django-reversion-extras Documentation

Release 0.0.2

Fabio C. Barrionuevo da Luz

June 27, 2015

Contents

Contents:

CHAPTER 1

django-reversion-extras

Extra tools to work with django-reversion

Danger: It is not ready for use, it does not have tests and only serves to try to validate the use of django-reversion for things which it was not designed

1.1 Documentation

The full documentation is at https://django-reversion-extras.readthedocs.org.

1.2 Quickstart

Install django-reversion-extras:

```
pip install django-reversion-extras
```

Then use it in a project:

```
from reversion_extras.views import DetailVersionListView, UpdateVersionListView
```

`DetailVersionListView` provides the same functionality as django.views.generic.DetailView

`UpdateVersionListView` provides the same functionality as django.views.generic.UpdateView

All inject in the template context some new variables:

`object_versions_list`: contains the list of django-reversion Versions of current model instance. The same value returned from `reversion.get_for_object(model_instance)`

`model_name_versions_list`: is a alias to object_versions_list

`version_paginator` `version_page_obj` `version_is_paginated`

1.3 Features

- TODO:

Create ReversionView Create CompareVersionView

CHAPTER 2

Installation

At the command line:

```
$ pip install django-reversion-extras
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-reversion-extras
$ pip install django-reversion-extras
```

CHAPTER 3

Usage

To use django-reversion-extras in a project:

```
# views.py
from django.views import generic
from reversion_extras.views import DetailVersionListView, UpdateVersionListView

from .models import FooModel
from .forms import FooModelForm


# it works just like django.views.generic.DetailView from django
class FooModelUpdateViewWithVersionList(UpdateVersionListView):
    model = FooModel
    version_paginate_by = 20
    success_url = reverse_lazy('foomodel_list')


# it works just like django.views.generic.UpdateView from django
class FooModelDetailViewWithVersionList(UpdateVersionListView):
    model = FooModel
    version_paginate_by = 20
    form_class = FooModelForm
    success_url = reverse_lazy('foomodel_list')


class FooModelListView(generic.ListView):
    model = FooModel


# models.py
from django.db import models
import reversion

class FooModel(models.Model):

    content = models.TextField()

reversion.register(FooModel)


# urls.py
from django.conf.urls import include, url
```

```
from .views import (
    FooModelListView,
    FooModelUpdateViewWithVersionList,
    FooModelDetailViewWithVersionList
)

urlpatterns = [
    url(r'^$', FooModelListView.as_view(), name='foomodel_list'),
    url(r'^update_with_versions/(?P<pk>\d+)/$', FooModelUpdateViewWithVersionList.as_view(), name='fo
    url(r'^detail_with_versions/(?P<pk>\d+)/$', FooModelDetailViewWithVersionList.as_view(), name='fo
]
```

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

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

```
$ git clone git@github.com:your_name_here/django-reversion-extras.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-reversion-extras
$ cd django-reversion-extras/
$ 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 reversion_extras tests
$ isort --recursive reversion_extras 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/luzfcb/django-reversion-extras/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_reversion_extras
```

CHAPTER 5

Credits

5.1 Development Lead

- Fabio C. Barrionuevo da Luz <bnafta@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2015-06-26)

- First release on PyPI.

CHAPTER 7

reversion_extras

7.1 reversion_extras package

7.1.1 Submodules

7.1.2 reversion_extras.models module

7.1.3 reversion_extras.version module

7.1.4 reversion_extras.views module

class reversion_extras.views.**MultipleVersionObjectMixin**
Bases: object

A mixin for views manipulating multiple django-reversion Versions of object.

get(*request*, **args*, ***kwargs*)

get_context_data(***kwargs*)
Get the context for this view.

get_version_allow_empty()
Returns True if the view should display empty version lists, and False if a 404 should be raised instead.

get_version_context_object_name()
Get the name of the version item to be used in the context.

get_version_ordering()
Return the field or fields to use for ordering the version queryset.

get_version_paginate_by(*queryset*)
Get the number of version items to paginate by, or None for no pagination.

get_version_paginate_orphans()
Returns the maximum number of orphans extend the last page by when paginating.

get_version_paginator(*queryset*, *per_page*, *orphans=0*, *allow_empty_first_page=True*, ***kwargs*)
Return an instance of the version paginator for this view.

get_version_queryset()
Return the list of version items for this view.

The return value must be an iterable and may be an instance of *QuerySet* in which case *QuerySet* specific behavior will be enabled.

paginate_version_queryset(*queryset*, *page_size*)
Paginate the version queryset, if needed.

version_allow_empty = True

version_context_object_name = None

version_model
alias of `Version`

version_object_list = None

version_ordering = u'-revision__date_created'

version_page_kwarg = u'versionpage'

version_paginate_by = None

version_paginate_orphans = 0

version_paginator_class
alias of `Paginator`

version_queryset = None

class `reversion_extras.views.`**DetailVersionListView**(***kwargs*)
Bases: *reversion_extras.views.MultipleVersionObjectMixin*, `django.views.generic.detail.DetailView`

Render some list of django-reversion Versions of object, set by *self.model* or *self.queryset*. *self.queryset* can actually be any iterable of items, not just a queryset.

template_name_suffix = u'_version_list'

class `reversion_extras.views.`**UpdateVersionListView**(***kwargs*)
Bases: *reversion_extras.views.MultipleVersionObjectMixin*, `django.views.generic.edit.UpdateView`

Render some list of versions of object, set by *self.model* or *self.queryset*. *self.queryset* can actually be any iterable of items, not just a queryset.

template_name_suffix = u'_form_version_list'

7.1.5 Module contents

r

D

G

M

P

R

T

U

V