A lion, a head, and a dash of YAML

Extending Sphinx to automate your documentation

FOSDEM 2018

@stephenfin

reStructuredText, Docutils & Sphinx

1

```
A little reStructuredText
=========================

This document demonstrates some basic features of |rst|. You can use
**bold** and *italics*, along with ``literals``. It’s quite similar
to `Markdown`_ but much more extensible. CommonMark may one day
approach this [1]_, but today is not that day. `docutils`__ does all
this for us.

.. |rst| replace:: **reStructuredText**
.. _Markdown: https://daringfireball.net/projects/markdown/
.. [1] https://talk.commonmark.org/t/444
.. __  http://docutils.sourceforge.net/
```

```
A little reStructuredText
=========================

This document demonstrates some basic features of |rst|. You can use
**bold** and *italics*, along with ``literals``. It’s quite similar
to `Markdown`_ but much more extensible. CommonMark may one day
approach this [1]_, but today is not that day. `docutils`__ does all
this for us.

.. |rst| replace:: **reStructuredText**
.. _Markdown: https://daringfireball.net/projects/markdown/
.. [1] https://talk.commonmark.org/t/444
.. __  http://docutils.sourceforge.net/
```

A little reStructuredText

This document demonstrates some basic features of **reStructuredText**. You can use **bold** and *italics*, along with `literals`. It's quite similar to Markdown but much more extensible. CommonMark may one day approach this [1], but today is not that day. docutils does all this for us.

[1] https://talk.commonmark.org/t/444/

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

A little more reStructuredText

The extensibility really comes into play with directives and roles. We can do things like link to RFCs (RFC 2324, anyone?) or generate some more advanced formatting (I do love me some H_2O).

> **Warning**
> The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The only limit is your imagination/interest.

reStructuredText provides the syntax

Docutils provides the parsing

reStructuredText provides the syntax

Docutils provides the parsing

Sphinx provides the cross-referencing and file generation

Docutils use readers, parsers, transforms, and writers

Docutils works with individual files

Docutils use readers, parsers, transforms, and writers

Docutils works with individual files

Sphinx uses readers, writers, transforms, and builders

Sphinx works with multiple, cross-referenced files

Documentation tool

Multiple output formats

Extensive cross-referencing support

Extensions

Documentation tool

Multiple output formats

Extensive cross-referencing support

Extensions

```
sphinx-quickstart

sphinx-build

sphinx-apidoc

sphinx-autogen
```

sphinx-quickstart

sphinx-build

sphinx-apidoc

sphinx-autogen

Let’s Get To Extending…

2

Current version of Sphinx (1.7.0) - APIs may change

Python knowledge is expected

Some possible references to OpenStack projects

See github.com/stephenfin/fosdem-sphinx-demo for more

Extensions are registered via **`sphinx.application.Application`**

`add_builder`	(Builders)
`add_config_value`	(Config Values)
`add_domain`	(Domains)
`add_event`	(Events)
`add_node`	(docutils Nodes)
`add_directive`	(Directives)
`add_role`	(Interpreted Text Roles, a.k.a. Roles)
`connect, disconnect`	(Hooks)
...	...

Extensions are registered via `sphinx.application.Application`

`add_builder`	(Builders)
`add_config_value`	(Config Values)
`add_domain`	(Domains)
`add_event`	(Events)
`add_node`	(docutils Nodes)
`add_directive`	(Directives)
`add_role`	(Interpreted Text Roles, a.k.a. Roles)
`connect, disconnect`	(Hooks)
`...`	...

Interpreted Text Roles

(a.k.a. roles)

3

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

```rst
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

```python
def xyz_role(name, rawtext, text, lineno, inliner, options={},
             content=[]):
  # code...

def setup(app):
  app.add_role('xyz', xyz_role)
  return {'version': '1.0', 'parallel_read_safe': True}
```

```
Fixes
=====

* #2951: Add ``--implicit-namespaces`` PEP-0420 support to apidoc.
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram.
* #2471: Add config variable for default doctest flags.
* Convert linkcheck builder to requests for better encoding
  handling
* #2463, #2516: Add keywords of "meta" directive to search index
```

source/changes.rst

