Sphinx – a tool for unified documentation

Georg Brandl

… Introductions …

About me

- Instrument control group at JCNS/MLZ
- Writing Tango servers
  - Tango devices with generic interfaces using a Python framework called Entangle

- Python core developer
- Original author of Sphinx

About Sphinx

- sphinx-doc.org

About Sphinx

- sphinx-doc.org

?

About Sphinx

- sphinx-doc.org

?

About Sphinx

- sphinx-doc.org

?

So, what does it really do?

- Processes markup to output formats
  - reStructuredText
  - (Markdown)

  to
  - HTML, ePub
  - LaTeX (→ PDF)
  - Manpages, Texinfo, HTML Help, QtHelp, ...

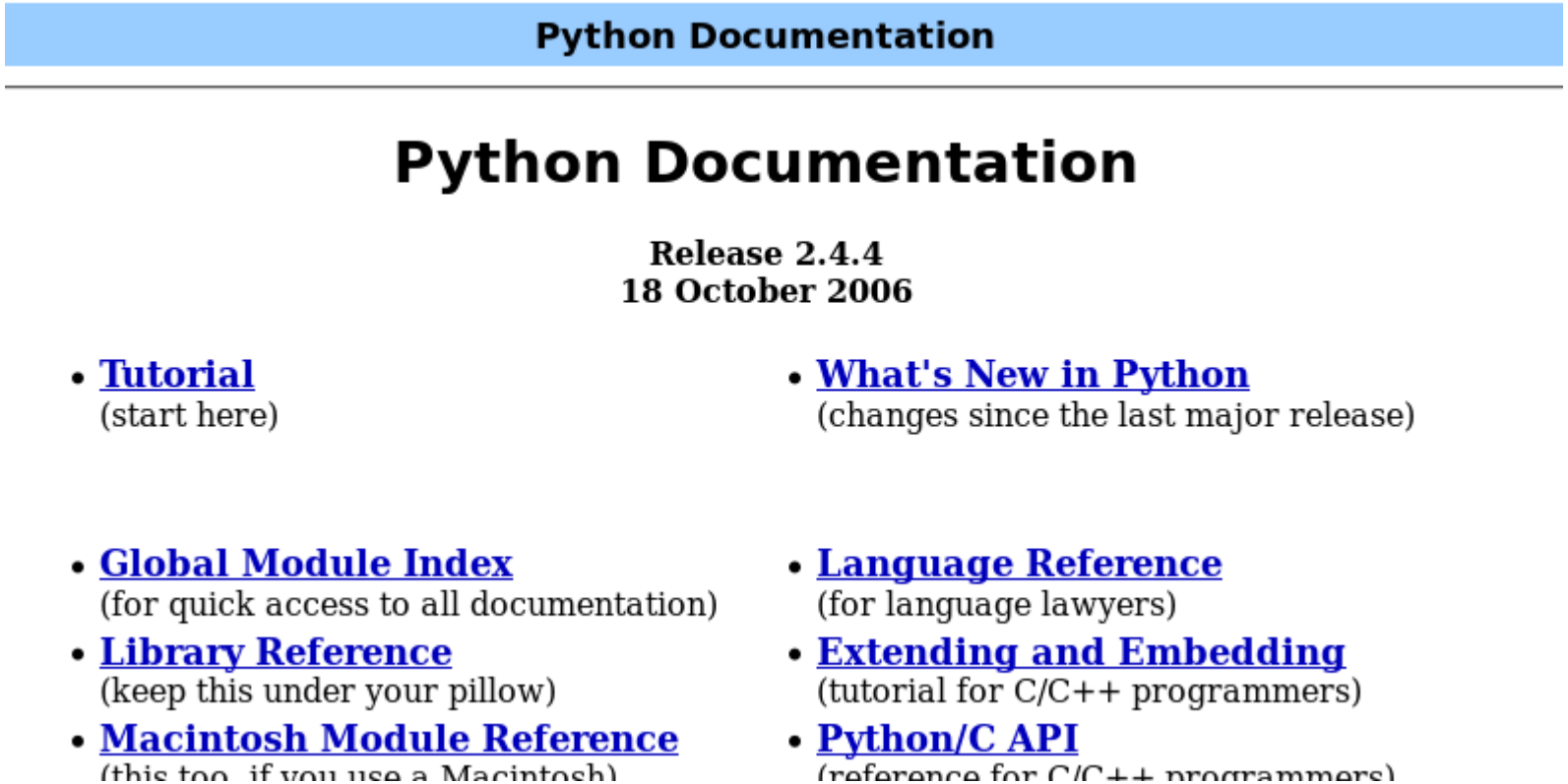
So, what does it really do?

