sphinxcontrib.traceables

Release 0.2.0

unknown

2020-07-12

CONTENTS

Release 0.2.0

Date 2020-07-12

CHAPTER

ONE

CONTENTS

1.1 Getting started

1.1.1 Overview

This extension provides functionality to define traceable items and relationships between them in documentation generated by Sphinx. It also offers visualization of the traceables in various forms, such as relationship graphs.

More information is available here:

- Documentation: https://superzerg.github.io/sphinxcontrib-traceables
- Download: https://pypi.python.org/pypi/sphinxcontrib-traceables0.2
- Development: https://github.com/superzerg/sphinxcontrib-traceables

As the original package (https://github.com/t4ngo/sphinxcontrib-traceables) seems to be unmaintained, I updated it so it is compatible with python3 and Sphinx >= 2.1. Thanks to t4ngo for the great extension and to rexut for the upgrade to python3.

1.1.2 Installation

Install this package using `pip install sphinxcontrib-traceables0.2`, or from source using `python setup.py install`. Then add it to your project's Sphinx configuration file `conf.py`:

```
extensions = ['sphinxcontrib-traceables']
```

1.1.3 Minimal example

In your project's documentation, you can define a new traceable as shown below. Note that the tag `LOREM-IPSUM` given to the traceable should be unique throughout the documentation, because it is used to reference the traceable.

```
.. traceable:: LOREM-IPSUM
   :title: Lorem ipsum

   Lorem ipsum dolor sit...
```

The traceable defined above can be referenced in line as follows:

```
Lorem ipsum :traceable:`LOREM-IPSUM` dolor sit...
```

See *Usage* for more information about this and more functionality provided by this extension.

1.2 Usage

1.2.1 Defining traceables

Traceables are defined using the `traceable` directive described below.

`.. traceable:: TAG`
Create a traceable item. The `TAG` should be a unique string throughout the documentation, because it is used to reference the traceable.

The traceable can be given attributes, such as a title, a version, and relationships with other traceables. Attributes are specified using the form `:<option-name>: <option-value>`[1]. For example, the text below defines a traceable with tag `LOREM-IPSUM`, title "Lorem ipsum", and content "Lorem ipsum dolor sit...":

```
.. traceable:: LOREM-IPSUM
   :title: Lorem ipsum

   Lorem ipsum dolor sit...
```

Traceable directives have the following properties:

- A single argument defining the traceable's tag (`.. traceable:: <TAG>`)
  - The tag must be unique throughout the documentation
  - The tag may not contain spaces
- Zero or more options (`:<option-name>: <option-value>`)
  - The special option `title` defines the traceable's title
  - Options with a valid relationship name define relationships which this traceable has with other traceable
  - Arbitrary option names and values are allowed; if not special options as listed above, the values are stored as traceable attributes and can be used for filtering/querying traceables
- Optional content
  - If content is given, it will be parsed as regular ReST.

1.2.2 Referencing traceables

Traceables can be referenced using the `traceable` role described below.

`:traceable:`
Create a reference to a traceable defined elsewhere in the documentation. For example:

```
Lorem ipsum :traceable:`LOREM-IPSUM` dolor sit...
```

[1] The formal specification of reStructuredText directives is documented here: http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#directives

1.2.3 Showing traceables matrices

Relationships between traceables can be shown using the `traceable-matrix` directive described below.

`.. traceable-matrix::`
Generate a traceables matrix. The matrix shows pairs of traceables which are related to each other by a given relationship (see the `:relationship:` option below). Various formats are available for showing the pairs of traceables (see the `:format:` option below).

The following options are available for this directive:

`:relationship:` – name of traceables relationship *(required)* Specify which relationship between traceables to show in the generated matrix.

`:filter-primaries:` – traceable filter expression *(optional)* Specify a filter to limit the primary traceables included in the matrix.

If this option is not given, then all traceables that have one or more relationships of the type specified by `:relationship:` will be included.

