

About Sphinx

Eunchong YU
kroisse@gmail.com

유은총

<http://krois.se/>

<https://github.com/Kroisse>

Developing with Python for food

Univ. of Seoul (2007~)

StyleShare (2011)

SMARTSTUDY (2012~)

소스 코드 문서화

소프트웨어와 소스 코드에 관한 내용들을 기록
으로 남기는 일

- 튜토리얼
- API 레퍼런스
- 체인지로그
- 기타 필요한 글들...

문서화가 필요할 때

- 여러 명이 협업하는 프로젝트
 - 팀원 간에 코드 명세, 인터페이스 등을 공유
- 오픈소스 소프트웨어 공개
 - 모름지기 문서 잘 쓴 오픈소스가 잘 팔립니다.
- 미래의 나에게 보내는 메시지
 - “.....내가 이걸 왜 이렇게 짰더라?”

문서화 도구...?

-워드?
-위키?

더 나은 문서화 도구

- C/C++: doxygen
- Java: Javadoc
- Python: Sphinx

Python 커뮤니티의 de facto standard



SPHINX

Python Documentation Generator

<http://sphinx-doc.org/>

A  project

- [Flask](#), [Jinja](#), [Pygments](#),
- <http://www.pocoo.org/>

Sphinx로 작성된 문서들

<http://docs.python.org/>

<http://flask.pocoo.org/docs/>

<http://docs.sqlalchemy.org/>

<http://docs.djangoproject.org/>

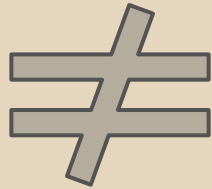
.....등등 수많은 소프트웨어 문서들을 Sphinx를
사용해서 작성하고 있습니다.

주의사항

검색엔진 [Sphinx](#)와 헷갈리지 마세요



Sphinx



SPHINX

Python Documentation Generator

How to use

Quickstart

<http://sphinx-doc.org/tutorial.html>

```
$ pip install sphinx
```

```
$ sphinx-quickstart
```

그리고 시키는 대로 하시면 됩니다.

Quickstart (계속)

```
2. fish /Users/ecdysis/workspace/fruitseller/docs (fish)
~/G/P/fruitseller sphinx-quickstart
Welcome to the Sphinx 1.1.3 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 [.]: docs

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: FruitSeller
> Author name(s): Eunhong Yu

Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1. If you don't need this dual structure,
just set both to the same value.
> Project version: 0.1
> Project release [0.1]: 0.1.0

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst". Only files with this suffix are considered documents.
> Source file suffix [.rst]:
```

Quickstart (계속)

2. fish /Users/ecdysis/workspace/fruitseller/docs (fish)

One document is special in that it is considered the top node of the "contents tree", that is, it is the root of the hierarchical structure of the documents. Normally, this is "index", but if your "index" document is a custom template, you can also set this to another filename.

> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:

> Do you want to use the epub builder (y/N) [n]:

Please indicate if you want to use one of the following Sphinx extensions:

> autodoc: automatically insert docstrings from modules (y/N) [n]: y

> doctest: automatically test code snippets in doctest blocks (y/N) [n]:

> intersphinx: link between Sphinx documentation of different projects (y/N) [n]: y

> todo: write "todo" entries that can be shown or hidden on build (y/N) [n]:

> coverage: checks for documentation coverage (y/N) [n]:

> pngmath: include math, rendered as PNG images (y/N) [n]:

> mathjax: include math, rendered in the browser by MathJax (y/N) [n]:

> ifconfig: conditional inclusion of content based on config values (y/N) [n]:

> viewcode: include links to the source code of documented Python objects (y/N) [n]:

A Makefile and a Windows command file can be generated for you so that you only have to run e.g. 'make html' instead of invoking sphinx-build directly.

> Create Makefile? (Y/n) [y]:

> Create Windows command file? (Y/n) [y]:

Creating file docs/conf.py.

Creating file docs/index.rst.

Creating file docs/Makefile.

Creating file docs/make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file docs/index.rst and create other documentation source files. Use the Makefile to build the docs, like so:

make builder

where "builder" is one of the supported builders, e.g. html, latex or linkcheck.

일단 문서가 뜨는 걸 구경해봅시다

```
2. fish /Users/ecdysis/workspace/fruitseller/docs (fish)
~/w/fruitseller ➤ cd docs
~/w/f/docs ➤ make html
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.1.3
loading pickled environment... not yet created
loading intersphinx inventory from http://docs.python.org/objects.inv...
building [html]: targets for 1 source files that are out of date
updating environment: 1 added, 0 changed, 0 removed
reading sources... [100%] index

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index

writing additional files... genindex search
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.

Build finished. The HTML pages are in _build/html.
~/w/f/docs ➤ open _build/html/index.html
~/w/f/docs ➤
```

일단 문서가 뜨는 걸 구경해봅시다

Welcome to FruitSeller's documentation! — FruitSeller 0.1.0 documentation

FruitSeller 0.1.0 documentation » index

Table Of Contents

Welcome to FruitSeller's documentation!
Indices and tables

This Page

Show Source

Quick search

Go

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

Welcome to FruitSeller's documentation!

Contents:

Indices and tables

- [Index](#)
- [Module Index](#)
- [Search Page](#)

FruitSeller 0.1.0 documentation » index

© Copyright 2013, Euncheon Yu. Created using [Sphinx](#) 1.1.3.

