of module names excluding packages

Return type list

Raises TypeError

jinjaapidoc.gendoc.**get_subpackages**(*app*, *module*)
Get all subpackages for the given module/package

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **module** (*module | str*) – the module to query or module path

Returns list of packages names

Return type list

Raises TypeError

jinjaapidoc.gendoc.**import_name**(*app*, *name*)
Import the given name and return name, obj, parent, mod_name

Parameters **name** (*str*) – name to import

Returns the imported object or None

Return type object | None

Raises None

jinjaapidoc.gendoc.**is_excluded**(*root*, *excludes*)
Check if the directory is in the exclude list.

Note: by having trailing slashes, we avoid common prefix issues, like e.g. an exlude "foo" also accidentally excluding "foobar".

jinjaapidoc.gendoc.**main**(*app*)
Parse the config of the app and initiate the generation process

Parameters **app** (sphinx.application.Sphinx) – the sphinx app

Returns None

Return type None

Raises None

jinjaapidoc.gendoc.**make_environment**(*loader*)
Return a new jinja2.Environment with the given loader

Parameters **loader** (jinja2.BaseLoader) – a jinja2 loader

Returns a new environment

Return type jinja2.Environment

Raises None

jinjaapidoc.gendoc.**make_loader**(*template_dirs*)
Return a new jinja2.FileSystemLoader that uses the template_dirs

Parameters **template_dirs** (None | list) – directories to search for templates

Returns a new loader

Return type jinja2.FileSystemLoader

Raises None

jinjaapidoc.gendoc.**makename**(*package*, *module*)
Join package and module with a dot.

Package or Module can be empty.

Parameters

- **package** (str) – the package name
- **module** (str) – the module name

Returns the joined name

Return type str

Raises AssertionError, if both package and module are empty

jinjaapidoc.gendoc.**normalize_excludes**(*excludes*)
Normalize the excluded directory list.

jinjaapidoc.gendoc.**prepare_dir**(*app*, *directory*, *delete=False*)
Create apidoc dir, delete contents if delete is True.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **directory** (*str*) – the apidoc directory. you can use relative paths here
- **delete** (*bool*) – if True, deletes the contents of apidoc. This acts like an override switch.

Returns None

Return type None

Raises None

jinjaapidoc.gendoc.**recurse_tree**(*app*, *env*, *src*, *dest*, *excludes*, *followlinks*, *force*, *dryrun*, *private*, *suffix*)
Look for every file in the directory tree and create the corresponding ReST files.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **env** (jinja2.Environment) – the jinja environment
- **src** (str) – the path to the python source files
- **dest** (str) – the output directory
- **excludes** (list) – the paths to exclude
- **followlinks** (bool) – follow symbolic links
- **force** (bool) – overwrite existing files
- **dryrun** (bool) – do not generate files
- **private** (bool) – include "_private" modules
- **suffix** (str) – the file extension

jinjaapidoc.gendoc.**shall_skip**(*app*, *module*, *private*)
Check if we want to skip this module.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **module** (str) – the module name
- **private** (bool) – True, if privates are allowed

jinjaapidoc.gendoc.**write_file**(*app*, *name*, *text*, *dest*, *suffix*, *dryrun*, *force*)
Write the output file for module/package <name>.

Parameters

- **app** (sphinx.application.Sphinx) – the sphinx app
- **name** (str) – the file name without file extension

- **text** (str) – the content of the file
- **dest** (str) – the output directory
- **suffix** (str) – the file extension
- **dryrun** (bool) – If True, do not create any files, just log the potential location.
- **force** (bool) – Overwrite existing files

Returns None

Raises None

jinjaapidoc.gendoc.**AUTOSUMMARYTEMPLATE_DIR = 'autosummarytemplates'**
Templates for autosummary

jinjaapidoc.gendoc.**INITPY = '__init__.py'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

jinjaapidoc.gendoc.**MODULE_TEMPLATE_NAME = 'jinjaapi_module.rst'**
Name of the template that is used for rendering modules.

jinjaapidoc.gendoc.**PACKAGE_TEMPLATE_NAME = 'jinjaapi_package.rst'**
Name of the template that is used for rendering packages.

jinjaapidoc.gendoc.**PY_SUFFIXES = set(['.py', '.pyx'])**
set() -> new empty set object set(iterable) -> new set object

Build an unordered collection of unique elements.

jinjaapidoc.gendoc.**TEMPLATE_DIR = 'templates'**
Built-in template dir for jinjaapi rendering

jinjaapidoc.gendoc.**logger = <sphinx.util.logging.SphinxLoggerAdapter object>**
LoggerAdapter allowing type and subtype keywords.

Module contents

Functions

setup(app)	Setup the sphinx extension

jinjaapidoc.**setup**(*app*)
Setup the sphinx extension

This will setup autodoc and autosummary. Add the *ext.ModDocstringDocumenter*. Add the config values. Connect builder-inited event to *gendoc.main()*.

Parameters **app** (sphinx.application.Sphinx) – the sphinx app

Returns None

Return type None

Raises None

4.4 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:

4.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/storax/jinjaapidoc/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.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/storax/jinjaapidoc/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 :)

4.4.2 Get Started!

Ready to contribute? Here's how to set up *jinjaapidoc* for local development.

1. Fork the *jinjaapidoc* repo on GitHub.
2. Clone your fork locally:

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

3. Create a branch for local development:

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

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. 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
```

6. Submit a pull request through the GitHub website.

4.4.3 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.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/storax/jinjaapidoc under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

4.4.4 Tips

To run a subset of tests:

```
$ py.test test/test_jinjaapidoc.py
```

4.5 Credits

4.5.1 Development Lead

- David Zuber (@storax)

4.5.2 Contributors

- Ashley Whetter (@awhetter)

4.6 History

4.6.1 0.1.1 (2015-02-02)

- First release on PyPI.

4.6.2 0.2.0 (2015-02-03)

- Add autosummary template

4.6.3 0.2.1 (2015-03-11)

- Support Sphinx 1.3
- Fix recursion

4.6.4 0.3.0 (2015-05-23)

- Ignore classes and functions that got imported from other modules

4.6.5 0.4.0 (2016-07-09)

- Add `jinjaapi_include_from_all` option to include members of a module or package that are listed in `__all__` (#9). Thanks to @awhetter.

4.6.6 0.5.0 (2019-04-14)

- Fix compatibility with Sphinx >= 1.8.
- Remove Python 2 tests.

CHAPTER 5

Feedback

If you have any suggestions or questions about **Jinja Api Documentation** feel free to email me at zuber.david@gmx.de.

If you encounter any errors or problems with **Jinja Api Documentation**, please let me know! Open an Issue at the GitHub https://github.com/storax/jinjaapidoc main repository.

5.1 Indices and tables

- genindex
- modindex
- search

Python Module Index

j

Index

Symbols

A

C

D

G

I

J

L

M

N

O

P

R

S

T

W