django-redis-views Documentation

Release 0.2.2

Kevin London

October 21, 2015

Contents

Contents:

CHAPTER 1

django-redis-views

Simple Redis-based generic views for serving your Django-backed Ember CLI apps.

1.1 Documentation

The full documentation is at https://django-redis-views.readthedocs.org.

1.2 Features

- Serves your single page javascript apps easily through Django views.
- Works out-of-the-box with ember-cli-deploy and ember-deploy-redis.

1.3 Background

Ember CLI and other single-page javascript apps can be challenging to deploy.

Luke Melia presented a talk called Lightning Fast Deployment of Your Rails-backed JavaScript app, which eventually led to the creation of ember-cli-deploy.

This project acts as the glue between *ember-cli-deploy* and Django by providing generic views to serve Redis-backed index pages for single page javascript applications.

1.4 Quickstart

Let's assume we already have an Ember CLI app that we're ready to deploy. We're using the ember-deploy-redis adapter and we ran *ember deploy* to push the *index.html* file into Redis. In this case, we'll pretend that the Ember CLI project's name is *ember-cli-my-great-app*.

First, install django-redis-views:

```
pip install django-redis-views
```

In your Django settings file, set the Redis url. For example, you may want to access Redis on the localhost running on the default port. In which case, you would add something like this to the your *settings.py* file:

```
REDIS_URL = 'redis://localhost:6379/0'
```

Then, to use it in your Django project, first add a new view to a *views.py* file:

```
from redis_views import RedisView


class EmberAppIndex(RedisView):
    app_name = 'ember-cli-my-great-app'
```

And then set it a route for it in your *urls.py* file:

```
from django.conf.urls import patterns
from myapp.views import EmberAppIndex


urlpatterns = patterns('',
    url(r'^$', EmberAppIndex.as_view()),
)
```

At this point, you should be able to go to your root url and see your index page!

TODO: Walk through a full example project.

1.5 Injecting Context to your Template

If you want to use Django's template engine to replace values in your Ember index file, you can do that by injecting the context. Let's pretend that we have this very simple Ember index page:

```
<p>Hello {{ name }}!</p>
```

In order to inject *{{ name }}* from Django into the Ember index page, you'll want to add to the context. This package is built upon the generic views in Django, so we inject context the same way that they do. In your *views.py* file (using the same conventions as above):

```
from redis_views import RedisView

class EmberAppIndex(RedisView):

    . . .

    def get_context_data(self, **kwargs):
        # Call the base implementation first to get a context
        context = super(EmberAppIndex, self).get_context_data(**kwargs)
        # Add in the name value (you could also use a dynamic value from a database object)
        context['name'] = 'Joe'
        return context
```

Then, in the template, it will fill in the value with your supplied value. As mentioned in the comment, you can inject pretty much anything that could normally be handled by Django templates, such as a CSRF token.

1.6 Running the Tests

To run the tests, please do the following in your terminal:

```
# Install the testing requirements
pip install -r requirements-test.txt

# Run the tests
py.test
```

1.7 Cookiecutter Tools Used in Making This Package

- cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-redis-views
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-redis-views
$ pip install django-redis-views
```

CHAPTER 3

Usage

To use django-redis-views in a project:

```
import redis_views
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/kevinlondon/django-redis-views/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

django-redis-views could always use more documentation, whether as part of the official django-redis-views docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kevinlondon/django-redis-views/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-redis-views* for local development.

1. Fork the *django-redis-views* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-redis-views.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-redis-views
$ cd django-redis-views/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 redis_views tests
$ python setup.py test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/kevinlondon/django-redis-views/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_redis_views
```

CHAPTER 5

Credits

5.1 Development Lead

- Kevin London <kevinlondon@gmail.com>

5.2 Contributors

- Ahmed Nassar <ektebli@yahoo.com>

CHAPTER 6

History

6.1 0.2.2 (2015-10-20)

- Fixed bug with setting the socket timeout without a kwarg.

6.2 0.2.1 (2015-10-20)

- Added a socket timeout and server pinging when connecting to the server to raise a ConnectionError if there are any connection issues.
- Added error logging of missing template keys.

6.3 0.2.0 (2015-09-21)

- Changed the GET parameter value from *version* to *index_key* to match the convention established by ember-cli-deploy.

6.4 0.1.0 (2015-08-22)

- First release on PyPI.