```
Fixes
=====

* #2951: Add ``--implicit-namespaces`` PEP-0420 support to apidoc.
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram.
* #2471: Add config variable for default doctest flags.
* Convert linkcheck builder to requests for better encoding
  handling
* #2463, #2516: Add keywords of "meta" directive to search index
```

source/changes.rst

```rst
Fixes
=====

* #2951: Add ``--implicit-namespaces`` PEP-0420 support to apidoc.
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram.
* #2471: Add config variable for default doctest flags.
* Convert linkcheck builder to requests for better encoding
  handling
* #2463, #2516: Add keywords of "meta" directive to search index
```

source/changes.rst

```rst
Fixes
=====

* Add ``--implicit-namespaces`` PEP-0420 support to apidoc
  (:ghissue:`2951`).
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram.
* Add config variable for default doctest flags (:ghissue:`2471`).
* Convert linkcheck builder to requests for better encoding
  handling
* Add keywords of "meta" directive to search index
  (:ghissue:`2463`, :ghissue:`2516`)
```

source/changes.rst

```rst
Fixes
=====

* Add ``--implicit-namespaces`` PEP-0420 support to apidoc
  (:ghissue:`2951`).
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram.
* Add config variable for default doctest flags (:ghissue:`2471`).
* Convert linkcheck builder to requests for better encoding
  handling
* Add keywords of "meta" directive to search index
  (:ghissue:`2463`, :ghissue:`2516`)
```

source/changes.rst

```python
from docutils import nodes

BASE_URL = 'https://github.com/sphinx-doc/sphinx/issues/{}'

def github_issue(name, rawtext, text, lineno, inliner, options={},
                 content=[]):
  refuri = BASE_URL.format(text)
  node = nodes.reference(rawtext, text, refuri=refuri, **options)
  return [node], []

def setup(app):
  app.add_role('ghissue', github_issue)
  return {'version': '1.0', 'parallel_read_safe': True}
```

ext/issue_role.py

```python
from docutils import nodes

BASE_URL = 'https://github.com/sphinx-doc/sphinx/issues/{}'

def github_issue(name, rawtext, text, lineno, inliner, options={},
                 content=[]):
  refuri = BASE_URL.format(text)
  node = nodes.reference(rawtext, text, refuri=refuri, **options)
  return [node], []

def setup(app):
  app.add_role('ghissue', github_issue)
  return {'version': '1.0', 'parallel_read_safe': True}
```

ext/issue_role.py

```rst
Fixes
=====

* Add ``--implicit-namespaces`` PEP-0420 support to apidoc
  (:ghissue:`2951`)
* Add ``:caption:`` option for sphinx.ext.inheritance_diagram
* Add config variable for default doctest flags (:ghissue:`2471`)
* Convert linkcheck builder to requests for better encoding
  handling
* Add keywords of "meta" directive to search index
  (:ghissue:`2463`, :ghissue:`2516`)
```

source/changes.rst

Fixes

- Add `--implicit-namespaces` PEP-0420 support to apidoc (2951)
- Add `:caption:` option for sphinx.ext.inheritance_diagram
- Add config variable for default doctest flags (2471)
- Convert linkcheck builder to requests for better encoding handling
- Add keywords of “meta” directive to search index (2463, 2516)

Directives

4

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!*. The
only limit is your imagination/interest.
```

```python
from docutils import nodes
from docutils.parsers.rst import Directive

class XYZDirective(Directive):
  def run(self):
    section = nodes.section(ids=['test'])
    section += nodes.title(text='Test')
    section += nodes.paragraph(text='Hello, world!')
    return [section]

def setup(app):
  app.add_directive('xyz-directive', XYZDirective)
  return {'version': '1.0', 'parallel_read_safe': True}
```

```rst
Issues
======

Add keywords of "meta" directive to search index (#2463)
--------------------------------------------------------

Opened by TimKam ::

    It would be great to have the keywords of `meta` directives
    included in the search index.

    Like this, one can help users who are searching for a synonym
    of the "correct" term through simply adding synonyms as
    keywords to a meta directive on the corresponding page.
```

