sphinx-git Documentation

Release 8

Daniel Watkins

November 27, 2013

Contents

sphinx-git is an extension to the Sphinx documentation tool that allows you to include excerpts from your git history within your documentation. This could be used for release changelogs, to pick out specific examples of history in documentation, or just to surface what is happening in the project.

See *Recent Changes* below for an example of what can be done with it.

CHAPTER 1

Contents

1.1 Getting Started

1.1.1 Including sphinx-git In Your Project

This guide assumes that you already have a Sphinx documentation project configured and building. If that is not the case, see the Sphinx documentation first and then come back.

Installing sphinx-git

The first thing you will need to do is install sphinx-git:

```
pip install sphinx-git
```

You may also want to include it in your setup.py or requirements.txt to ensure that sphinx-git is installed wherever you generate your documentation; each project will probably have a different way of doing this.

Including sphinx-git In Your Sphinx Configuration

Once you have installed sphinx-git, you need to configure Sphinx to look for it. Find the Sphinx conf.py which is used to generate your documentation. Somewhere in that file (generally towards the top), you will find the `extensions` setting. Add `sphinx_git` to this list (note the underscore):

```
extensions = ['sphinx_git']
```

Add A git Changelog To Your Project

All the hard parts are done, now you can add a git changelog to your project! Find a documentation file where you want it and add:

```
Recent Changes
--------------

.. git_changelog::
```

Build your documentation and, voila!, you have a git changelog right there in your docs!

There are a number of ways you can configure sphinx-git to output precisely what you want, which are outlined in the next section of the documentation.

1.2 Using sphinx-git

Currently, sphinx-git provides a single extension to Sphinx: the `git_changelog` directive.

1.2.1 git_changelog Directive

The `git_changelog` directive produces a list of commits in the repository in which the documentation build is happening.

By default, it will output the most recent 10 commits. So:

```
.. git_changelog::
```

produces:

- **Finalise CHANGELOG for v8.** by *Daniel Watkins* at *2013-11-27 03:41:23*
- **Add Read the Docs documentation to index.** by *Daniel Watkins* at *2013-11-26 11:52:59*
- **Add documentation on how to configure on Read the Docs.** by *Daniel Watkins* at *2013-11-26 11:51:47*
- **Reword README intro and add to index.rst.** by *Daniel Watkins* at *2013-11-26 10:28:40*
- **Replace most of the README with a link to RtD.** by *Daniel Watkins* at *2013-11-26 10:03:25*
- **Update PR checklist now that examples.rst is gone.** by *Daniel Watkins* at *2013-11-26 09:59:05*
- **Add Using section, superseding examples.rst.** by *Daniel Watkins* at *2013-11-26 09:57:56*
- **Mention move to package in CHANGELOG.** by *Daniel Watkins* at *2013-11-26 07:21:00*
- **Output detailed commit messages as paragraphs rather than captions to fix PDF output.** by *Daniel Watkins* at *2013-11-26 07:06:59*
- **Update .gitignore.** by *Daniel Watkins* at *2013-11-26 06:58:54*

As you can see, each revision has the message, author and date output in a list. If a commit has a detailed message (i.e. any part of the commit message that is not on the first line), that will be output below the list item for that commit.

Changing Number of Revisions in Output

If you want to change the number of revisions output by `git_changelog`, then you can specify the `:revisions:` argument. So:

```
.. git_changelog::
    :revisions: 2
```

produces:

- **Finalise CHANGELOG for v8.** by *Daniel Watkins* at *2013-11-27 03:41:23*
- **Add Read the Docs documentation to index.** by *Daniel Watkins* at *2013-11-26 11:52:59*

If you specify more revisions than the history contains, all revisions in the history will be displayed.

Specifying Range of Revisions to Output

If you want even more control over the output of `git_changelog`, then you can specify precisely the revisions you want included using the `:rev-list:` argument. So:

```
.. git_changelog::
    :rev-list: v3..v4
```

produces a list of all the commits between the v3 and v4 tags:

- **Add feature credits to CHANGELOG.** by *Daniel Watkins* at *2013-11-16 17:08:14*
- **Add v4 changelog entry.** by *Daniel Watkins* at *2013-11-16 17:06:36*
- **Merge pull request #10 from OddBloke/rev_list** by *OddBloke* at *2013-11-16 17:02:41* Add the possiblity to specify commits using a rev-list parameter.
- **Make a couple of formatting changes.** by *Daniel Watkins* at *2013-11-16 16:59:21*
- **add rev-list explanation in README** by *Gregory Eric Sanderson* at *2013-09-30 12:23:39*
- **display a changelog using a range of commits** by *Gregory Eric Sanderson* at *2013-09-30 12:06:15* Adds a new option 'rev-list'. rev-list lets you define which commit to start from when displaying the changelog. You can use a tag, a branch, a commit hash, an explicit range, or anything else supported by git-rev-parse. Examples:

  .. git_changelog:: :rev-list: v1.0..HEAD

  .. git_changelog:: :rev-list: master..topicbranch

  Consult the man pages of git-rev-parse for more details on the syntax.
- **use a version of gitpython that provides iter_commits()** by *Gregory Eric Sanderson* at *2013-09-30 12:04:34*
- **Add CHANGELOG.** by *Daniel Watkins* at *2013-07-07 03:35:43*
- **Bump to v4 (for development).** by *Daniel Watkins* at *2013-07-07 03:35:32*

and:

```
.. git_changelog::
    :rev-list: v1
```

gives you a list of all commits up to the v1 tag (most of which involved me wrestling with setuptools):

- **I despise setuptools.** by *Daniel Watkins* at *2012-07-09 11:46:00*
- **Start again.** by *Daniel Watkins* at *2012-07-09 11:39:20*
- **Fix requirements.** by *Daniel Watkins* at *2012-07-09 11:37:48*
- **Add setup.py.** by *Daniel Watkins* at *2012-07-09 11:34:03*
- **Add README.** by *Daniel Watkins* at *2012-07-09 11:33:18*
- **Initial implementation.** by *Daniel Watkins* at *2012-07-09 11:16:13*

`:rev-list:` lets you specify revisions using anything that `git rev-parse` will accept. See the man page for details.