디렉토리 구조

`_build/`

빌드된 문서가 저장되는 곳

`_static/`

문서에 쓸 이미지 등을 넣는 곳

`_templates/`

커스텀 템플릿을 넣는 곳

`conf.py`

설정 파일

`index.rst`

시작 문서

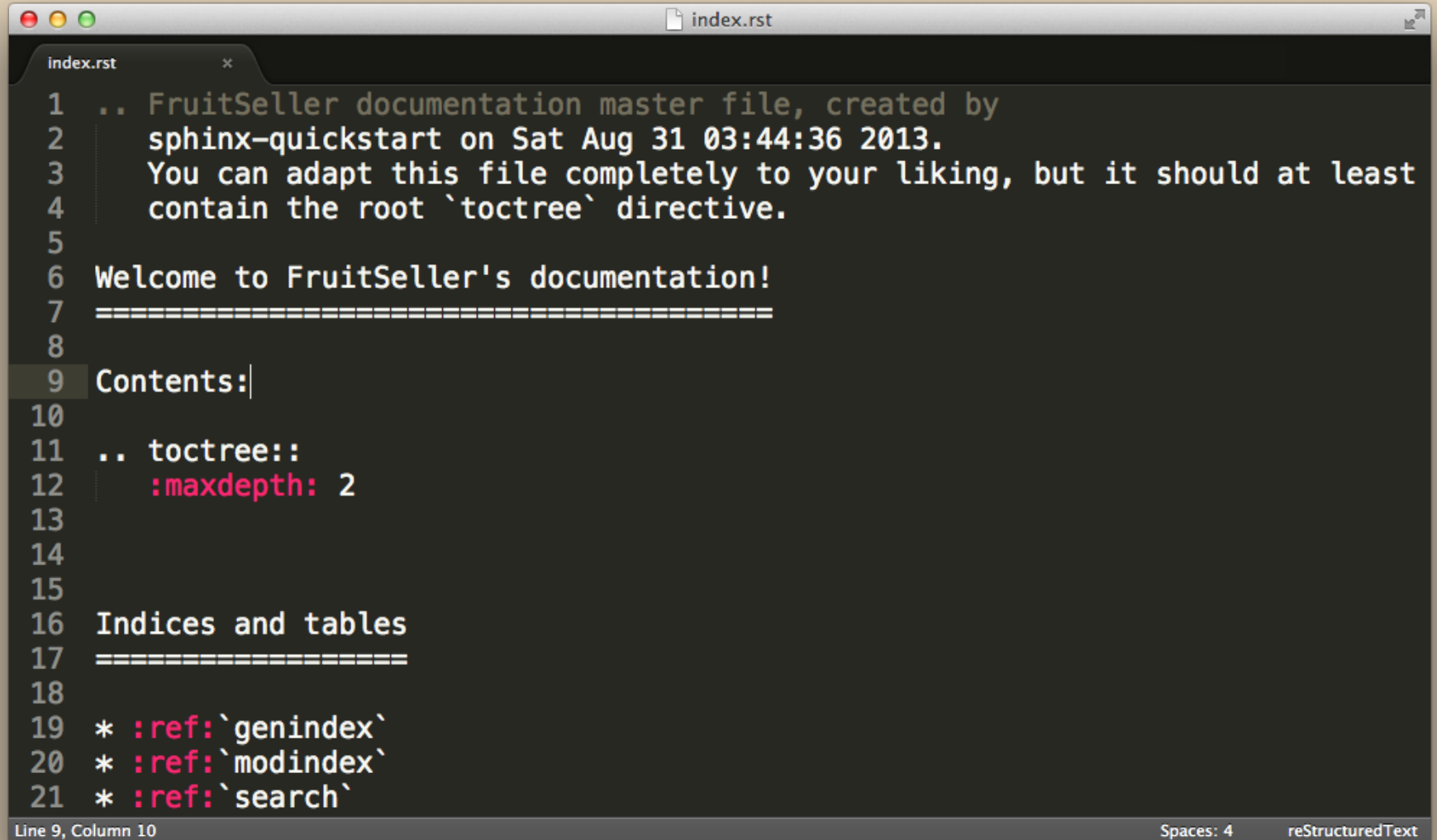
`Makefile`

`make html`이 가능하게 해 주는 파일

`make.bat`

윈도에서 `make html`이 되게 해 주는 파일

index.rst

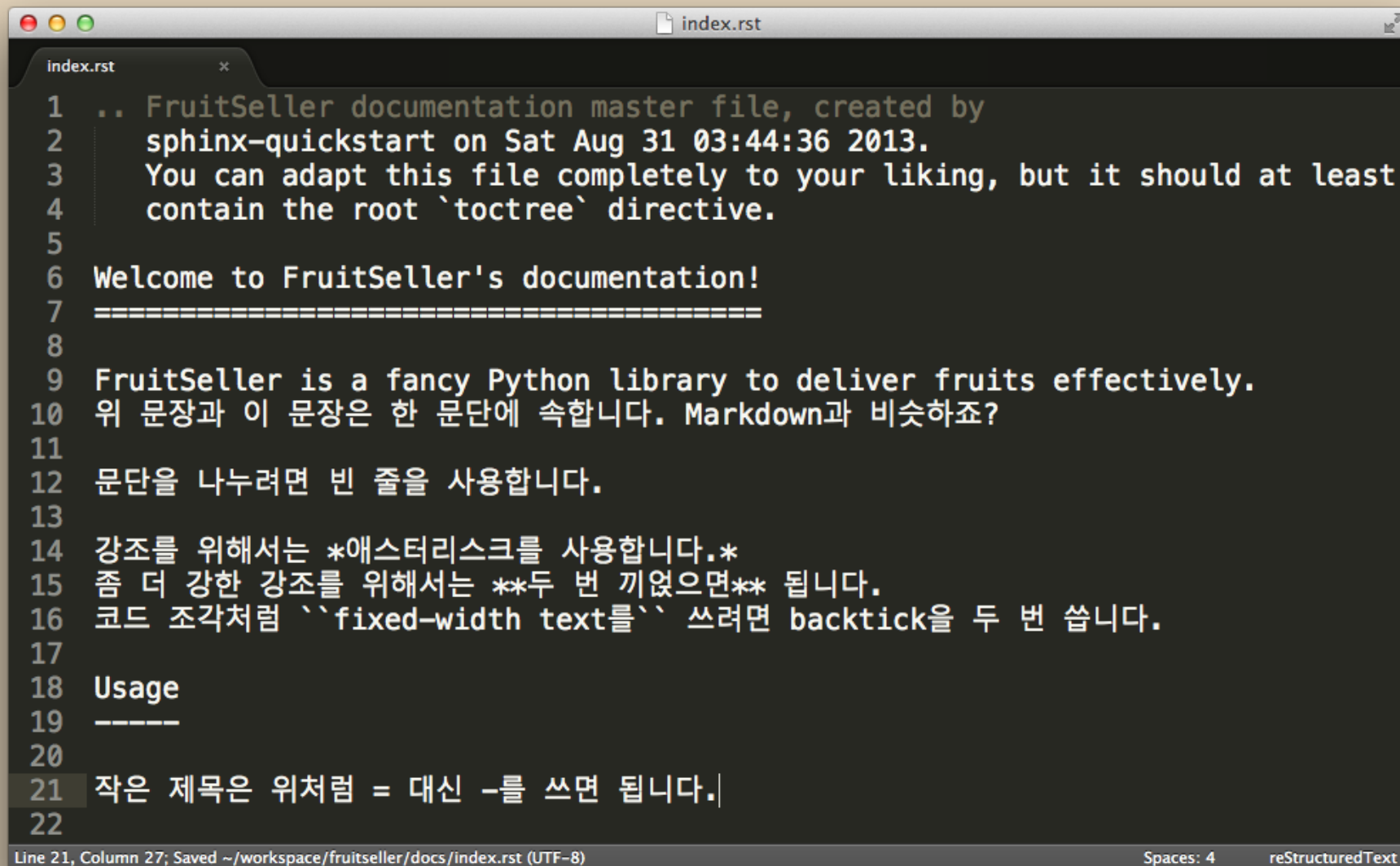


```
1  .. FruitSeller documentation master file, created by
2     sphinx-quickstart on Sat Aug 31 03:44:36 2013.
3     You can adapt this file completely to your liking, but it should at least
4     contain the root `toctree` directive.
5
6  Welcome to FruitSeller's documentation!
7  =====
8
9  Contents:|
10
11  .. toctree::
12     :maxdepth: 2
13
14
15
16  Indices and tables
17  =====
18
19  * :ref:`genindex`
20  * :ref:`modindex`
21  * :ref:`search`
```

