django-template-field Documentation

Release 0.3.1

Jess Johnson

Sep 06, 2017

Contents

Contents:

CHAPTER 1

django-template-field

A Django template field with a manager to return the rendered template.

Documentation

The full documentation is at https://django-template-field.readthedocs.org.

Quickstart

Install django-template-field:

```
pip install django-template-field
```

Then use it in a project:

```
from django.db import models

from templatefield import fields, managers


class TemplatedText(models.Model):
    value = fields.TemplateTextField()

    # Manager that returns rendered templates. This will be the default
    # manager since it is first. Now, when accessed via `Related Models`_
    # this field will also be rendered.
    objects_rendered = managers.RenderTemplateManager()
    # Django's default manager returns unrendered templates.
    objects_unrendered = models.Manager()
```

Extra context can be added in `settings` like so:

```
TEMPLATE_FIELD_CONTEXT = { 'template_var': value }
```

Context can also be added to querysets like so:

```
TemplatedText.objects_rendered.with_context({'template_var2': value2})
```

If you dump fixtures with `RenderTemplateManager` as the default manager, django will render the exported data. To work around that, create an alternate settings file for your project with the following setting:

```
TEMPLATE_FIELD_RENDER = False
```

Then you can dump your unrendered data like so:

```
./manage.py dumpdata myapp.mymodel --settings=myapp.dump_settings
```

Related Models

If a `TemplateTextField` will be accessed from another model through a `ForeignKey` relationship, Django will use the default manager to render the `TemplateTextField`. For example, if we define this additional model:

```
class RelatedToTemplatedText(models.Model):
    templated_text = models.ForeignKey(TemplatedText)
```

We can expect to see fields accessed via `templated_text` rendered properly.

Admin

Using `RenderTemplateManager` as the default has the unfortunate side effect of rendering your fields in the Django admin, so we have provided a class from which you can inherit to solve that problem. Ex:

```
from templatefield import admin

class TemplatedTextAdmin(admin.UnrenderedAdmin):
    ...
```

Running Tests

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install -r requirements/test.txt
(myenv) $ python runtests.py
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-template-field
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-template-field
$ pip install django-template-field
```

CHAPTER 3

Usage

To use django-template-field in a project:

```
import templatefield
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/orcasgit/django-template-field/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/orcasgit/django-template-field/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 :)

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/django-template-field.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-template-field
$ cd django-template-field/
$ 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 templatefield 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.

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/orcasgit/django-template-field/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 5

Credits

Development Lead

- Jess Johnson <jess@grokcode.com>

Contributors

- Percy Perez <percyp3@gmail.com>
- Brad Pitcher <bradpitcher@gmail.com>

CHAPTER 6

History

0.3.1 (2016-01-11)

- Add setting to disable rendering

0.3.0 (2016-01-06)

- Enable rendering in related field access

0.2.0 (2015-10-23)

- Add *with_context* to *RenderTemplateManager*

0.1.0 (2015-10-02)

- First release on PyPI.