django-gcse Documentation

Release 0.8.0

Steve Schwarz

August 27, 2014

Contents

Contents:

CHAPTER 1

django-gcse

A django reusable application for maintaining websites/data for use with Google Custom Search Engines.

1.1 Documentation

The full documentation is at http://django-gcse.rtfd.org.

1.2 Quickstart

Install django-gcse:

```
pip install django-gcse
```

Then use it in a project:

```
import gcse
```

1.3 Features

- Import existing Custom Search Engines and their Annotations via URLs or files via a management command.
- Convert OPML files into Annotations for use in a Custom Search Engine via a management command.
- Share Annotations across multiple Custom Search Engines to ease maintenance.
- Browsable views for all Custom Search Engines, Annotations and Labels. Search the entries for a CSE using Google directly from the CSE view. Browsable views can be disabled.
- All entries can be managed via django admin screens.
- TODO:
  - Slugify Label and Annotation classes.
  - Admin handle ordering of FacetItems within a CustomSearchEngine.
  - Browsing views visible via settings configuration.
  - Create demo/tests with Annotations shared across multiple CSEs.
  - Define caching attributes on XML views sent to Google.

  - Admin access to management commands.

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-gcse
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-gcse
$ pip install django-gcse
```

CHAPTER 3

Usage

To use django-gcse in a project:

```
import gcse
```

django-gcse dynamically generates the files used by Google Custom Search Engines (CSE) to implement a Linked Custom Search Engine.

Custom Search Engines must first be created within the Google administration screen **add link here** and then they are imported into django-gcse via the import_cse management command.

Due to the vast number of configuration options available django-gcse maintains a hybrid representation of the CSE. Fields likely to change are stored in the django-gcse database and less frequently changed values are stored in the XML supplied by Google. When a search is performed the database and XML data is merged and supplied to Google. Of course caching of that rarely changing view is useful.

The Linked Annotation file(s) **add link here** for a CSE are generated by collecting all the Annotations with the background Labels for that CSE and including them in the CSE XML. Each Linked Annotation file is served and is also cached.

There are three ways to add Annotations:

> # Directly via the Django admin.
>
> # From an OPML file.
>
> # From an existing Annotation XML file or URL.

Annotations can be shared across multiple CSEs by adding the background Label with the FILTER mode of the CSE to tthe Annotation.

Use the Django admin screens to enter the URLs to be searched (Annotations) and their search refinement Labels. Then you configure your Google CSE to retrieve the Annotations from your django-gcse URL(s).

django-gcse makes it easy to maintain your Annotations.

You can import an existing CSE along with it's Annotations or create a new one and populate it all from within the django-gcse admin screens.

1. First create a Google Custom Search Engine and configure the settings as you desire.
2. Download the CSE context from the Advanced tab.

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/saschwarz/django-gcse/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-gcse could always use more documentation, whether as part of the official django-gcse 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/saschwarz/django-gcse/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-gcse* for local development.

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

   ```
   $ git clone git@github.com:your_name_here/django-gcse.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-gcse
   $ cd django-gcse/
   $ 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 gcse 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/saschwarz/django-gcse/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_gcse
```

CHAPTER 5

Credits

5.1 Development Lead

- Steve Schwarz <steve@agilitynerd.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.8.0 (2013-12-20)

- First release on PyPI.