Line 9, Column 10

Spaces: 4 reStructuredText

reStructuredText 맛보기



```
index.rst
1  .. FruitSeller documentation master file, created by
2     sphinx-quickstart on Sat Aug 31 03:44:36 2013.
3     You can adapt this file completely to your liking, but it should at least
4     contain the root `toctree` directive.
5
6  Welcome to FruitSeller's documentation!
7  =====
8
9  FruitSeller is a fancy Python library to deliver fruits effectively.
10 위 문장과 이 문장은 한 문단에 속합니다. Markdown과 비슷하죠?
11
12 문단을 나누려면 빈 줄을 사용합니다.
13
14 강조를 위해서는 *애스터리스크를 사용합니다.*
15 좀 더 강한 강조를 위해서는 **두 번 끼었으면** 됩니다.
16 코드 조각처럼 ``fixed-width text``를 쓰려면 backtick을 두 번 씁니다.
17
18 Usage
19 -----
20
21 작은 제목은 위처럼 = 대신 -를 쓰면 됩니다.
22
```

Line 21, Column 27; Saved ~/workspace/fruitseller/docs/index.rst (UTF-8) Spaces: 4 reStructuredText

reStructuredText 맛보기 (계속)

Browser address bar: Welcome to FruitSeller's documentation! — FruitSeller 0.1.0 documentation

Page title: FruitSeller 0.1.0 documentation » [index](#)

Table Of Contents

- Welcome to FruitSeller's documentation!
 - Usage
 - Indices and tables

This Page

[Show Source](#)

Quick search

[Go](#)

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

Welcome to FruitSeller's documentation!

FruitSeller is a fancy Python library to deliver fruits effectively. 위 문장과 이 문장은 한 문단에 속합니다. Markdown과 비슷하죠?

문단을 나누려면 빈 줄을 사용합니다.

강조를 위해서는 *애스터리스크를 사용합니다*. 좀 더 강한 강조를 위해서는 두 번 **끼었으면 됩니다**. 코드 조각처럼 `fixed-width text`를 쓰려면 backtick을 두 번 씁니다.

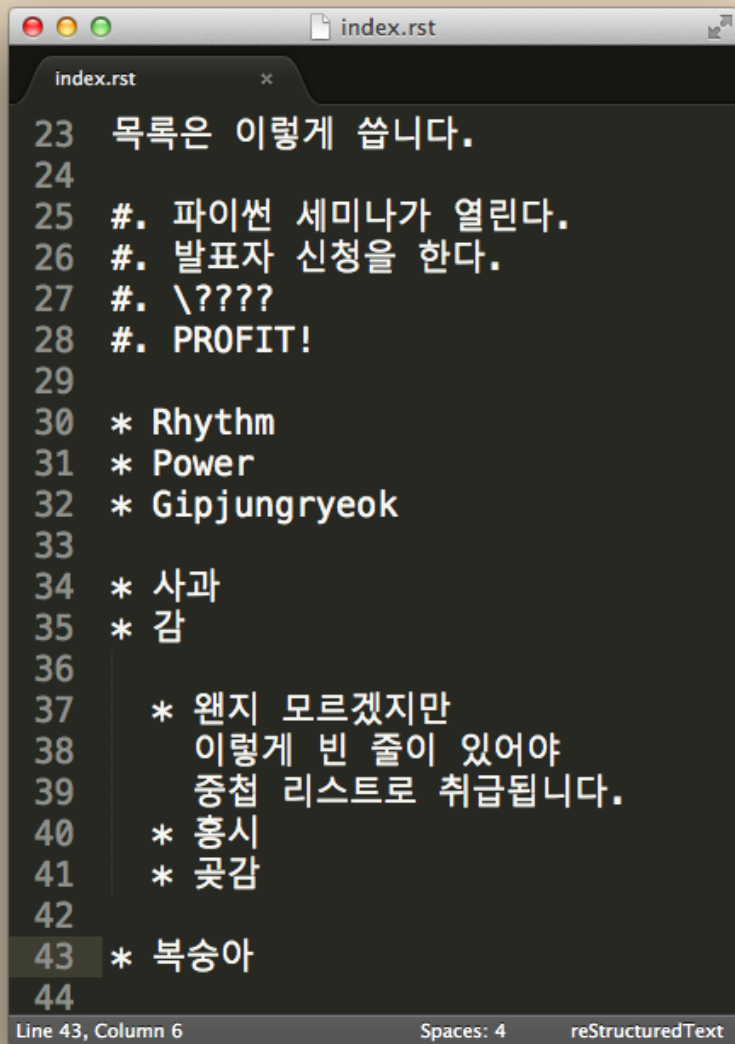
Usage

작은 제목은 위처럼 = 대신 -를 쓰면 됩니다.