source/issues.rst

```rst
Issues
======

Add keywords of "meta" directive to search index (#2463)
--------------------------------------------------------

Opened by TimKam ::

    It would be great to have the keywords of `meta` directives
    included in the search index.

    Like this, one can help users who are searching for a synonym
    of the "correct" term through simply adding synonyms as
    keywords to a meta directive on the corresponding page.
```

source/issues.rst

```rst
Issues
======

.. github-issue:: 2463
```

source/issues.rst

```python
from docutils import nodes
from docutils.parsers.rst import Directive
import requests

URL = 'https://api.github.com/repos/sphinx-doc/sphinx/issues/{}'

def get_issue(issue_id):
  issue = requests.get(URL.format(issue_id)).json()
  title = '%s (#%s)' % (issue['title'], issue_id)
  owner = 'Opened by %s' issue['user']['login']

  return issue_id, title, issue['body'], owner

...
```

ext/issue_directive.py

```python
...

class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    section = nodes.section(ids=['github-issue-%s' % issue[0]])
    section += nodes.title(text=issue[1])
    section += nodes.paragraph(text='Opened by %s' % issue[3])
    section += nodes.literal_block(text=issue[2])

    return [section]
```

ext/issue_directive.py

```python
...

class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    section = nodes.section(ids=['github-issue-%s' % issue[0]])
    section += nodes.title(text=issue[1])
    section += nodes.paragraph(text='Opened by %s' % issue[3])
    section += nodes.literal_block(text=issue[2])

    return [section]
```

ext/issue_directive.py

```python
...

class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    section = nodes.section(ids=['github-issue-%s' % issue[0]])
    section += nodes.title(text=issue[1])
    section += nodes.paragraph(text='Opened by %s' % issue[3])
    section += nodes.literal_block(text=issue[2])

    return [section]
```

ext/issue_directive.py

```python
...

def setup(app):
  app.add_directive('github-issue', ShowGitHubIssue)
  return {'version': '1.0', 'parallel_read_safe': True}
```

ext/issue_directive.py

```
Issues
======

.. github-issue:: 2463
```

source/issues.rst

Issues

Add keywords of “meta” directive to search index (#2463)

Opened by TimKam

```
It would be great to have the keywords of `meta` directives
included in the search index.

Like this, one can help users who are searching for a
synonym of the "correct" term through simply adding
synonyms as keywords to a `meta` directive on the
corresponding page.
```

```python
class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    section = nodes.section(ids=['github-issue-%s' % issue[0]])
    section += nodes.title(text=issue[1])
    section += nodes.paragraph(text='Opened by %s' % issue[3])
    section += nodes.literal_block(text=issue[2])

    return [section]
```

ext/issue_directive.py

```python
class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    section = nodes.section(ids=['github-issue-%s' % issue[0]])
    section += nodes.title(text=issue[1])
    section += nodes.paragraph(text='Opened by %s' % issue[3])
    section += nodes.literal_block(text=issue[2])

    return [section]
```

ext/issue_directive.py

```python
class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    result = statemachine.ViewList()
    for line in format_issue(issue):
      result.append(line, '<' + __name__ + '>')
      node = nodes.section(document=self.state.document)
      nested_parse_with_titles(self.state, result, node)

    return node.children
```

ext/issue_directive.py

```python
class ShowGitHubIssue(Directive):
  required_arguments = 1

  def run(self):
    issue = get_issue(self.arguments[0])

    result = statemachine.ViewList()
    for line in format_issue(issue):
      result.append(line, '<' + __name__ + '>')
      node = nodes.section(document=self.state.document)
      nested_parse_with_titles(self.state, result, node)

    return node.children
```

ext/issue_directive.py

```python
def format_issue(issue):
  num, title, body, owner = issue

  yield title
  yield '=' * len(title)
  yield ''
  yield '%s ::' % owner
  yield ''
  for line in body.splitlines():
    yield '    %s' % line if line else ''
```

ext/issue_directive.py

Issues

Add keywords of “meta” directive to search index (#2463)

Opened by TimKam

```
It would be great to have the keywords of `meta` directives
included in the search index.

Like this, one can help users who are searching for a
synonym of the "correct" term through simply adding
synonyms as keywords to a `meta` directive on the
corresponding page.
```

