sphinx-automodapi

Release 0.11.dev0

Feb 20, 2019

Contents

The sphinx-automodapi package provides Sphinx directives that help faciliate the automatic generation of API documentation pages for Python package modules. It was originally developped for the Astropy project, but is now developed as a standalone package that can be used for any project.

CHAPTER 1

Installation

This extension requires Sphinx 1.3 or later, and can be installed with:

```
pip install sphinx-automodapi
```

The extension is also available through conda package management system. It can be installed with:

```
conda install -c astropy sphinx-automodapi
```

CHAPTER 2

Quick start

To use this extension, you will need to add the following entry to the `extensions` list in your Sphinx `conf.py` file:

```
extensions = ['sphinx_automodapi.automodapi']
numpydoc_show_class_members = False
```

You can then add an `automodapi` block anywhere that you want to generate documentation for a module:

```
.. automodapi:: mypackage.mymodule
```

This will add a section with the docstring of the module, followed by a list of functions, and by a list of classes. For each function and class, a full API page will be generated. The `numpydoc_show_class_members=False` option is needed to avoid having methods and attributes of classes being shown multiple times.

By default, sphinx_automodapi will try and make a diagram showing an inheritance graph of all the classes in the module. This requires graphviz to be installed. To disable the inheritance diagram, you can do:

```
.. automodapi:: mypackage.mymodule
   :no-inheritance-diagram:
```

The `automodapi` directive takes other options that are described in more detail in the *User guide* below.

If you are documenting a module which imports classes from other files, and you want to include an inheritance diagram for the classes, you may run into Sphinx warnings, because the class may be documented as e.g. `astropy.table.Table` but the class really lives at `astropy.table.core.Table`. To fix this you can make use of the `'sphinx_automodapi.smart_resolver'` Sphinx extension, which will automatically try and resolve such differences. In this, case, be sure to include:

```
extensions = ['sphinx_automodapi.automodapi',
              'sphinx_automodapi.smart_resolver']
```

in your `conf.py` file.

CHAPTER 3

User guide

3.1 Automatically generating module documentation with automodapi

3.1.1 Overview

The main Sphinx directive provided by the sphinx-automodapi package is the `automodapi` directive. As described in the *Quick start* guide, before you use the directive, you will need to make sure the following extension is included in the `extensions` entry of your documentation's `conf.py` file:

```
extensions = ['sphinx_automodapi.automodapi']
```

You can then add an `automodapi` block anywhere that you want to generate documentation for a module:

```
.. automodapi:: mypackage.mymodule
   <options go here>
```

This will add a section with the docstring of the module, followed by a list of functions, and by a list of classes. For each function and class, a full API page will be generated.

The automodapi directive and allowed options are described in more detail below.

3.1.2 In detail

This directive takes a single argument that must be a module or package. It will produce a block of documentation that includes the docstring for the package, an *Overview* directive, and an *automod-diagram directive* if there are any classes in the module. If only the main docstring of the module/package is desired in the documentation, use automodule instead of *automodapi*.

It accepts the following options:

- **`:include-all-objects:`** If present, include not just functions and classes, but all objects. This includes variables, for which a possible docstring after the variable definition will be shown.

- **:no-inheritance-diagram:** If present, the inheritance diagram will not be shown even if the module/package has classes.
- **:skip: str** This option results in the specified object being skipped, that is the object will *not* be included in the generated documentation. This option may appear any number of times to skip multiple objects.
- **:no-main-docstr:** If present, the docstring for the module/package will not be generated. The function and class tables will still be used, however.
- **:headings: str** Specifies the characters (in one string) used as the heading levels used for the generated section. This must have at least 2 characters (any after 2 will be ignored). This also *must* match the rest of the documentation on this page for sphinx to be happy. Defaults to "-^", which matches the convention used for Python's documentation, assuming the automodapi call is inside a top-level section (which usually uses '=').
- **:no-heading:** If specified do not create a top level heading for the section. That is, do not create a title heading with text like "packagename Package". The actual docstring for the package/module will still be shown, though, unless `:no-main-docstr:` is given.
- **:allowed-package-names: str** Specifies the packages that functions/classes documented here are allowed to be from, as comma-separated list of package names. If not given, only objects that are actually in a subpackage of the package currently being documented are included.
- **:inherited-members: / :no-inherited-members:** The global sphinx configuration option `automodsumm_inherited_members` decides if members that a class inherits from a base class are included in the generated documentation. The option `:inherited-members:` or `:no-inherited-members:` allows the user to overrride the global setting.