`:filter-secondaries:` – traceable filter expression *(optional)* Specify a filter to limit the secondary traceables included in the matrix.

If this option is not given, then all traceables that have one or more relationships of the reverse of the type specified by `:relationship:` will be included.

`:format:` – name of format *(optional)* Specify the matrix format to generate. The following formats are available:

- `:format: table` *(default)*

  Traditional traceability table format. This format allows the following additional options to be set:

  `:split-secondaries:` – positive integer The maximum number of columns to output. If more columns would have been necessary, multiple tables will be generated.

  `:split-primaries:` – positive integer The maximum number of rows to output. If more rows would have been necessary, multiple tables will be generated.

- `:format: columns`

  2-column table of related traceables

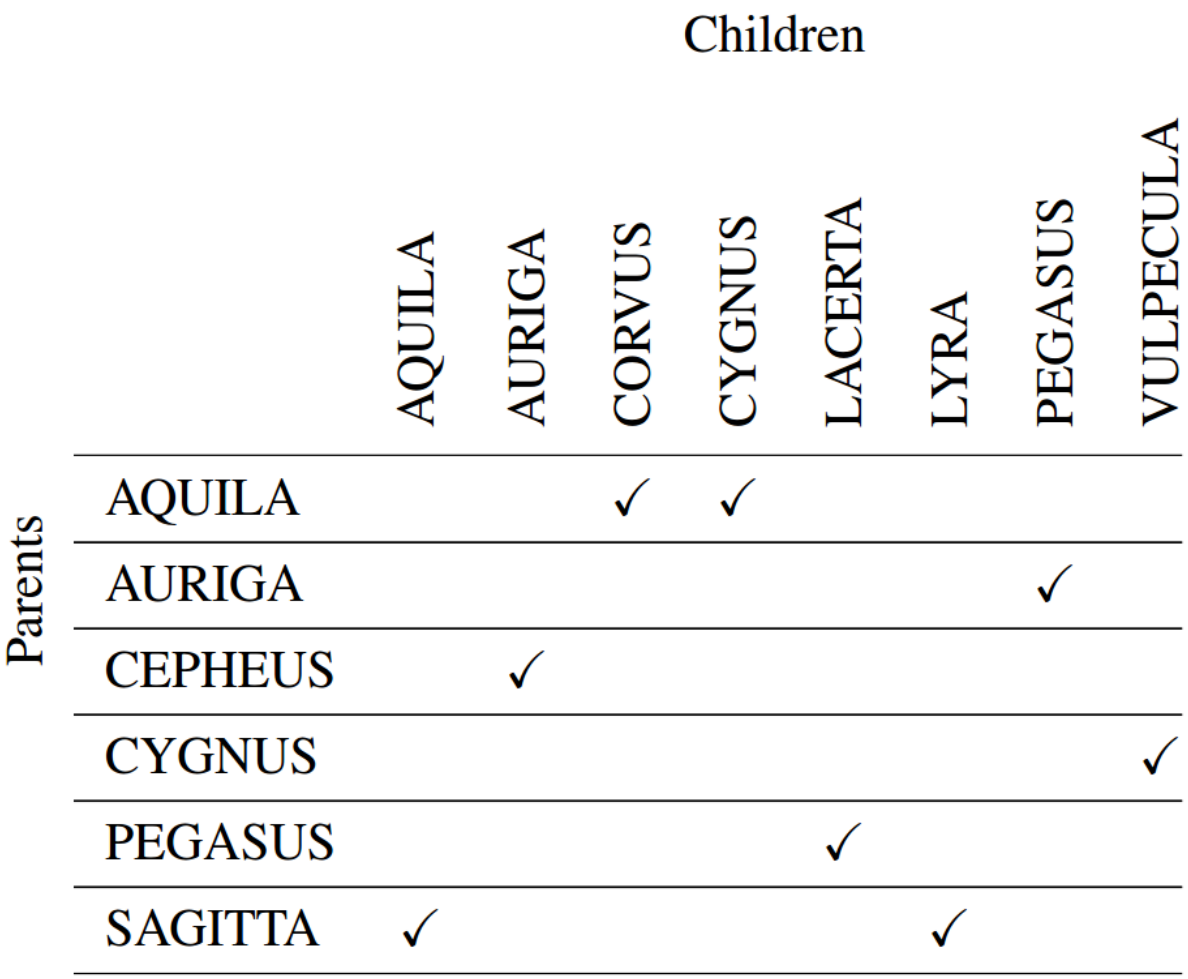
- `:format: list`

  2-level list of related traceables

For example, the directive shown below would generate a table showing all traceables related to each other with the `children` relationship:

```
.. traceable-matrix::
   :relationship: children
   :format: table
```

The directive shown above creates a traceability matrix as shown below. The image below is a screenshot from the latex-PDF generated for the example constellation project which is part of this extension's source code.



1.2.4 Showing traceables graphs

Relationships between traceables can be shown using the `traceable-graph` directive described below.