Indices and tables

- [Index](#)
- [Module Index](#)
- [Search Page](#)

reStructuredText 맛보기 (계속)



```
index.rst
23  목록은 이렇게 씁니다.
24
25  #. 파이썬 세미나가 열린다.
26  #. 발표자 신청을 한다.
27  #. \????
28  #. PROFIT!
29
30  * Rhythm
31  * Power
32  * Gipjungryeok
33
34  * 사과
35  * 감
36
37      * 웬지 모르겠지만
38      이렇게 빈 줄이 있어야
39      중첩 리스트로 취급됩니다.
40  * 홍시
41  * 귤감
42
43  * 복숭아
44
```

Line 43, Column 6 Spaces: 4 reStructuredText

목록은 이렇게 씁니다.

1. 파이썬 세미나가 열린다.
 2. 발표자 신청을 한다.
 3. \????
 4. PROFIT!
- Rhythm
 - Power
 - Gipjungryeok
 - 사과
 - 감
 - 웬지 모르겠지만 이렇게 빈 줄이 있어야 중첩 리스트로 취급됩니다.
 - 홍시
 - 귤감
 - 복숭아

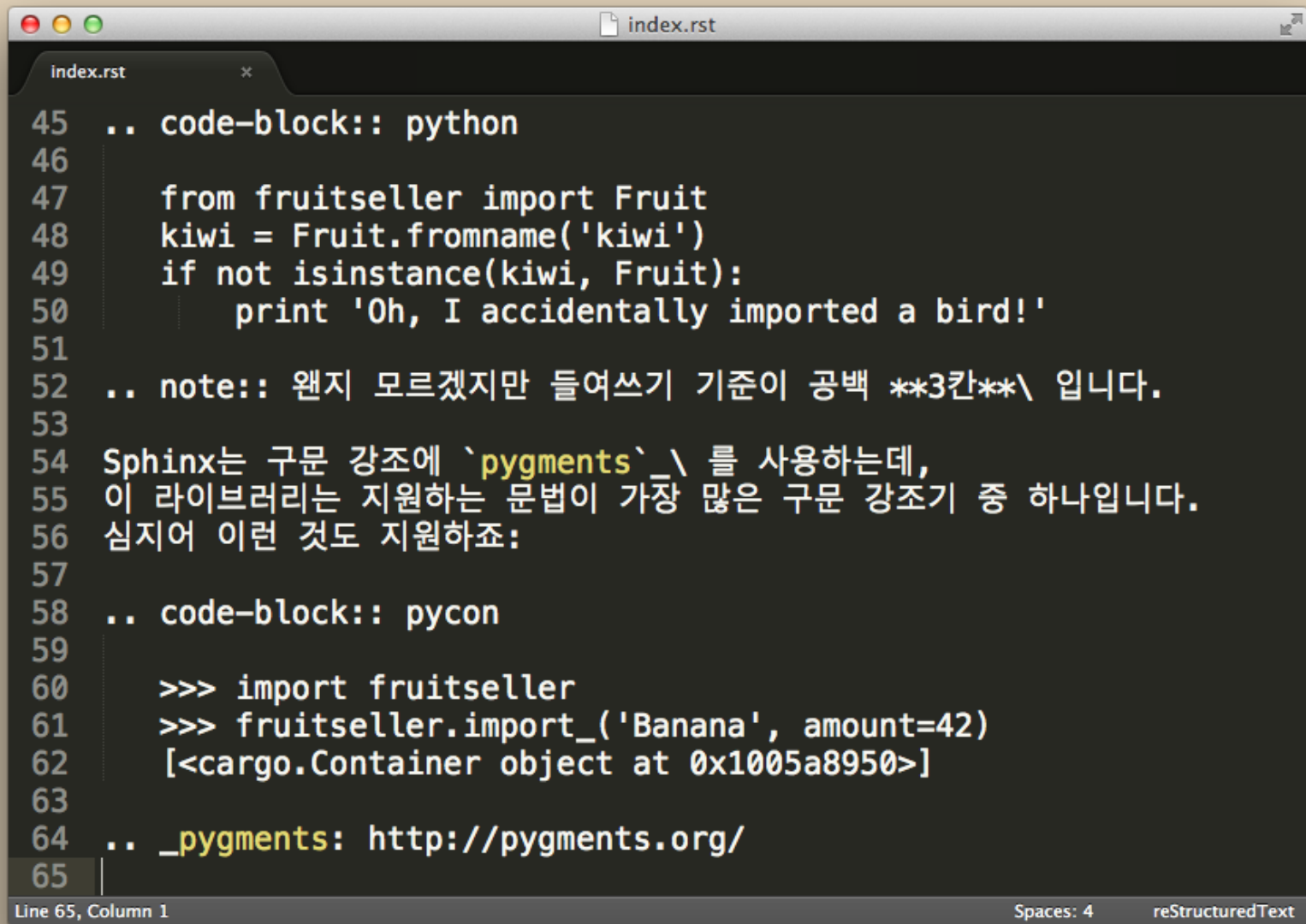
함정 카드

- **강조**를 (X)
- **강조**\ 를 (O)

마크업은 항상 공백으로 구분되어야 합니다.

덕분에 한글로 문서 쓰기가 좀 귀찮습니다.

