Read The Docs Documentation

Release 1.0

Eric Holscher, Charlie Leifer, Bobby Grace

July 06, 2015

Contents

Read the Docs hosts documentation for the open source community. It supports Sphinx docs written with reStructuredText, and can pull from your Subversion, Bazaar, Git, and Mercurial repositories. The code is open source, and available on github.

CHAPTER 1

Contents

1.1 Getting Started

It's really easy to start using RTD for your project's documentation. This section shows you how.

If you are already using Sphinx for your docs, skip ahead to *Import Your Docs*.

1.1.1 Write Your Docs

Install Sphinx, and create a directory inside your project to hold your docs:

```
$ cd /path/to/project
$ mkdir docs
```

Run `sphinx-quickstart` in there:

```
$ cd docs
$ sphinx-quickstart
```

This will walk you through creating the basic configuration; in most cases, you can just accept the defaults. When it's done, you'll have an `index.rst`, a `conf.py` and some other files. Add these to revision control.

Now, edit your `index.rst` and add some information about your project. Include as much detail as you like (refer to the reStructuredText syntax if you need help). Build them to see how they look:

```
$ make html
```

Edit and rebuild until you like what you see, then commit and/or push your changes to your public repository.

1.1.2 Import Your Docs

Sign up for an account and log in. Visit your dashboard and click Import to add your project to the site. Fill in the name and description, then specify where your repository is located. This is normally the URL or path name you'd use to checkout, clone, or branch your code. Some examples:

- Git: `http://github.com/ericholscher/django-kong.git`
- Subversion: `http://varnish-cache.org/svn/trunk`
- Mercurial: `https://bitbucket.org/ianb/pip`
- Bazaar: `lp:pasta`

Add an optional homepage URL and some keywords, then click "Create".

Within a few minutes your code will automatically be fetched from your public repository, and the documentation will be built.

If you want to keep your code updated as you commit, configure your code repository to hit our Post Commit Hooks. Otherwise your project will get rebuilt nightly.

1.2 Webhooks

Web hooks are pretty amazing, and help to turn the web into a push instead of pull platform. We have support for hitting a URL whenever you commit to your project and we will try and rebuild your docs. This only rebuilds them if something has changed, so it is cheap on the server side. As anyone who has worked with push knows, pushing a doc update to your repo and watching it get updated within seconds is an awesome feeling.

1.2.1 Github

If your project is hosted on Github, you can easily add a hook that will rebuild your docs whenever you push updates:

- Go to the "admin" page for your project
- Click "Service Hooks"
- In the available service hooks, click "ReadTheDocs"
- Check "Active"
- Click "Update Settings"

1.2.2 Others

Your ReadTheDocs project detail page has your post-commit hook on it; it will look something along the lines of `http://readthedocs.org/build/<pk>`. Regardless of which revision control system you use, you can just hit this URL to kick off a rebuild.

You could make this part of a hook using Git, Subversion, Mercurial, or Bazaar, perhaps through a simple script that accesses the build URL using `wget` or `curl`.

1.3 Alternate Domains

Read the Docs supports a number of custom domains for your convinence. Shorter urls make everyone happy, and we like making people happy!

1.3.1 Subdomain Support

Every project has a subdomain that is available to serve it's documentation. If you go to <slug>.readthedocs.org, it should show you the latest version of documentation. A good example is http://pip.readthedocs.org

1.3.2 CNAME Support

If you have your own domain, you can still host with us. If you point a CNAME record in your database to the subdomain for your project, it should magically serve your latest documentation on the custom domain. Using pip as another example, http://www.pip-installer.org resolves, but is hosted on our infrastructure.

1.3.3 RTFD.org

You can also use <slug>.rtfd.org as a short URL for the front page of your subdomain'd site. For example, http://pip.rtfd.org redirects to it's documentation page. We're looking for more fun ways to use this domain, so feel free to suggestion an idea.

1.4 Getting Help

The easiest way to get help with the project is to join the `#readthedocs` channel on Freenode. We hang out there and you can get real-time help with your projects. The other good way is to open an issue on Github.

The email at ReadTheDocs@readthedocs.org is also available for support.