`.. traceable-graph::`
Generate a traceables graph. The graph shows all traceables which are related to the give traceables (see the `:tags:` option below).

The following options are available for this directive:

`:tags:` – traceable tags *(required)* Specify comma separated tags for which the graph will be built. All traceables related (with any depth) to these tags with the given `:relationships:` will be part of the resulting graph.

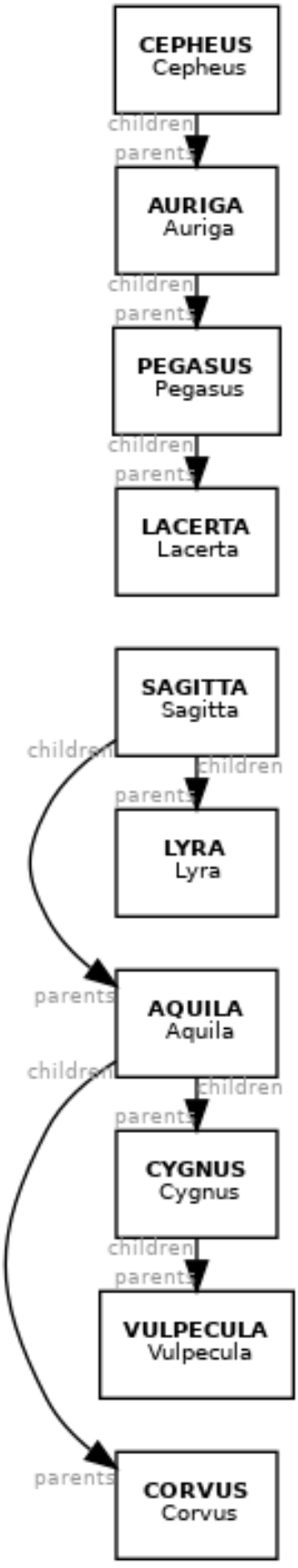
`:relationships:` – name of traceables relationship *(optional)* Specify which relationships between traceables to show in the generated graph. If not specifed, all relationships are shown.

`:caption:` – string *(optional)* Caption for the graph.

For example, the directive shown below would generate a graph showing all traceables related to each other with the `children` relationship:

```
.. traceable-graph::
   :tags: SAGITTA, PEGASUS
```

The directive shown above creates a traceability graph as shown below. The image below is taken from the HTML generated for the example constellation project which is part of this extension's source code.


CEPHEUS
Cepheus
children
parents
AURIGA
Auriga
children
parents
PEGASUS
Pegasus
children
parents
LACERTA
Lacerta
SAGITTA
Sagitta
children
children
parents
LYRA
Lyra
parents
AQUILA
Aquila
children
children
parents
CYGNUS
Cygnus
children
parents
VULPECULA
Vulpecula
parents
CORVUS
Corvus

1.3 Extension internals

1.3.1 Logical architecture

Behavioral architecture model

When Sphinx is run, this extension performs the steps listed below.