Events

5

builder-inited(app)

config-inited(app, config)

source-read(app, docname, source)

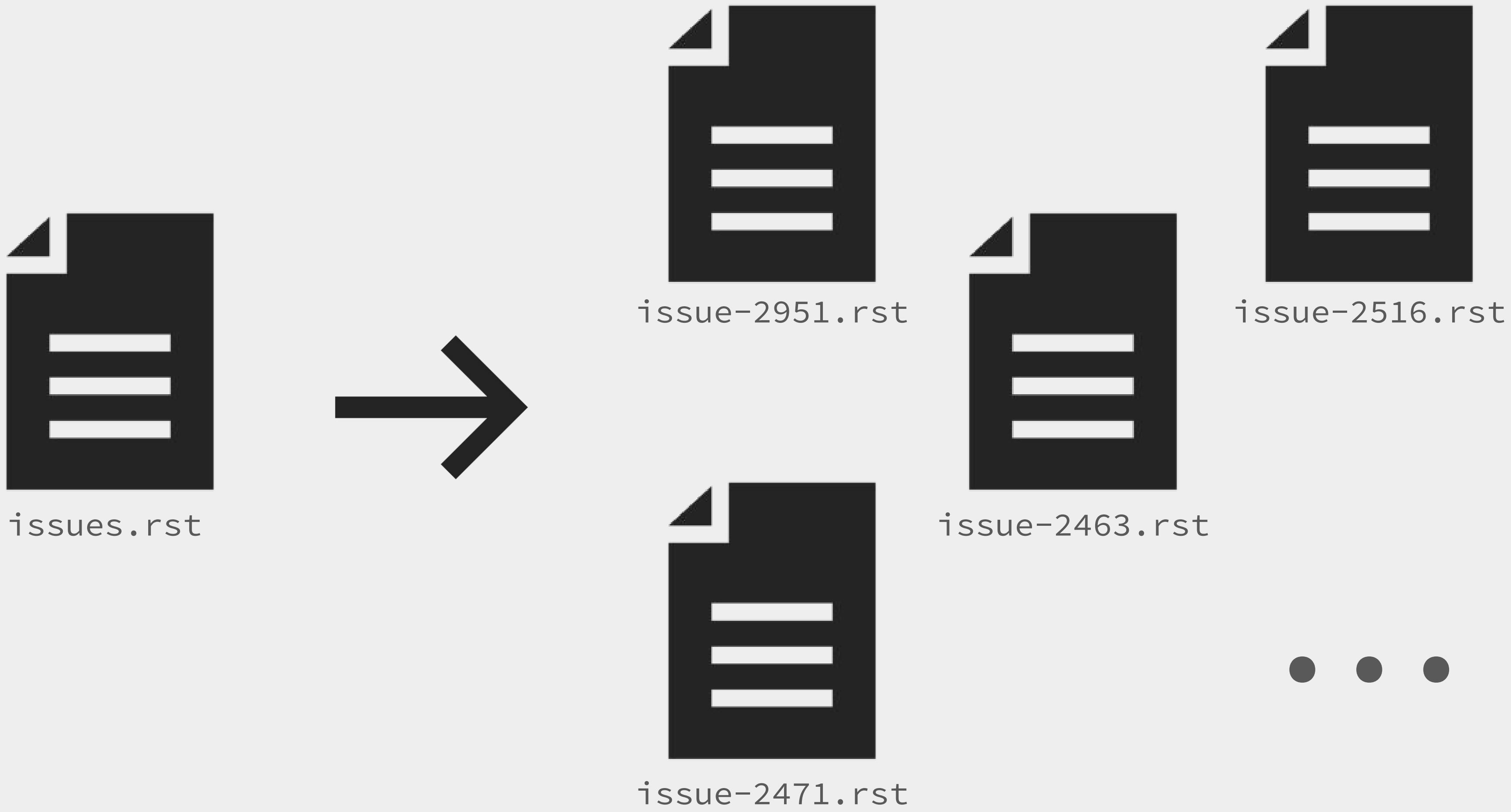
doctree-read(app, doctree)