1.5 Installation

Installing RTD is pretty simple. Here is a step by step plan on how to do it.

First, obtain Python and virtualenv if you do not already have them. Using a virtual environment will make the installation easier, and will help to avoid clutter in your system-wide libraries. You will also need Git in order to clone the repository.

Once you have these, create a virtual environment somewhere on your disk, then activate it:

```
virtualenv rtd
cd rtd
source bin/activate
```

Create a folder in here, and clone the repository:

```
mkdir checkouts
cd checkouts
git clone http://github.com/rtfd/readthedocs.org.git
```

Next, install the dependencies using `pip` (included with virtualenv):

```
cd readthedocs.org
pip install -r pip_requirements.txt
```

This may take a while, so go grab a beverage. When it's done, build your database:

```
./manage.py syncdb
```

This will prompt you to create a superuser account for Django. Do that. Then:

```
./manage.py migrate
```

If you like, you can load up some test projects:

```
./manage.py loaddata test_data
./manage.py update_repos
```

Or you can skip it and start with a fresh, empty site. Finally, you're ready to start the webserver:

```
./manage.py runserver
```

Visit http://127.0.0.1:8000/ in your browser to see how it looks; you can use the admin interface via http://127.0.0.1:8000/admin (logging in with the superuser account you just created).

1.5.1 What's available

After registering with the site (or creating yourself a superuser account), you will be able to log in and view the dashboard

From the dashboard you can either create new documentation, or import your existing docs provided that they are in a git or mercurial repo.

Creating new Docs

One of the goals of readthedocs.org is to make it easy for any open source developer to get high quality hosted docs with great visibility! We provide a simple editor and two sample pages whenever a new project is created. From there its up to you to fill in the gaps - we'll build the docs, give you access to history on every revision of your files, and we plan on adding more features in the weeks and months to come.

Importing existing docs

The other side of readthedocs.org is hosting the docs you've already built. Simply provide us with the clone url to your repo, we'll pull your code, extract your docs, and build them! We make available a post-commit webhook that can be configured to update the docs on our site whenever you commit to your repo, effectively letting you 'set it and forget it'.

1.5.2 Caveats

We are auto-importing and generating `conf.py` files, so projects with special extensions, themes, or templates won't work correctly. This is because of the possibility of code execution within the python files. We are planning to support popular themes and white list users that we trust to have these abilities.

1.6 API

This is the Read The Docs API documentation, autogenerated from the source code.

1.6.1 bookmarks

bookmarks.admin

bookmarks.models

bookmarks.urls

bookmarks.views

1.6.2 builds

builds.admin

builds.models

builds.urls

builds.views

1.6.3 core

core.admin

core.forms

core.middleware

core.models

class core.models.**UserProfile**(**args*, ***kwargs*)
Additional information about a User.

get_contribution_details()
Gets the line to put into commits to attribute the author.

Returns a tuple (name, email)

core.search_sites

core.views

core.management.commands

This is where custom `manage.py` commands are defined.

1.6.4 projects

projects.admin

projects.constants

Default values and other various configuration for projects, including available theme names and repository types.

projects.forms

projects.models

projects.search_indexes

projects.tasks

projects.utils

Utility functions used by projects.

projects.utils.**diff**(*txt1*, *txt2*)
Create a 'diff' from txt1 to txt2.

projects.utils.**find_file**(*file*)
Find matching filenames in the current directory and its subdirectories, and return a list of matching filenames.

projects.utils.**run**(**commands*)
Run one or more commands, and return `(status, out, err)`. If more than one command is given, then this is equivalent to chaining them together with `&&`; if all commands succeed, then `(status, out, err)` will represent the last successful command. If one command failed, then `(status, out, err)` will represent the failed command.

projects.utils.**safe_write**(*filename*, *contents*)
Write `contents` to the given `filename`. If the filename's directory does not exist, it is created. Contents are written as UTF-8, ignoring any characters that cannot be encoded as UTF-8.

projects.views

projects.views.public

projects.views.private

1.6.5 watching

watching.admin

watching.models

watching.urls

watching.views

b

c

p

w

B

C

D

F

G

P

R

S

U

W