1. Initialization
   - Sphinx calls `setup()`, which registers the extension's directives, roles, event handlers, etc.
2. Parsing source files
   - When Sphinx encounters a traceables directive or role while parsing input, it calls the associated class.
   - The directive or role class processes its input and creates one or more nodes which are then inserted into the resulting doctree; these will be processed later after all source files have been parsed.
   - A directive that defines a new traceable item adds that item to the traceables cache that this extension maintains, so that later processing logic knows about all traceable items.
3. Processing doctrees
   - During initialization, this extension registered a handler for the event-doctree-resolved event; that handler is called after all source files have been parsed into doctrees.
   - The event handler calls the *ProcessorManager* to process the given doctree; it in turn calls classes derived from *ProcessorBase* to perform the various traceables functionalities.
4. Maintaining state information
   - During initialization, this extension registered a handler for the event-env-purge-doc event; that handler is called to clean up any state this extension may keep related to a given source file.
   - The event handler calls the *TraceablesStorage* to remove all information stored in its cache related to the given source file.

During parsing of source files

This extension adds the following directives and roles which are used by Sphinx during parsing of source.

Traceable directive

When Sphinx encounters a traceable directive during parsing, a *TraceableDirective* is called to process the directive. It creates a target node, so that the traceable can be referenced from elsewhere in the documentation, and a presentation node, to show the traceable's definition at the directive's location in the documentation.

The attributes defined in the traceable directive may contain references to other traceables which cannot be resolved until the entire doctree has been resolved. The attributes are therefore stored in a `traceable_attribute_list` node for later processing.

Traceable role

When Sphinx encounters a traceable role during parsing, a *traceable_xref* cross reference node is created. Later on that note will be replaced by an appropriate final node.

After doctree has been resolved

This extension registers the `process_traceables_in_doctree()` to be called when the `doctree-resolved` event fires. That callback function invokes the `TraceablesProcessor` to process the main business logic of this extension.

The `TraceablesProcessor` performs the following activities:

1. Collect all traceables defined throughout the documentation
2. Analyze relationships between traceables; this is done with help from the `RelationshipManager` class
3. Process `traceable_attribute_list` nodes; these are part of traceable directives
4. Resolve `traceable_xref` cross reference nodes; these are created by traceable roles, amongst other possible sources

Purging of old state

This extension registers the `purge_traceables()` to be called when the `env-purge-doc` event fires. That callback function removes the relevant data from the `TraceablesStorage`.

1.3.2 The `infrastructure` module: Infrastructure for processing traceables

class sphinxcontrib.traceables.infrastructure.**FormatProcessorBase**(*app*, *process_node_type=None*)

classmethod directive_format_choice(*argument*)

formatter_defaults = {93986948901016: <sphinxcontrib.traceables.display.TraceableDisp

formatters = {93986948901016: {'admonition': <sphinxcontrib.traceables.display.Trace

classmethod get_default_formatter()

get_formatter(*node*, *name*)

classmethod get_registered_formatters()

process_node(*node*, *doctree*, *docname*)

process_node_with_formatter(*node*, *formatter*, *doctree*, *docname*)

classmethod register_formatter(*name*, *formatter*, *default=False*)

class sphinxcontrib.traceables.infrastructure.**InfrastructureLogging**

document_warning(*env*, *msg*, *lineno=None*)

node_warning(*env*, *msg*, *node*)

class sphinxcontrib.traceables.infrastructure.**ProcessorBase**(*app*, *process_node_type=None*)