Directives



A screenshot of a text editor window titled 'index.rst'. The editor shows a file with line numbers 45 to 65. The content includes Sphinx directives for code blocks, a note, and a link. The code blocks are in Python and Pycon syntax. The note is in Korean. The link is to the Pygments website.

```
45 .. code-block:: python
46
47     from fruitseller import Fruit
48     kiwi = Fruit.fromname('kiwi')
49     if not isinstance(kiwi, Fruit):
50         print 'Oh, I accidentally imported a bird!'
51
52 .. note:: 왠지 모르겠지만 들여쓰기 기준이 공백 **3칸**\ 입니다.
53
54 Sphinx는 구문 강조에 `pygments` \ 를 사용하는데,
55 이 라이브러리는 지원하는 문법이 가장 많은 구문 강조기 중 하나입니다.
56 심지어 이런 것도 지원하죠:
57
58 .. code-block:: pycon
59
60     >>> import fruitseller
61     >>> fruitseller.import_('Banana', amount=42)
62     [<cargo.Container object at 0x1005a8950>]
63
64 .. _pygments: http://pygments.org/
65
```

Line 65, Column 1

Spaces: 4 reStructuredText

Directives (계속)

```
from fruitseller import Fruit
kiwi = Fruit.fromname('kiwi')
if not isinstance(kiwi, Fruit):
    print 'Oh, I accidentally imported a bird!'
```

Note: 왠지 모르겠지만 들여쓰기 기준이 공백 3칸입니다.

Sphinx는 구문 강조에 `pygments`를 사용하는데, 이 라이브러리는 지원하는 문법이 가장 많은 구문 강조기 중 하나입니다. 심지어 이런 것도 지원하죠:

```
>>> import fruitseller
>>> fruitseller.import_('Banana', amount=42)
[<cargo.Container object at 0x1005a8950>]
```

레퍼런스는 개발자의 친구입니다 :)

- <http://sphinx-doc.org/rest.html>
- <http://docutils.sourceforge.net/rst.html>

커다란 구조를 만듭니다

```
2. fish /Users/ecdysis/workspace/fruitseller/docs (fish)
~/w/f/docs ➤ cat > install.rst
Installation
=====

.. code-block:: bash

    $ pip install fruitseller
~/w/f/docs ➤ mkdir api/
~/w/f/docs ➤ cat > api/fruits.rst
과 일 API
=====
~/w/f/docs ➤
```

```
index.rst
index.rst
66 목차
67 ----
68
69 .. toctree::
70     :maxdepth: 2
71
72     install
73     api/fruits
74
Line 74, Column 1 Spaces: 4 reStructuredText
```

커다란 구조를 만듭니다

목차

- Installation
- 과일 API

FruitSeller 0.1.0 documentation >>

Previous topic

Welcome to FruitSeller's documentation!

Next topic

과일 API

Installation

```
$ pip install fruitseller
```

FruitSeller 0.1.0 documentation >>

Previous topic

Installation

This Page

과일 API

커다란 구조를 만듭니다

```
index.rst
66 목차
67 ----
68
69 .. toctree::
70     :maxdepth: 2
71
72     install
73     reference
74     web/routes
75

reference.rst
1 레퍼런스
2 =====
3
4 과일 객체
5 -----
6
7 유틸리티 함수
8 -----
9
```

