
sphinxdoc-test Documentation

Release 0.1

Ryan Dale

January 11, 2011

CONTENTS

1	Publishing sphinx-generated docs on github	3
2	Protocol	5
2.1	Set up main repository	5
2.2	Set up sphinx within main repository	5
2.3	Set up separate docs repository	5
2.4	Makefile changes	6
2.5	index.rst changes	7
3	Creating and committing workflow	9
4	Directory structure	11
5	Indices and tables	13

Contents:

PUBLISHING SPHINX-GENERATED DOCS ON GITHUB

This strategy uses ideas from <http://lucasbardella.com/report/hosting-your-sphinx-docs-in-github/>, which uses a separate directory for docs and keeps the autogenerated stuff out of the main repo. This in contrast to suggestions on <http://pages.github.com/>, which does stuff within the repo directory. Using a separate docs dir made things easier for me to figure out and configure easier with the Sphinx makefile.

See <http://daler.github.com/sphinxdoc-test> for the Sphinx-generated version of this README, created using the commands documented in it...

PROTOCOL

2.1 Set up main repository

First set up your main repo. These are the commands I used to set up this very repo (how meta!):

```
mkdir sphinxdoc-test
cd sphinxdoc-test
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@github.com:daler/sphinxdoc-test.git
git push origin master
```

Throughout this document, I'll refer to this as the 'main repo' or the 'code dir'.

2.2 Set up sphinx within main repository

Make a dir, `docs`, that will store documentation source from Sphinx:

```
mkdir sphinxdoc-test/docs
```

Then set up Sphinx from the `docs` dir, accepting all the defaults as you see fit:

```
cd docs
sphinx-quickstart
```

2.3 Set up separate docs repository

Now we need to set up a completely new directory that will serve as the build directory for Sphinx. Here I'm calling it `sphinxdoc-test-docs`. Note that it's outside of the main repo dir:

```
cd ..
mkdir sphinxdoc-test-docs
cd sphinxdoc-test-docs
```

Then clone the repo you just set up on github into a dir called `html` (which will be created automatically with the following command):

```
git clone git@github.com:daler/sphinxdoc-test.git html
cd html
```

The `html` dir now has a clone of the repo.

The following commands do git fancy stuff that I don't completely understand yet, suffice to say that after these 3 commands you get a new branch, `gh-pages`, the branch is cleaned out with no files in it, and checks out the `gh-pages` branch:

```
git symbolic-ref HEAD refs/heads/gh-pages # auto-switches branches to gh-pages
rm .git/index
git clean -fdx
```

Make sure we're on `gh-pages`:

```
git branch
```

I'll refer to this as the 'gh-pages repo'.

2.4 Makefile changes

OK, now the docs repo is set up. Now it's time to make some changes to the sphinx-generated Makefile back in the main repo so that it builds documentation in our new `gh-pages` branch and directory, instead of cluttering the main code dir.

So go back to the code dir's `doc` dir:

```
cd ../sphinxdoc-test
cd docs
```

Here are the changes we're going to make to `sphinxdoc-test/docs/Makefile` ... first, change:

```
BUILDDIR      = build
```

to:

```
BUILDDIR      = ../../sphinxdoc-test-docs
PDFBUILDDIR    = /tmp
PDF            = ../manual.pdf
```

The first new line points to the new dir and `gh-pages` branch we just set up. So now, running `make html` in `sphinxdoc-test/docs` will create an `html` dir in `../../sphinxdoc-test-docs` ... and luckily, that's exactly what we set up the `gh-pages` repo in.