Warning: The `:revisions:` argument and the `:rev-list:` argument don't play nicely together. `:rev-list:` will always take precedence, and all commits specified by the revision specification be output regardless of the `:revisions:` argument [a].
Sphinx will output a warning if you specify both.

[a] *Patches welcome!*

Preformatted Output for Detailed Messages

If you would prefer for the detailed commit messages to be output as preformatted text (e.g. if you include code samples in your commit messages), then you can specify this preference using the `:detailed-message-pre:` argument. So:

```
.. git_changelog::
    :rev-list: 3669419^..3669419
    :detailed-message-pre: True
```

becomes:

- **display a changelog using a range of commits** by *Gregory Eric Sanderson* at *2013-09-30 12:06:15*

  ```
  Adds a new option 'rev-list'.
  rev-list lets you define which commit to start from when displaying the
  changelog. You can use a tag, a branch, a commit hash, an explicit
  range, or anything else supported by git-rev-parse. Examples:

  .. git_changelog::
      :rev-list: v1.0..HEAD

  .. git_changelog::
      :rev-list: master..topicbranch

  Consult the man pages of git-rev-parse for more details on the syntax.
  ```

1.3 Enabling on Read the Docs

Read the Docs is an excellent website that hosts Sphinx-generated documentation (including the documentation for this project, which is probably where you are reading it). This document assumes that you already have your project configured on Read the Docs, using their default configuration [1].

As a custom extension, sphinx-git isn't supported out-of-the-box by Read the Docs, but it is very easy to get it working!

1.3.1 Creating a Documentation Requirements File

The first thing you'll need to do is create a pip requirements file for your documentation. Create a file containing:

```
sphinx-git
```

and commit it somewhere in your repository [2] (I will assume it is in `requirements/doc.txt` for the rest of this document).

1.3.2 Configuring Read the Docs

Navigate to the Read the Docs admin page for your project. This will be of the form `https://readthedocs.org/dashboard/<PROJECT NAME>/edit/`. Once on this page, you need to do two things:

[1] Follow the Read the Docs getting started guide if you haven't already.

[2] You should probably pin that requirement to a specific version, but that is outside the scope of this documentation. This is probably a good place to start reading about it: http://nvie.com/posts/pin-your-packages/

- Tick the box under "Use virtualenv", so that Read the Docs will install our custom documentation requirements, and
- Enter your documentation requirements file name in the "Requirements file" box (`requirements/doc.txt` from above).

Submitting the form should cause your project to be rebuilt, now with sphinx-git available!

1.4 Contributing

sphinx-git is already the work of more than just myself! There are a number of ways that you can contribute to the sphinx-git project.

1.4.1 Open Issues on GitHub

If there's a problem with how sphinx-git works, or if there's a feature that you'd like to see, open up an issue in GitHub. Give as much information as you can and I'll do my best to get to it!

1.4.2 Submit a Patch

If you feel confident enough, have a stab at scratching your own itch in sphinx-git. Fork the project on GitHub, make your changes and submit a pull request.

Pull requests will need to pass the Travis CI build; you can run this by doing the following:

```
$ pip install -r requirements.txt  # This installs all of the development requirements
$ travis-solo
```

This will run the build on both Python 2.6 and Python 2.7. If you're on an environment that doesn't have both available then do the best you can, and then open up your pull request; Travis will pick this up and build it for you.

Pull Request Checklist

Once you've got a patch ready, check the following things:

- You've written tests for your change
- The Travis CI build passes; this includes:
  - PEP-8 on the sphinx_git package and the tests
  - Pylint on the sphinx_git package
  - Passing unit tests (of course!)
- You've added a line to the CHANGELOG
- You've added documentation (if appropriate)

CHAPTER 2

Recent Changes

- **Finalise CHANGELOG for v8.** by *Daniel Watkins* at *2013-11-27 03:41:23*
- **Add Read the Docs documentation to index.** by *Daniel Watkins* at *2013-11-26 11:52:59*
- **Add documentation on how to configure on Read the Docs.** by *Daniel Watkins* at *2013-11-26 11:51:47*
- **Reword README intro and add to index.rst.** by *Daniel Watkins* at *2013-11-26 10:28:40*
- **Replace most of the README with a link to RtD.** by *Daniel Watkins* at *2013-11-26 10:03:25*
- **Update PR checklist now that examples.rst is gone.** by *Daniel Watkins* at *2013-11-26 09:59:05*
- **Add Using section, superseding examples.rst.** by *Daniel Watkins* at *2013-11-26 09:57:56*
- **Mention move to package in CHANGELOG.** by *Daniel Watkins* at *2013-11-26 07:21:00*
- **Output detailed commit messages as paragraphs rather than captions to fix PDF output.** by *Daniel Watkins* at *2013-11-26 07:06:59*
- **Update .gitignore.** by *Daniel Watkins* at *2013-11-26 06:58:54*