SphinxTest Documentation

Release packagetest

Lucas GC Bandeira

Mar 06, 2019

Quick start:

CHAPTER 1

The index.rst file

This file is located within the *docs* root and creates the table of contents (TOC) for displaying next to the page.

```
.
├── _build
│   └── html
├── _static
│   ├── aboutindex.rst
│   ├── introduction.rst
│   └── numpydocstrings.rst
├── _templates
├── conf.py
├── index.rst <— the index file
├── make.bat
└── Makefile
```

index.rst

While not mandatory, this helps with navigation quite a bit.

Unfortunately, the index.rst is much more difficult to automate as it is the basic source file of Sphinx.

If a new modules is created, its path needs to be written down in the index.rst file for it to show up in the TOC.

CHAPTER 2

Getting Started with this test

Sphinx is made with reStructured text. It is a bit different than markdown.

- This is some wild stuff right here.
- directives use ":"
- it is quite annoying to get used to

2.1 Some math

$$\frac{\sum_{t=0}^{N} f(t,k)}{N}$$

That was some math. I bet it looks cool. Every tool I have for previewing this sucks.

2.2 Using markdown

Markdown may be used at the cost of some advanced functionalities

Blog post of some nerd ranting about it

Main points:

- Markdown has no set standard
  - There are a million interpreters with millions of syntax
- Harder to expand
  - May rely on CSS stuff
  - Breaks portability for other tools

Basically,

reStructured text is awful to write but is more stable and standard for documentation

CHAPTER 3

Docstring Format

Back in the day, docstrings used to be just plain reStructuredText inside of docstrings.

- It was very gross.
- For everyone.

3.1 Solution

Numpy Project and Google defined a non-gross docstring formats that everyone accepts and Sphinx recognizes.

- Sphinx needs an addon called "Napoleon" to interpret them.
- It is not a problem. This is defined in the doc's conf.py.

Numpy docstrings Example

conf.py:

```
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.intersphinx',
    'sphinx.ext.todo',
    'sphinx.ext.coverage',
    'sphinx.ext.mathjax',
    'sphinx.ext.ifconfig',
    'sphinx.ext.viewcode',
    'sphinx.ext.githubpages',
    'sphinx.ext.napoleon', # <- Here is Napoleon
]
```

See also: Google docstrings Example

The code in *sphinxtest.alpha* and *sphinxtest.beta* utilize this Docstring format and are automatically generated.

CHAPTER 4

Testing the Code

The *Foo* is an extremely powerful class. The number of bar power defined by its *power*, which is a int.

alpha	The alpha __init__ docstr

4.1 sphinxtest.alpha package

4.1.1 Module contents

The alpha __init__ docstr

class Foo(*barPower*)

Bases: object

This is the Foo class.

Create powerful Foo objects for generating foobars

power (*int*)

Amount of bar energy contained in this Foo instance.

Parameters **barPower** (*int*) – The bar energy to be contained in this Foo instance.

Raises AssertError – If barPower is less than 6, crap itself.

Examples

```
>>> from test import Foo
>>> x = Foo(6)
>>> x.bar()
foobar0
```

(continues on next page)

(continued from previous page)

```
foobar1
foobar2
foobar3
foobar4
foobar5
```

bar()
Generate foobars based on bar energy.

CHAPTER 5

sphinxtest package

5.1 Subpackages

5.1.1 sphinxtest.beta package

Module contents

The __init__ docstr

class Waldo(*pos*)

Bases: object

This is the Waldo class

Parameters

- **pos** (*(Tuple(int,int,int)):*) – Waldo's current position
- **pos** – Start Waldo in this position

whereIsHe()

Show Waldo's Position

Returns 3 dimension coordinate of Waldo's position

Return type list

Example

```
>>> from sphinxtest.beta import Waldo
>>> w=Waldo([2,3,5])
>>> w.whereIsHe()
[2,3,5]
```

5.2 Module contents

5.2.1 Top level doctstring

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index