- Lots of built-in semantic markup and features suited for software documentation
- Even more built-in features suited for Python software documentation

- Extensible!

(Short) history

- “Ye olde” Python docs



Python Documentation

Python Documentation

Release 2.4.4
18 October 2006

- **Tutorial**
  (start here)
- **What's New in Python**
  (changes since the last major release)
- **Global Module Index**
  (for quick access to all documentation)
- **Library Reference**
  (keep this under your pillow)
- **Macintosh Module Reference**
  (this too, if you use a Macintosh)
- **Language Reference**
  (for language lawyers)
- **Extending and Embedding**
  (tutorial for C/C++ programmers)
- **Python/C API**
  (reference for C/C++ programmers)

(Short) history

- “Ye olde” Python docs – pre-2007
  - LaTeX source format
  - latex2html + custom hacks conversion

- Problem 1: contributions
- Problem 2: LaTeX hacking required

- Obvious solution: write something in Python
- tailored to the core Python docs

(Short) history

- “Ye olde” Python docs – pre-2007
  - LaTeX source format
  - latex2html + custom hacks conversion

- Problem 1: contributions
- Problem 2: LaTeX hacking required

- Obvious solution: write something in Python
- tailored to the core Python docs – at first!

Status today

- Stable release series 1.4.x
- Available on GitHub (recent move from hg/BitBucket)
- Team of about 5 active developers
- De-facto standard for Python projects
  - Also adopted by a few other languages, like Julia
- Automated deployment and hosting with ReadTheDocs.org
- Used for in-house docs in companies

… Features …

Basics

- Project: directory with `conf.py`
- Many files, one logical hierarchy → cross-referencing
- Still, file structure is output structure (for HTML based backends)
- reStructuredText markup, similar to Markdown but standardized, with defined extensibility – provided by Docutils
- Source code highlighting
- Extensive indices
- Builtin offline search (JS based)
- To build: "`make html`" etc (only thin wrapper around `sphinx-build`)
- Parallel build available

Configuration

- `conf.py` is a Python module
- Config settings are globals in that module after execution
- Can do complex tasks to determine e.g. version number
- Deployment specific behavior, with “tags”

```
sphinx-build -t short
```

- Extensions can introduce their own configuration values
- “Quickstart” tool generates config with most common configuration values and comments
- `conf.py` is also a Sphinx extension!

Quickstart

```
Welcome to the Sphinx 1.4.4 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]:

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other
static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name:
```

Configuration

```
# Sphinx documentation build configuration file

import re
import sphinx


extensions = ['sphinx.ext.autodoc', 'sphinx.ext.doctest', 'sphinx.ext.todo',
              'sphinx.ext.autosummary', 'sphinx.ext.extlinks',
              'sphinx.ext.viewcode']

master_doc = 'contents'
templates_path = ['_templates']

project = 'Sphinx'
copyright = '2007-2016, Georg Brandl and the Sphinx team'
version = sphinx.__released__
release = version
show_authors = True
```

Markup

```
Heading
=======

.. index:: reStructuredText

This is a *sample* paragraph to show **reStructuredText**.
It has `links <http://sphinx-doc.org>`_ and references [1]_.
Some :ref:`internal links <internal>` too.

.. note:: This only shows a small fraction of possible markup.

.. class:: DeviceProxy

   .. method:: exec_cmd(name: str, inarg: any) -> any

      :param name: The command name.

      See also :meth:`read_attr`.

      .. versionadded:: 5.0
```

Domains

- Language-specific markup is grouped into a “domain”
- Supports typical API items for different languages, including indices and doc fields (Javadoc-like `@param foo`)
- Markup items prefixed, e.g.