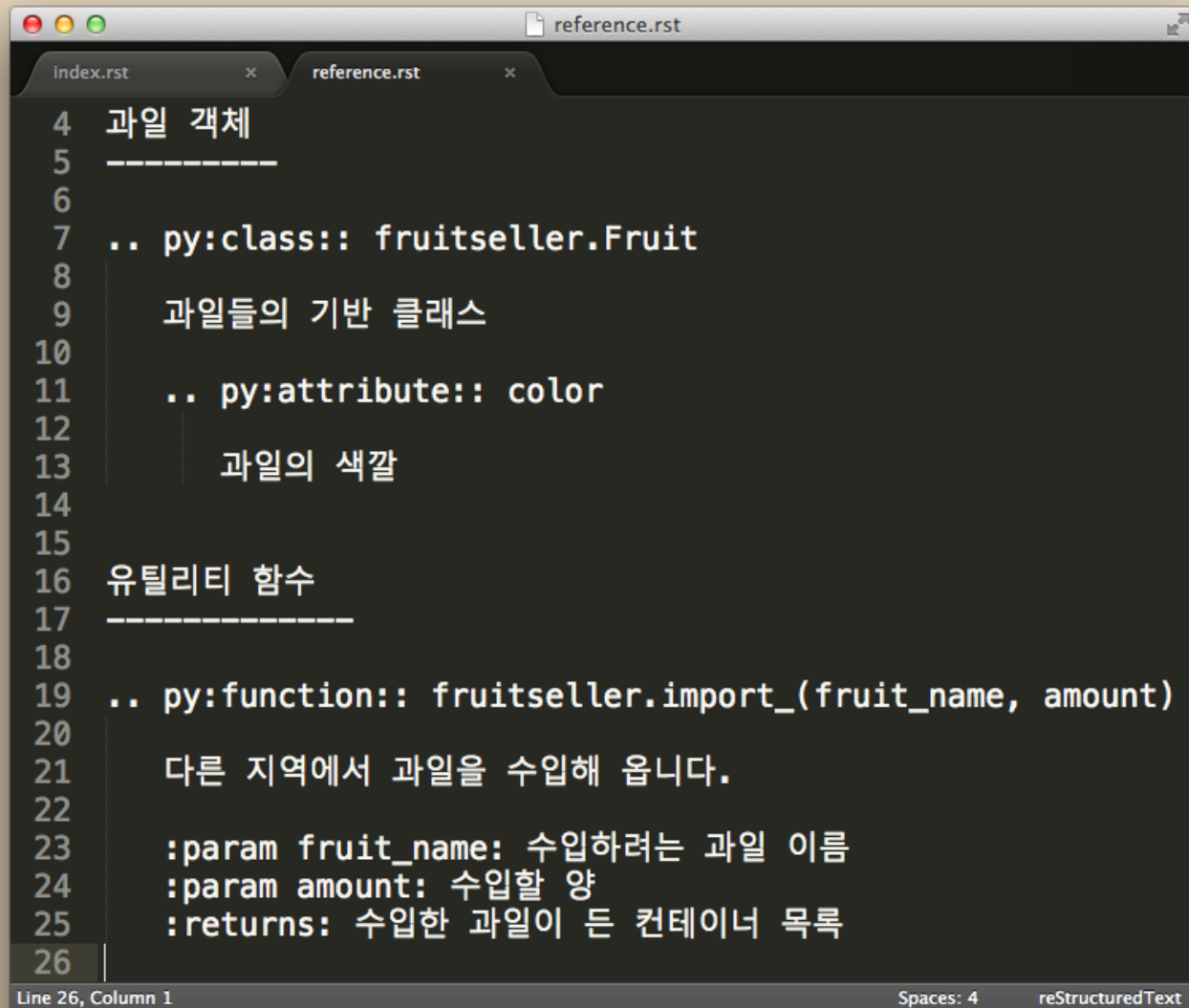
Line 72, Column 11 Spaces: 2 reStru

목차 ¶

- 설치하기
- 레퍼런스
 - 과일 객체
 - 유틸리티 함수
- 과일 API

Features

레퍼런스를 작성해 봅시다



```
4 과일 객체
5 -----
6
7 .. py:class:: fruitseller.Fruit
8
9     과일들의 기반 클래스
10
11     .. py:attribute:: color
12
13         과일의 색깔
14
15
16 유틸리티 함수
17 -----
18
19 .. py:function:: fruitseller.import_(fruit_name, amount)
20
21     다른 지역에서 과일을 수입해 옵니다.
22
23     :param fruit_name: 수입하려는 과일 이름
24     :param amount: 수입할 양
25     :returns: 수입한 과일이 든 컨테이너 목록
26
```

Line 26, Column 1

Spaces: 4 reStructuredText

레퍼런스를 작성해 봅시다

과일 객체

```
class fruitseller.Fruit ¶
```

과일들의 기반 클래스

color

과일의 색깔

유틸리티 함수

```
fruitseller.import_(fruit_name, amount)
```

다른 지역에서 과일을 수입해 옵니다.

Parameters:

- **fruit_name** - 수입하려는 과일 이름
- **amount** - 수입할 양

Returns: 수입한 과일이 든 컨테이너 목록

링크하기



The screenshot shows a text editor window with two tabs: 'index.rst' and 'reference.rst'. The 'index.rst' tab is active. The code in the editor is as follows:

```
66 다른 위치에서 :class:`fruit seller.Fruit` 를 링크할 수 있습니다.  
67  
68 * :class:`과일 <fruit seller.Fruit>`  
69 * :func:`~fruit seller.import_`  
70
```

The status bar at the bottom indicates 'Line 69, Column 31', 'Spaces: 2', and 'reStructuredText'.

다른 위치에서 **fruit seller.Fruit**를 링크할 수 있습니다.

- 과일
- **import_()**

Cross-reference

<http://sphinx-doc.org/markup/inline.html#cross-referencing-syntax>

다른 문서에 있는 함수, 클래스, 메서드, 변수, 모듈, 섹션 등을 링크할 수 있습니다.

:func: `add_fruit`

:class: `Banana`

:meth: `Banana.eat`

:data: `fruitseller.currency`

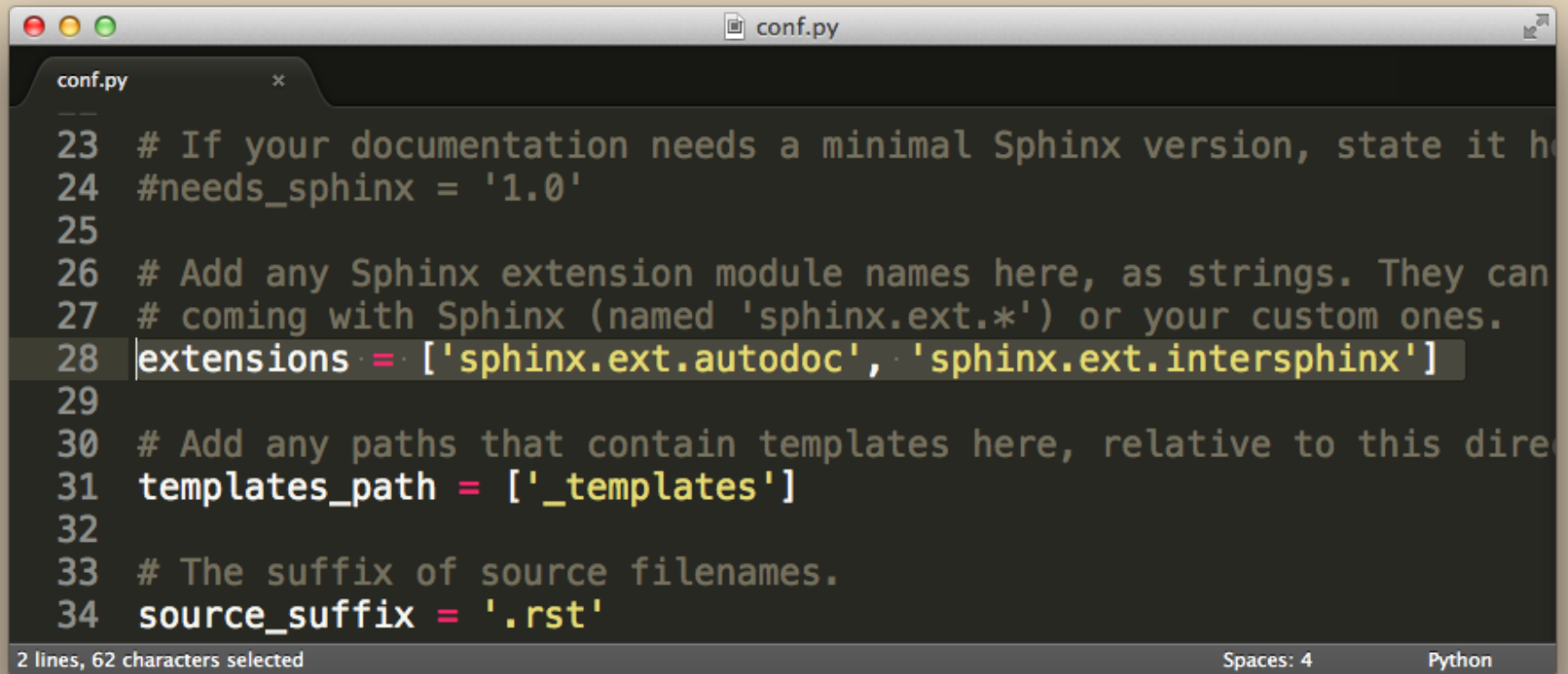
:mod: `fruitseller`

:ref: `how-to-install`

이제 레퍼런스 문서를 쓸 수 있긴 한데...