Before the next two lines make sense, need to make another change ... I've added commented lines pointing to the changes:

```
latexpdf:
    $(SPHINXBUILD) -b latex $(ALLSPHINXOPTS) $(BUILDDIR)/latex
    @echo "Running LaTeX files through pdflatex..."
    make -C $(BUILDDIR)/latex all-pdf
    @echo "pdflatex finished; the PDF files are in $(BUILDDIR)/latex."
```

to:

```
latexpdf:
    $(SPHINXBUILD) -b latex $(ALLSPHINXOPTS) $(PDFBUILDDIR)/latex
    #                                     ^^^
    @echo "Running LaTeX files through pdflatex..."
```

```
make -C $(PDFBUILDDIR) /latex all-pdf
#      ^^^
cp $(PDFBUILDDIR) /latex/*.pdf $(PDF)
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
@echo "pdflatex finished; see $(PDF) "
```

All these PDF build dir changes put all the LaTeX stuff in a temporary directory, and then only copy the resulting PDF file to the root dir of the main repo. So no cluttering of the main repo with autogenerated doc files, only the latest build of the PDF manual is included.

2.5 index.rst changes

Next, I'd like to only worry about making changes in a single file (README.rst), and have that propagated to all the docs in various places. On github, if you have a README.rst file in the root dir, it'll be converted to nice-ish looking docs. (Sphinx is much better looking, plus can include module, class, and function documentation to boot, hence going through all this trouble).

So we need to point sphinx's `index.rst` to the `README.rst` file in the root of the main repo. Turns out that relative path names don't work in `index.rst`, so here's a workaround:

Make a new file, `sphinxdoc-test/docs/source/includeme.rst`. In there, put an include directive pointing to the true "README.rst". So `includeme.rst` should look like this:

```
.. include:: ../../README.rst
```

Then in `index.rst`, add `includeme` to the `toctree`. So the relevant part of `index.rst` should look something like:

```
.. toctree::
    :maxdepth: 2

    includeme
```

OK, we should be done with the setup now.

CREATING AND COMMITTING WORKFLOW

Commit all code and README.rst (and any other doc source files) in the main repo, like always:

```
git add docs
git add README.rst
git commit -m "added docs and README.rst"
```

Then, when you're ready to recreate the docs:

```
cd docs
make html
make latexpdf
```

Should probably add the newly built manual:

```
cd ..
git add manual.pdf
git commit -m "added manual.pdf"
```

Next, change to the gh-pages repo dir and commit the stuff that the `make html` command made:

```
cd ../sphinxdoc-test-docs
git add .
git commit -m "rebuilt docs"
```

And then publish the newly built docs:

```
git push origin gh-pages
```

Rinse and repeat. Of course, you could always add a task to the Makefile to do this building and committing docs, something like:

```
buildandcommithtml: html latexpdf

    cd $(BUILDDIR)/html; git add . ; git commit -m "rebuilt docs"; git push origin gh-pages
```


DIRECTORY STRUCTURE

So that we're on the same page, the final directory structure looks like this:

```
sphinxdoc-test
|-- pymodule                <-- whatever your normal python package dir structure is
|   |-- somepythonmodule.py
|   `-- othercode.py
|-- docs
|   |-- Makefile            <-- edited as described above
|   `-- source
|       |-- conf.py
|       |-- includeme.rst    <-- edited as described above
|       `-- index.rst        <-- edited as described above
|-- manual.pdf              <-- created by running make latexpdf
`-- README.rst              <-- where you do most of your writing
```

```
sphinxdoc-test-docs
|-- doctrees                <-- this dir is autogenerated, but not
|   |-- environment.pickle   committed to gh-pages
|   |-- includeme.doctree
|   |-- index.doctree
|   `-- README.doctree
`-- html                    <-- The docs repo, on the gh-pages branch.
    |-- genindex.html        Everything under here is committed.
    |-- includeme.html
    |-- index.html
    |-- objects.inv
    |-- README.html
    |-- search.html
    |-- searchindex.js
    |-- _sources
    |   |-- includeme.txt
    |   |-- index.txt
    |   `-- README.txt
    `-- _static
        |-- basic.css
        |-- default.css
        |-- doctools.js
        |-- file.png
        |-- jquery.js
        |-- minus.png
        |-- plus.png
        |-- pygments.css
        `-- searchtools.js
```

```
|-- sidebar.js  
'-- underscore.js
```


INDICES AND TABLES

- *genindex*
- *modindex*
- *search*