Error
alias of sphinx.errors.ExtensionError

process_doctree(*doctree*, *docname*)

process_node(*node*, *doctree*, *docname*)

class sphinxcontrib.traceables.infrastructure.**ProcessorManager**(*app*)

process_doctree(*doctree*, *docname*)

processor_classes = []

classmethod register_processor_classes(*processors*)

class sphinxcontrib.traceables.infrastructure.**Traceable**(*target_node*, *unresolved_tag=None*)

has_title()

property is_unresolved

make_reference_node(*builder*, *docname*)

classmethod split_tags_string(*tags_string*)

property title

class sphinxcontrib.traceables.infrastructure.**TraceablesFilter**(*traceables*)

filter(*expression_string*)

traceable_matches(*matcher*, *traceable*)

class sphinxcontrib.traceables.infrastructure.**TraceablesStorage**(*env*)

add_traceable(*node*)

analyze_relationship_types()

get_or_create_traceable_by_tag(*tag*)

get_relationship_direction(*name*)

get_relationship_opposite(*name*)

get_traceable_by_tag(*tag*)

is_valid_relationship(*name*)

purge(*docname*)

property traceables_dict

property traceables_set

sphinxcontrib.traceables.infrastructure.**add_static_files**(*app*)

sphinxcontrib.traceables.infrastructure.**copy_static_files**(*app*, *exception*)

sphinxcontrib.traceables.infrastructure.**process_doctree**(*app*, *doctree*, *docname*)

sphinxcontrib.traceables.infrastructure.**purge_docname**(*app*, *env*, *docname*)

sphinxcontrib.traceables.infrastructure.**setup**(*app*)

1.3.3 The `traceables` module: Core traceables functionality

class sphinxcontrib.traceables.traceables.**DefaultDict**(*default*)

class sphinxcontrib.traceables.traceables.**RelationshipsProcessor**(*app*, *process_node_type=None*)

process_doctree(*doctree*, *docname*)

class sphinxcontrib.traceables.traceables.**TraceableDirective**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

create_index_node(*env*, *tag*, *attributes*)

create_target_node(*env*, *tag*, *attributes*)

final_argument_whitespace = False

has_content = True

option_spec = {}

optional_arguments = 0

required_arguments = 1

run()

class sphinxcontrib.traceables.traceables.**XrefProcessor**(*app*, *process_node_type=None*)

process_doctree(*doctree*, *docname*)

sphinxcontrib.traceables.traceables.**setup**(*app*)

class sphinxcontrib.traceables.traceables.**traceable_xref**(*rawsource=''*, **children*, ***attributes*)

1.3.4 The `matrix` module: Matrices of traceables

class sphinxcontrib.traceables.matrix.**BulletMatrixFormatter**

format(*app*, *docname*, *node*, *matrix*, *options*)

class sphinxcontrib.traceables.matrix.**MatrixFormatterBase**

format(*app*, *docname*, *node*, *matrix*, *options*)

class sphinxcontrib.traceables.matrix.**MatrixProcessor**(*app*)

build_traceable_matrix(*forward*, *filter1*, *filter2*)

process_node_with_formatter(*matrix_node*, *formatter*, *doctree*, *docname*)

class sphinxcontrib.traceables.matrix.**TableMatrixFormatter**

create_cross_table(*app*, *docname*, *node*, *matrix*, *options*)

format(*app*, *docname*, *node*, *matrix*, *options*)

class sphinxcontrib.traceables.matrix.**TraceableMatrix**(*forward_relationship*, *backward_relationship*)

add_primary(*primary*)

add_secondary(*secondary*)

add_traceable_pair(*primary*, *secondary*)

ascii_table()

property backward_relationship

calculate_ranges(*total_length*, *max_range_length*)

property forward_relationship

get_boolean_row(*primary*)

get_relatives(*primary*)

property primaries

property secondaries

split(*max_secondaries*, *max_primaries=None*)

class sphinxcontrib.traceables.matrix.**TraceableMatrixDirective**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

option_spec = {'filter-primaries': <function unchanged>, 'filter-secondaries': <func

optional_arguments = 0

required_arguments = 0

run()

class sphinxcontrib.traceables.matrix.**TwoColumnMatrixFormatter**

format(*app*, *docname*, *node*, *matrix*, *options*)

sphinxcontrib.traceables.matrix.**setup**(*app*)

class sphinxcontrib.traceables.matrix.**traceable_checkmark**(*rawsource=''*, **children*, ***attributes*)

Placeholder node to be replaced by a builder-specific checkmark symbol.