This extension also adds three sphinx configuration options:

- **automodapi_toctreedirnm** This must be a string that specifies the name of the directory the automodsumm generated documentation ends up in. This directory path should be relative to the documentation root (e.g., same place as `index.rst`). Defaults to `'api'`.
- **automodapi_writereprocessed** Should be a bool, and if *True*, will cause *automodapi* to write files with any *automodapi* sections replaced with the content Sphinx processes after *automodapi* has run. The output files are not actually used by sphinx, so this option is only for figuring out the cause of sphinx warnings or other debugging. Defaults to *False*.
- **automodsumm_inherited_members** Should be a bool and if `True` members that a class inherits from a base class are included in the generated documentation. Defaults to `False`.

3.2 Generating tables of module objects with automodsumm

3.2.1 Overview

The `automodsumm` directive takes all objects in a module and generates a table of these objects and associated API pages. The *automodapi* directive then calls `automodsumm` once for functions and once for classes, and adds the module docstring - but effectively, the bulk of the work in creating the API tables and pages is done by `automodsumm`.

Nevertheless, in most cases, users should not need to call `automodsumm` directly, except if finer control is desired. The syntax of the directive is:

```
.. automodsumm:: mypackage.mymodule
   <options go here>
```

The automodsumm directive is described in more detail below.

3.2.2 In detail

This directive will produce an "autosummary"-style table for public attributes of a specified module. See the sphinx.ext.autosummary extension for details on this process. The main difference from the autosummary directive is that autosummary requires manually inputting all attributes that appear in the table, while this captures the entries automatically.

This directive requires a single argument that must be a module or package.

It also accepts any options supported by the autosummary directive- see sphinx.ext.autosummary for details. It also accepts some additional options:

- **`:classes-only:`** If present, the autosummary table will only contain entries for classes. This cannot be used at the same time with `:functions-only:` or `:variables-only:`.
- **`:functions-only:`** If present, the autosummary table will only contain entries for functions. This cannot be used at the same time with `:classes-only:` or `:variables-only:`.
- **`:variables-only:`** If present, the autosummary table will only contain entries for variables (everything except functions and classes). This cannot be used at the same time with `:classes-only:` or `:functions-only:`.
- **`:skip: obj1, [obj2, obj3, ...]`** If present, specifies that the listed objects should be skipped and not have their documentation generated, nor be included in the summary table.
- **`:allowed-package-names: pkgormod1, [pkgormod2, pkgormod3, ...]`** Specifies the packages that functions/classes documented here are allowed to be from, as comma-separated list of package names. If not given, only objects that are actually in a subpackage of the package currently being documented are included.
- **`:inherited-members:` or `:no-inherited-members:`** The global sphinx configuration option `automodsumm_inherited_members` decides if members that a class inherits from a base class are included in the generated documentation. The flags `:inherited-members:` or `:no-inherited-members:` allows overrriding this global setting.

This extension also adds two sphinx configuration options:

- **`automodsumm_writereprocessed`** Should be a bool, and if `True`, will cause *automodsumm* to write files with any `automodsumm` sections replaced with the content Sphinx processes after `automodsumm` has run. The output files are not actually used by sphinx, so this option is only for figuring out the cause of sphinx warnings or other debugging. Defaults to `False`.
- **`automodsumm_inherited_members`** Should be a bool and if `True`, will cause *automodsumm* to document class members that are inherited from a base class. This value can be overriden for any particular automodsumm directive by including the `:inherited-members:` or `:no-inherited-members:` options. Defaults to `False`.

automod-diagram directive

This directive will produce an inheritance diagram like that of the sphinx.ext.inheritance_diagram extension.

This directive requires a single argument that must be a module or package. It accepts no options.

Note: Like 'inheritance-diagram', 'automod-diagram' requires graphviz to generate the inheritance diagram.

Python Module Index

s

Index

S