- 귀찮습니다.
 - 어차피 소스코드에 그 함수 있는데 또 써야 돼?
 - Javadoc처럼 함수 근처에 바로 쓸 수 있다면...

conf.py를 열어봅니다



```
conf.py
23 # If your documentation needs a minimal Sphinx version, state it here
24 #needs_sphinx = '1.0'
25
26 # Add any Sphinx extension module names here, as strings. They can be
27 # coming with Sphinx (named 'sphinx.ext.*') or your custom ones.
28 extensions = ['sphinx.ext.autodoc', 'sphinx.ext.intersphinx']
29
30 # Add any paths that contain templates here, relative to this directory
31 templates_path = ['_templates']
32
33 # The suffix of source filenames.
34 source_suffix = '.rst'
```

2 lines, 62 characters selected

Spaces: 4 Python

sphinx.ext.autodoc

```
fruitseller.py
1 # -*- coding: utf-8 -*-
2
3 class Fruit(object):
4     """과일들의 기반 클래스
5
6     .. attribute:: color
7
8         과일의 색깔
9
10    """
11
12    def __init__(self, name, color):
13        self.name = name
14        self.color = color
15
16    def import_(fruit_name, amount):
17        """다른 지역에서 과일을 수입해
18
19        :param fruit_name: 수입하려는
20        :param amount: 수입할 양
21        :returns: 수입한 과일이 든 컨테이너
22
23    """
24    pass
25
```

Line 17, Column 18 Spaces: 4 Python

```
reference.rst
index.rst      reference.rst
1 레퍼런스
2 =====
3
4 과일 객체
5 -----
6
7 .. autoclass:: fruitseller.Fruit
8
9
10 유틸리티 함수
11 -----
12
13 .. autofunction:: fruitseller.import_
14
```

Line 14, Column 1 Spaces: 4 reStructuredText

다른 extensions들

- sphinx.ext.intersphinx
 - Sphinx로 작성된 사이트간에 링크가 가능
 - ex: `:obj: `basestring``
→ docs.python.org로 링크
- sphinx.ext.todo
 - todo directive를 추가