This node type has no traceable-specific attributes.

class sphinxcontrib.traceables.matrix.**traceable_matrix**(*rawsource=''*, **children*, ***attributes*)

Placeholder node to be replaced by a traceables matrix.

traceables-relationship
The name of the relationship to display between primary and secondary traceables.

traceables-format
The name of the format with which to display the data.

traceables-filter-primaries
The filter expression to determine which traceables are the primary set.

traceables-filter-secondaries
The filter expression to determine which traceables are the secondary set.

traceables-split-primaries
If set, causes the output to be split after the specified number of primary traceables.

traceables-split-secondaries
If set, causes the output to be split after the specified number of secondary traceables.

class sphinxcontrib.traceables.matrix.**traceable_matrix_crosstable**(*rawsource=''*, **children*, ***attributes*)

Placeholder node to be replaced by a builder-specific matrix.

traceables-matrix
Instance of *TraceableMatrix* storing data to be presented in the output.

sphinxcontrib.traceables.matrix.**visit_traceable_checkmark_latex**(*self*, *node*)

sphinxcontrib.traceables.matrix.**visit_traceable_matrix_crosstable_latex**(*self*, *node*)

1.3.5 The graph module: Visualization of traceables

class sphinxcontrib.traceables.graph.**GraphInput**(*storage*, *relationship_length_pairs*)

add_traceable_walk(*traceable*)

property relationships

property traceables

class sphinxcontrib.traceables.graph.**GraphProcessor**(*app*)

add_dot_node(*dot*, *tag*, *title*, *style*)

add_dot_traceable(*dot*, *traceable*)

construct_graph_input(*traceables*, *relationship_length_pairs*)

generate_dot(*graph_input*)

get_start_traceables(*tags_string*, *node*)

parse_relationships(*input*)

process_doctree(*doctree*, *docname*)

class sphinxcontrib.traceables.graph.**TraceableGraphDirective**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

```
    final_argument_whitespace = False
    has_content = True
    option_spec = {'caption':  <function unchanged>, 'relationships':  <function unchanged
    optional_arguments = 0
    required_arguments = 0
    run()
```

sphinxcontrib.traceables.graph.**setup**(*app*)

class sphinxcontrib.traceables.graph.**traceable_graph**(*rawsource=''*, **children*, ***attributes*)

1.4 Examples

1.4.1 Documenation examples

See the `examples` directory (available on GitHub) within this extension's source code for a complete Sphinx project demonstrating various features provided by this extension.

The documentation obtained from these examples is also directly available here:

- constellation example
- requirements example

1.4.2 Tests examples

Most functionalities of this extension are tested with pytest. The data for the tests are in the `tests/data` directory (available on GitHub) within this extension's source code.

The documentation obtained from the tests not expecting to raise warnings or errors are also directly available here:

- Graph test
- List test
- matrix test

1.5 License

License Apache License

Version 2.0

Date January 2004

URL http://www.apache.org/licenses/LICENSE-2.0

1.5.1 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means *(i)* the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or *(ii)* ownership of fifty percent (50%) or more of the outstanding shares, or *(iii)* beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- If the Work includes a "`NOTICE`" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such `NOTICE` file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a `NOTICE` text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the `NOTICE` file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the `NOTICE` text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the `NOTICE` file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an **"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND**, either express or implied, including, without limitation, any warranties or conditions of **TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE**. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

CHAPTER

TWO

COPYRIGHT

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

G

H

I

M

N

O

P

R

S

T

V

X