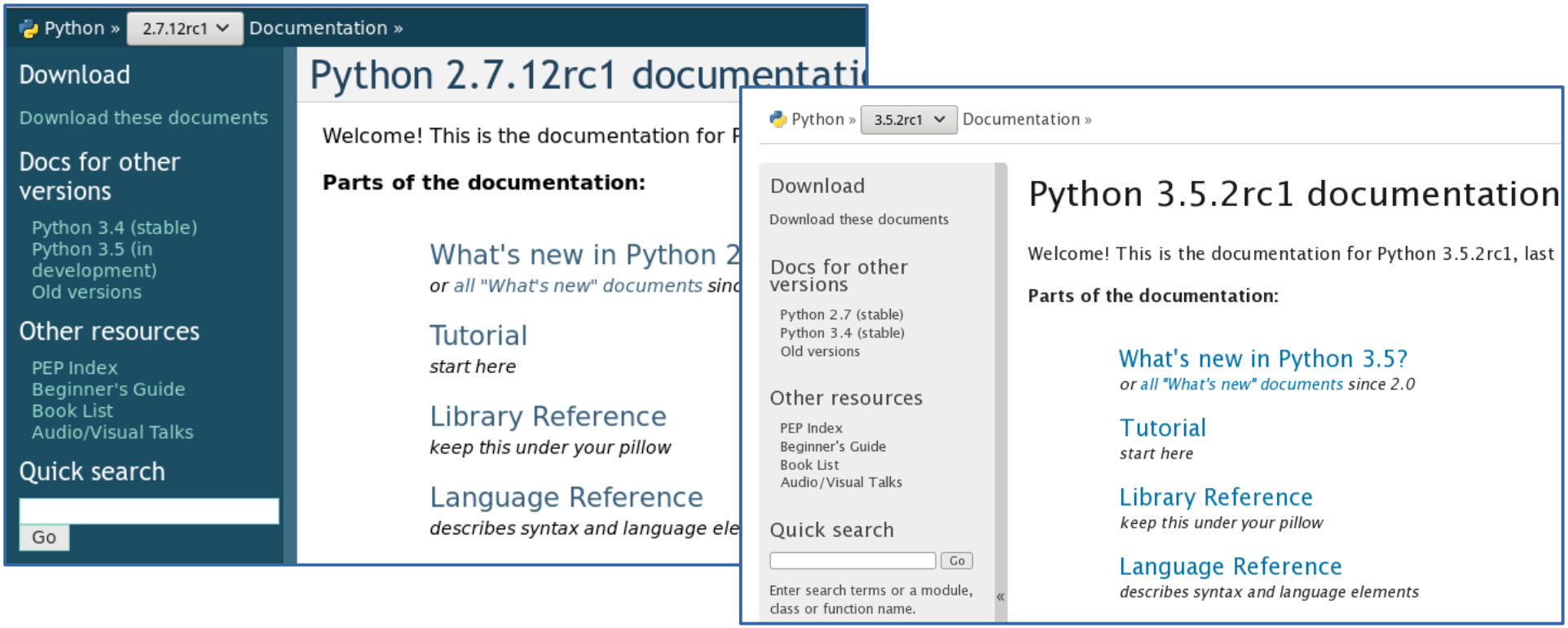
```
.. py:class:: Database

.. cpp:method:: exec_command
```

- Default domain can be selected per-project/per-file
- Built-in: Python, C, C++, JS, reST
- Further domains can be added by extensions

Theming

- HTML output is themable, including templates
- Integration into existing layouts possible
- External themes based on layout frameworks, e.g. twitter bootstrap



Where's the source code?

- Origins: Python docs were (still are) separate from docstrings
- Focus on hand-written, high-quality prose with semantic markup and cross-refs
- But most tools provide “automatic” API documentation
- Synthesis: `autodoc`
  - Structure still given by doc files
  - Pull in docstrings where sensible
  - Allows writing high quality docstrings *and* glue, without being tied to source code structure
  - Keep large examples out of source code

Autodoc

- Imports the Python code
- Granularity is variable

```
The database module
===================

.. automodule:: tango.database
   :members:
```

```
The database module
===================

.. module:: tango.database

In this module, the following
functions are provided:

.. autofunction:: add_device

.. autofunction:: add_server

   Example: ...
```