sphinx-contrib

<https://bitbucket.org/birkenfeld/sphinx-contrib/>

유용한 sphinx 확장들 모음

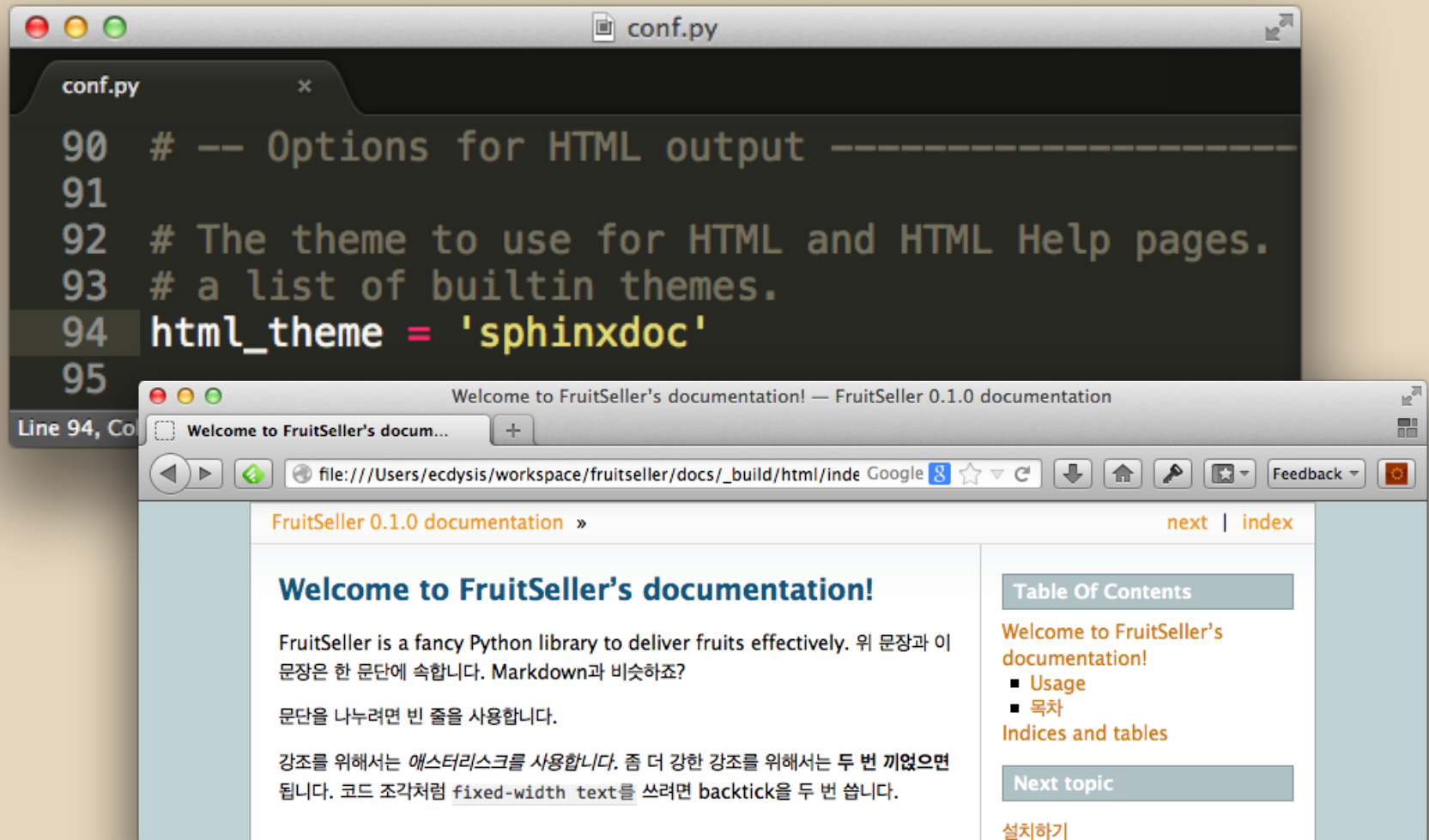
List of extensions

- aafig: render embeded ASCII art as nice images using [aafigure](#).
- actdiag: embed activity diagrams by using [actdiag](#)
- adadomain: an extension for Ada support (Sphinx 1.0 needed)
- ansi: parse ANSI color sequences inside documents
- autorun: Execute code in a runblock directive.
- blockdiag: embed block diagrams by using [blockdiag](#)
- cheeseshop: easily link to PyPI packages
- clearquest: create tables from [ClearQuest](#) queries.
- coffeedomain: a domain for (auto)documenting CoffeeScript source code.
- context: a builder for ConTeXt

● sphinxcontrib-httpdomain

- <http://pythonhosted.org/sphinxcontrib-httpdomain/>

테마를 바꿔봅시다



테마를 바꿔봅시다

<http://sphinx-doc.org/theming.html#builtin-themes>

Theme overview

[Python v2.6.4 documentation](#) > [The Python Standard Library](#) > [9. Data Types](#) > [previous](#) | [next](#) | [modules](#) | [index](#)

Table Of Contents

- 9.8. sched — Event scheduler
- 9.8.1. Scheduler Objects

Previous topic

9.7. sets — Unordered collections of unique elements

Next topic

9.9. sets — Mutual exclusion support

This Page

Show source

Quick search

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

9.8. sched — Event scheduler

The `sched` module defines a class which implements a general purpose event scheduler:

```

class sched, scheduler(object, datetime)
    The scheduler class defines a generic interface to scheduling events. It needs two functions to actually deal with the "outside world" — timefunc should be callable without arguments, and return a number (the "time", in any units whatsoever). The datetime function should be callable with one argument, compatible with the output of timefunc, and should delay that many time units. datetime will also be called with the argument 0 after each event is run to allow other threads an opportunity to run in multi-threaded applications.

    Example:

    >>> import sched, time
    >>> s = sched.scheduler(time.time, time.sleep)
    >>> def print_time(): print "from print_time", time.time()
    >>>
    >>> def print_some_times():
    ...     print time.time()
    ...     s.enter(5, 0, print_time, ())
    ...     s.enter(10, 1, print_time, ())
    ...     s.run()
    ...     print time.time()
    >>>
    >>> s.print_some_times()
    900343000.257
    from print_time 900343005.274
    from print_time 900343010.279
    900343015.276
  
```

In multi-threaded environments, the `scheduler` class has limitations with respect to thread-safety. Inability to insert a new task before the one currently pending in a running scheduler, and holding up the main thread until the event queue is empty. Instead, the preferred approach is to use the `threading.Timer` class instead.

default



Introduction | API | Basics | Table Of Contents

API

This document describes the API to Jinja2 and not the template language. It will be more useful as reference to those implementing the template interface to the application and not those who are creating Jinja2 templates.

Basics

Jinja2 uses a central object called the `template Environment`. Instances of this class are used to store the configuration, global objects and are used to load templates from the file system or other locations. Even if you are creating templates from strings by using the constructor of `Template` class, an `environment` is created automatically for you, albeit a shared one.

Most applications will create one `Environment` object on application initialization and use that to load templates. In some cases it's however useful to have multiple environments side by side, if different configurations are in use.



SPHINX

PYTHON DOCUMENTATION GENERATOR

[Sphinx home](#) | [Documentation](#) > [previous](#) | [next](#) | [modules](#) | [index](#)

HTML theming support

New in version 0.6:

Sphinx supports changing the appearance of its HTML output via themes. A theme is a collection of HTML templates, stylesheets and other static files. Additionally, it has a configuration file which specifies from which theme to inherit, which highlighting style to use, and what options exist for customizing the theme's look and feel.

Themes are meant to be project-unaware, so they can be used for different projects without change.

Using a theme

Using an existing theme is easy: if the theme is builtin to Sphinx, you only need to set the `html_theme` config value. With the `html_theme_options` config value you can set theme-specific options that change the look and feel. For example, you could have the following in your `conf.py`:

```

html_theme = "default"
html_theme_options = {
    "rightsidebar": "true",
    "relbarbgcolor": "black"
}
  
```

Table Of Contents

- HTML theming support
- Using a theme
- Builtin themes
- Creating themes
- Templating
- Static templates

Previous topic

The built configuration file

Next topic

Templating

This Page

Show source

Quick search

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

sphinxdoc

python-sqlparse v0.1.0 documentation

[INDEX](#) | [MODULES](#) | [NEXT](#) | [PREVIOUS](#)

Analyzing the Parsed Statement

When the `parse()` function is called the returned value is a tree-ish representation of the analyzed statements. The returned objects can be used by applications to retrieve further information about the parsed SQL.

Base Classes

All returned objects inherit from these base classes. The `Token` class represents a single token and `TokenList` class is a group of tokens. The latter provides methods for inspecting it's child tokens.

```

class sqlparse.sql.Token(object, value)
    Base class for all other classes in this module.

    It represents a single token and has two instance attributes: value is the unchange value of the token and type is the type of the token.
  
```

CONTENT

- Introduction
- sqlparse — Parse SQL statements
- Analyzing the Parsed Statement
- Base Classes
- SQL Representing Classes
- User Interfaces
- Changes in python-sqlparse

SEARCH

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



Read the docs

<http://readthedocs.org/>

저장소를 등록하면
자동으로 Sphinx를 빌드해서 올려줍니다.

Happy documentation :)