sphinx-git Documentation

Release 10.1.1

Daniel Watkins

Dec 16, 2017

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.

Add Details of the Latest Commit to Your Project

You can also display information about the state of the repository when the documentation was compiled with the `git_commit_detail` directive:

```
.. git_commit_detail::
    :branch:
    :commit:
```

1.2 Using sphinx-git

Currently, sphinx-git provides two extensions to Sphinx: the `git_changelog` and `git_commit_detail` directives.

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:

- **Prepare v10.1.1 release** by *Daniel Watkins* at *2017-12-16 17:35:13*
- **Make CHANGELOG formatting consistent** by *Daniel Watkins* at *2017-12-16 17:34:10*
- **Update CHANGELOG** by *Daniel Watkins* at *2017-12-16 17:32:52*
- **Merge pull request #47 from OddBloke/fix_rtd** by *Daniel Watkins* at *2017-12-16 17:31:05* Handle detached HEADs
- **Fix operating on a detached HEAD** by *Daniel Watkins* at *2017-12-16 17:27:27* This (a) should work, and (b) is the environment that ReadTheDocs builds in, so we need this to get updated docs.
- **Inline unnecessary method** by *Daniel Watkins* at *2017-12-16 17:27:04*
- **Prepare for development in next release** by *Daniel Watkins* at *2017-12-16 17:12:28*
- **Finalise v10.1.0** by *Daniel Watkins* at *2017-12-16 17:06:33*
- **Merge pull request #46 from OddBloke/boeddeker** by *Daniel Watkins* at *2017-12-15 23:10:39* Add repo-dir option plus tests
- **Add CHANGELOG entry** by *Daniel Watkins* at *2017-12-15 23:06:57*

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:

- **Prepare v10.1.1 release** by *Daniel Watkins* at *2017-12-16 17:35:13*
- **Make CHANGELOG formatting consistent** by *Daniel Watkins* at *2017-12-16 17:34:10*

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 23:08:14*
- **Add v4 changelog entry.** by *Daniel Watkins* at *2013-11-16 23:06:36*
- **Merge pull request #10 from OddBloke/rev_list** by *OddBloke* at *2013-11-16 23:02:41* Add the possiblity to specify commits using a rev-list parameter.
- **Make a couple of formatting changes.** by *Daniel Watkins* at *2013-11-16 22:59:21*
- **add rev-list explanation in README** by *Gregory Eric Sanderson* at *2013-09-30 17:23:39*
- **display a changelog using a range of commits** by *Gregory Eric Sanderson* at *2013-09-30 17:06:15* 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 17:04:34*
- **Add CHANGELOG.** by *Daniel Watkins* at *2013-07-07 08:35:43*
- **Bump to v4 (for development).** by *Daniel Watkins* at *2013-07-07 08:35:32*

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

`: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[1].

Sphinx will output a warning if you specify both.

Filter Revisons to Matching Only Certain Files Based on Filenames

If you only want to see the changelog regarding certain files (eg. for devops reasons you need to have both SaSS and CSS in your repository or you only want to see the changes made to the docs directory) you can use the `:filename_filter:` argument with `git_changelog`. `:filename_filter:` is expecting anything that can be evaluated as a regular expression. So:

```
.. git_changelog::
    :filename_filter: doc/.*\.rst
```

will produce the list of commits that modified documentation content.

Note: The `:filename_filter:` argument is compatible with both `:revisions:` and `:rev-list:`. Filtering on filenames is then performed on the selected (number of) revisions.

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 17:06:15*

  ```
  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
  ```

[1] *Patches welcome!*

```
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.
```

1.2.2 git_commit_detail Directive

The `git_commit_detail` directive produces information about the current commit in the repository against which the documentation is being built. The following options are available:

branch Display the branch name.

commit Display the commit hash.

sha_length Set the number of characters of the hash to display.

no_github_link By default, if the repository's origin remote is GitHub, the commit will link to the GitHub page for the commit. Use this option to disable this.

uncommitted Show a warning if there are uncommitted changes in the repository.

untracked Show a warning if there are untracked files in the repository directory.

For example:

```
.. git_commit_detail::
    :branch:
    :commit:
    :sha_length: 10
    :uncommitted:
    :untracked:
```

becomes

Commit 466cd70261

> **Warning:** There were uncommitted changes when this was compiled.

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

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:

- 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, which uses tox. You can run this by doing the following:

```
$ pip install tox
$ tox
```

This will run the build on all supported Python versions. 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.

[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/

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

- **Prepare v10.1.1 release** by *Daniel Watkins* at *2017-12-16 17:35:13*
- **Make CHANGELOG formatting consistent** by *Daniel Watkins* at *2017-12-16 17:34:10*
- **Update CHANGELOG** by *Daniel Watkins* at *2017-12-16 17:32:52*
- **Merge pull request #47 from OddBloke/fix_rtd** by *Daniel Watkins* at *2017-12-16 17:31:05* Handle detached HEADs
- **Fix operating on a detached HEAD** by *Daniel Watkins* at *2017-12-16 17:27:27* This (a) should work, and (b) is the environment that ReadTheDocs builds in, so we need this to get updated docs.
- **Inline unnecessary method** by *Daniel Watkins* at *2017-12-16 17:27:04*
- **Prepare for development in next release** by *Daniel Watkins* at *2017-12-16 17:12:28*
- **Finalise v10.1.0** by *Daniel Watkins* at *2017-12-16 17:06:33*
- **Merge pull request #46 from OddBloke/boeddeker** by *Daniel Watkins* at *2017-12-15 23:10:39* Add repo-dir option plus tests
- **Add CHANGELOG entry** by *Daniel Watkins* at *2017-12-15 23:06:57*