… Extensions …

Intersphinx

- Each project saves an “inventory” of its API items together with HTML output
- Intersphinx loads that and can generate deep cross-refs to those API items
- For example, projects can link to the main Python documentation
- Inventories can be local or remote (downloaded at build time)

- Connect different doc projects that are maintained individually

More standard extensions

- Internationalization: processes translatable text into gettext .po files and allows translation with standard tools, e.g. web-based systems like Transifex

- Support for LaTeX style math (with PNG, MathJax output)
- Support for diagrams/graphs with Graphviz
  - also automatic inheritance diagrams
- Doctest-ing Python examples
- Showing source code for API items, either formatted with output or just links on repository browser (GitHub or others)
- Link checking

Writing extensions

- Extensions have different ways to hook into the build process:
  - add new output formats
  - add new markup items (directives and roles)

```
.. device::    :dev:`reference`
```

  - add configuration values to be recognized in `conf.py`
  - add domain or domain-specific markup
  - react to events (like “document parsed”, “cross-reference missing”)
  - add format specific things (like JavaScripts for HTML)

- Requires (some) docutils expertise, but tutorial and plenty of examples available

Writing extensions

```
class Todo(Directive):
    has_content = True
    required_arguments = 0
    optional_arguments = 0
    final_argument_whitespace = False

def setup(app):
    app.add_config_value('todo_include_todos', False, 'html')
    app.add_config_value('todo_link_only', False, 'html')

    app.add_node(todo_node,
                 html=(visit_todo_node, depart_todo_node),
                 latex=(visit_todo_node, depart_todo_node))
    app.add_directive('todo', Todo)

    app.connect('doctree-read', process_todos)
    ...
```

Example

```
Phytron motor controllers
-------------------------

These devices support the `Phytron`_ Motor controller:

 * `MCC-1`_
 * `MCC-2`_
 * `OMC/TMC`_

.. autodev:: phytron.Motor
.. autodev:: phytron.Sensor

.. _Phytron: http://www.phytron.de
.. _MCC-2: http://www.phytron.eu/antrieb/index.php?Set_ID=165&PID=9
.. _MCC-1: http://www.phytron.eu/antrieb/index.php?Set_ID=165&PID=39
.. _OMC/TMC: http://www.phytron.eu/antrieb/index.php?Set_ID=165&PID=10
```

Table Of Contents

Previous topic

Motor control devices

Next topic

IMS (Schneider Motion) motor controllers

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Phytron motor controllers

These devices support the Phytron Motor controller:

- MCC-1
- MCC-2
- OMC/TMC

phytron.Motor

class entangle.device.phytron.**Motor**

Bases: **LinearActuator**, **PhytronProps**, **Motor**

Controls one axis of a Phytron MCC, OMC or TMC controller.

Properties

absmax *(DevDouble)*

If set, gives an absolute upper limit for the "value" attribute. This limit should be checked in the server before sending commands to the hardware, except if both absmin and absmax are zero.

Inherited from a base class.

absmin *(DevDouble)*

If set, gives an absolute lower limit for the "value" attribute. This limit should be checked in the server before sending commands to the hardware, except if both absmin and absmax are zero.

Inherited from a base class.

address *(DevDouble - validator: an integer in the range [0, 15])*

Read the Docs

- rtfd.org
- Automated platform that builds and hosts a project
- Not much configuration necessary for normal setups
- Multiple branches/versions available to build, selectable when viewing
- Supports i18n
- Supports downloadable documents (PDF, ePub)
- Might be problematic with C/C++ dependencies (PyTango)

Integration for other languages

- C++ – domain is builtin, autodoc-like functionality supported via Doxygen bridge (`breathe`)
- Uses Doxygen to parse C++, extract doc comments
  - Directives like `.. autodoxygenfile::`
  - Can embed reStructuredText into Doxygen markup
- Java – Javasphinx provides a domain and automatic API docs

- Third-party projects – might need tweaking or additions
- I'm happy to collaborate – just ask!

... Questions?

Addresses of interest:

- http://sphinx-doc.org
- http://github.com/sphinx-doc/sphinx
- sphinx-users@googlegroups.com

- g.brandl@fz-juelich.de