...

```python
from docutils import nodes
from docutils.parsers.rst import Directive


def builder_inited_handler(app):
    # code here...


def setup(app):
    app.connect('builder-inited', builder_inited_handler)
```


issues.rst
issue-2951.rst
issue-2516.rst
issue-2463.rst
issue-2471.rst

```python
from docutils import nodes
from docutils.parsers.rst import Directive
import requests

URL = 'https://api.github.com/repos/sphinx-doc/sphinx/issues/'

def get_issues():
  issues = requests.get(URL).json()

  for issue in issues:
    title = '%s (#%s)' % (issue['title'], issue['number'])
    owner = 'Opened by %s' % issue['user']['login']
    yield issue['number'], title, issue['body'], owner
```

ext/issue_event.py

```python
...

def generate_issue_docs(app):
  for num, title, body, owner in get_issues():
    filename = os.path.join(app.srcdir, 'issues', '%s.rst' % num)

    with io.open(filename, 'w') as issue_doc:
      print(title, file=issue_doc)
      print('=' * len(title), file=issue_doc)
      print('', file=issue_doc)
      print('%s ::' % owner, file=issue_doc)
      print('', file=issue_doc)
      for line in body.splitlines():
        print('    %s' % line if line else '', file=issue_doc)
```

```python
...

def setup(app):
  app.connect('builder-inited', generate_issue_docs)
  return {'version': '1.0', 'parallel_read_safe': True}
```

```
Issues
======

.. toctree::
   :maxdepth: 1
   :glob:

   issues/*
```

source/index.rst

Issues

- Drop special support for rst2pdf (#4463)
- Proposal: Integrate source_suffix and source_parsers (#4474)
- [RFC] Implement delayed resolution in TOC (#4475)
- Not possible to update individual ‘po’ files (#4476)
- Build fails during eclim (aur) build: Babel data files not available (#4481)
- Proposal: Allow to switch parsers on parsing document (#4482)
- Integrate source suffix and source parsers (#4483)
- …

Enabling Your Extensions

5

```python
import os
import sys

sys.path.insert(0, os.path.abspath('../ext'))

extensions = [
  'issue_role',
  'issue_directive',
  'issue_event',
]
```

source/conf.py

```python
import os
import sys

sys.path.insert(0, os.path.abspath('../ext'))

extensions = [
  'issue_role',
  'issue_directive',
  'issue_event',
  'oslo_config.sphinxext',
]
```

source/conf.py

Wrap Up

6

Extensions are registered via **`sphinx.application.Application`**

`add_builder`	(Builders)
`add_config_value`	(Config Values)
`add_domain`	(Domains)
`add_event`	(Events)
`add_node`	(docutils Nodes)
`add_directive`	(Directives)
`add_role`	(Interpreted Text Roles, a.k.a. Roles)
`connect, disconnect`	(Hooks)
`...`	...

Extensions are registered via **`sphinx.application.Application`**

`add_builder`	(Builders)
`add_config_value`	(Config Values)
`add_domain`	(Domains)
`add_event`	(Events)
`add_node`	(docutils Nodes)
`add_directive`	(Directives)
`add_role`	(Interpreted Text Roles, a.k.a. Roles)
`connect, disconnect`	(Hooks)
...	...

Extensions are registered via **`sphinx.application.Application`**

Builder-specific Extensions (HTML themes, LaTeX templates, …)

(Post) Transforms

Translators

Parsers

Search languages

...

Fin

A lion, a head, and a dash of YAML

Extending Sphinx to automate your documentation

FOSDEM 2018

@stephenfin

References

- Quick reStructuredText
- Docutils Reference Guide
  - reStructuredText Markup Specification
  - reStructuredText Directives
  - reStructuredText Interpreted Text Roles
- Docutils How-Tos
  - Creating reStructuredText Interpreted Text Roles
  - Creating reStructuredText Directives
- Docutils Hacker’s Guide
- Sphinx Tutorial: Writing a simple extension

References

- Defining Custom Roles in Sphinx -- Doug Hellmann
- The Power of Sphinx - Integrating Jinja with RST -- Eric Holscher
- Docutils Snippets -- Aurélien Gâteau
- OpenStack + Sphinx In A Tree -- Stephen Finucane (💥)