django-rest-framework-multi-slug-field Documentation

Release 0.1.0

Piper Merriam

July 15, 2014

Contents

Contents:

CHAPTER 1

django-rest-framework-multi-slug-field

A field for representing a relationship via multiple fields on the target

1.1 Documentation

The full documentation is at https://django-rest-framework-multi-slug-field.readthedocs.org.

1.2 Quickstart

Install django-rest-framework-multi-slug-field:

```
pip install django-rest-framework-multi-slug-field
```

Then use it in a project:

```
from rest_framework_msf import MultiSlugField
```

1.3 Features

- `MultiSlugField` - Serializer field for representing a relationship via a set of fields on the target.

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-rest-framework-multi-slug-field
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-rest-framework-multi-slug-field
$ pip install django-rest-framework-multi-slug-field
```

CHAPTER 3

MultiSlugRelatedField

class `MultiSlugRelatedField`(*slug_fields*[, *many*, *required*, *queryset*])
Serializer field which may be used to represent the target of a relationship using a set of fields on the target.

For example, the following serializer:

```
class AddressSerializer(serializers.ModelSerializer):
    postal_code = serializers.SlugRelatedField(many=True, read_only=True,
                                               slug_fields=('code', 'country'))

    class Meta:
        model = Address
        fields = ('street', 'city', 'state', 'postal_code')
```

Would serialize to a representation like this:

By default this field is read-write, although you can change this behavior using the `read_only` flag.

When using `MultiSlugRelatedField` as a read-write field, you will normally want to ensure that the slug fields corresponds to a set of model field declared as `unique_together`.

3.1 Arguments

- `slug_fields` - The fields on the target that should be used to represent it. This should be a set of fields that uniquely identifies any given instance. For example, `postal_code', 'country')`. **required**
- `many` - If applied to a to-many relationship, you should set this argument to `True`.
- `required` - If set to `False` the field will accept values of `None` or the empty-string for nullable relationships.
- `queryset` - By default `ModelSerializer` classes will use the default queryset for the relationship. `Serializer` classes must either set a queryset explicitly, or set `read_only=True`.

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/pipermerriam/django-rest-framework-multi-slug-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.

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-rest-framework-multi-slug-field could always use more documentation, whether as part of the official django-rest-framework-multi-slug-field 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/pipermerriam/django-rest-framework-multi-slug-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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-rest-framework-multi-slug-field* for local development.

1. Fork the *django-rest-framework-multi-slug-field* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/django-rest-framework-multi-slug-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-rest-framework-multi-slug-field
   $ cd django-rest-framework-multi-slug-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 rest_framework_msf 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/pipermerriam/django-rest-framework-multi-slug-field/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_rest_framework_msf
```

CHAPTER 5

Credits

5.1 Development Lead

- Piper Merriam <piper@simpleenergy.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2014-01-01)

- First release on PyPI.