django-generic-filters Documentation

Release 0.11.dev0

Novapost

August 28, 2014

Contents

django-generic-filters is a toolkit to filter results of Django's `ListView`, using forms.

Main use cases are obviously search forms and filtered lists.

As a developer, given you have a `ListView`, in order to let the user filter the results:

- use a form to easily render the filters as HTML;
- the user typically sends the filters via GET;
- validate the user's input using a Django form;
- filter the Django view's queryset using form's cleaned data.

CHAPTER 1

Example

views.py

```
from django_genericfilters.views import FilteredListView


class UserListView(FilteredListView):
    # ListView options. FilteredListView inherits from ListView.
    model = User
    template_name = 'user/user_list.html'
    paginate_by = 10
    context_object_name = 'users'

    # FormMixin options. FilteredListView inherits from FormMixin.
    form_class = UserListForm

    # FilteredListView options.
    search_fields = ['first_name', 'last_name', 'username', 'email']
    filter_fields = ['is_active', 'is_staff', 'is_superuser']
    default_order = 'last_name'

    def form_valid(self, form):
        """Return the queryset when form has been submitted."""
        queryset = super(UserListView, self).form_valid(form)

        # Handle specific fields of the custom ListForm
        # Others are automatically handled by FilteredListView.

        if form.cleaned_data['is_active'] == 'yes':
            queryset = queryset.filter(is_active=True)
        elif form.cleaned_data['is_active'] == 'no':
            queryset = queryset.filter(is_active=False)

        if form.cleaned_data['is_staff'] == 'yes':
            queryset = queryset.filter(is_staff=True)
        elif form.cleaned_data['is_staff'] == 'no':
            queryset = queryset.filter(is_staff=False)

        if form.cleaned_data['is_superuser'] == 'yes':
            queryset = queryset.filter(is_superuser=True)
        elif form.cleaned_data['is_superuser'] == 'no':
            queryset = queryset.filter(is_superuser=False)

        return queryset
```

forms.py

```
from django import forms
from django.utils.translation import ugettext_lazy as _
from django_genericfilters import forms as gf


class UserListForm(gf.QueryFormMixin, gf.OrderFormMixin, gf.FilteredForm):
    is_active = gf.ChoiceField(label=_('Status'),
                               choices=(('yes', _('Active')),
                                        ('no', _('Unactive'))))

    is_staff = gf.ChoiceField(label=_('Staff'))

    is_superuser = gf.ChoiceField(label=_('Superuser'))

    def get_order_by_choices(self):
        return [('date_joined', _(u'date joined')),
                ('last_login', _(u'last login')),
                ('last_name', _(u'Name'))]
```

CHAPTER 2

Forms

Several form mixins are provided to cover frequent use cases:

- `OrderFormMixin` with order_by and order_reverse fields.
- `QueryFormMixin` for little full-text search using icontains.

See “mixin” documentation for details.

CHAPTER 3

Ressources

- Documentation: http://django-generic-filters.readthedocs.org
- PyPI page: http://pypi.python.org/pypi/django-generic-filters
- Code repository: https://github.com/novapost/django-generic-filters
- Bugtracker: https://github.com/novapost/django-generic-filters/issues
- Continuous integration: https://travis-ci.org/novapost/django-generic-filters

CHAPTER 4

Contents

4.1 Demo project

The `demo/` folder holds a demo project to illustrate django-generic-filters usage.

4.1.1 Browse demo code online

See **'demo folder in project's repository'_**.

4.1.2 Deploy the demo

System requirements:

- **'Python'_** version 2.6 or 2.7, available as `python` command.

  Note: You may use **'Virtualenv'_** to make sure the active `python` is the right one.

- `make` and `wget` to use the provided `Makefile`.

Execute:

```
git clone git@github.com:novapost/django-generic-filters.git
cd django-generic-filters/
python setup.py develop
cd demo/
python setup.py develop
```

Browse and use `demo/demoproject/` as a sandbox.

4.2 Installation

This project is open-source, published under BSD license. See *License* for details.

If you want to install a development environment, you should go to *Contributing to the project* documentation.

Install the package with your favorite Python installer. As an example, with pip:

```
pip install django-generic-filters
```

Don't forget to register your application in your Django `INSTALLED_APPS` setting.

4.3 Django Generic Filter API

4.3.1 FilteredListView API

FilteredListView is a generic ListView allowing to filter queryset. It supports RawQuerySet, Python Object List as well as normal queryset.

It use a form to define which filters you want and some options to control the way querysets must be managed.

Example

First we define a form to handle our filters

```
from django import forms
from django_genericfilters.forms import FilteredForm


class TicketListForm(FilteredForm):
    query = forms.CharField(label=_('Query'), required=False)

    priority = forms.ChoiceField(label=_('Priority'),
                                 required=False,
                                 choices=(
                                     ('0', 'High'),
                                     ('1', 'Normal'),
                                     ('2', 'Low')
                                 )
```

This form handle filter on a raw query and priority.

```
from django_genericfilters.views import FilteredListView
from myapp.models import Ticket


class TicketListView(FilteredListView):
    model = Ticket
    paginate_by = 10
    form_class = TicketListForm
    search_fields = ['creator__last_name', 'creator__first_name',
                     'subject', 'body']
    filter_fields = ['status', 'priority']
    qs_filter_fields = {'category__last_name': 'category',
                        'status': 'status',
                        'priority': 'priority'
                       }
    default_order = 'last_updated_at'
```

FilteredListView Options

TicketListView define a set of useful specific options:

search_fields

a list of fields to search against with the "query" field defined on the form (see above)

filter_fields

a list of fields used to construct the form filters in the template. In this example, we define status and priority. By default, if we do not define a qs_filter_fields it will be used to filter the resulting queryset as well.

qs_filter_fields

a dict used to filter the results queryset. Useful if you need to make complex filters or related models filters.

qs_filter_fields_conditions

A dict used to filter the results queryset. Useful to add extra condition for a special field from qs_filter_fields.

default_order

the order you want your results to be sorted by default.

FilteredListView Method

FilteredForm

4.4 About django-generic-filters

4.4.1 License

4.4.2 Authors & contributors

Maintainer: Rémy Hubscher <remy.hubscher@novapost.fr>

Original code by Novapost team:

- Benoît Bryon <benoit@marmelune.net>
- Rémy Hubscher <remy.hubscher@novapost.fr>
- Natal Ngétal <natal.ngetal@novapost.fr>
- Yohann Gabory <yohann.gabory@novapost.fr>

4.4.3 Changelog

0.11 (unreleased)

- Nothing changed yet.

0.10 (2014-08-19)

- Add a new attribute to manage condition.
- Reviwed readme.

0.9 (2014-07-18)

- Fix filter list pagination
- Fix travis-ci test suite

0.8 (2014-04-17)

- Remove support of the pagination.

0.7 (2014-02-25)

- Warning remove order_by_list which was merged with default_order
- Is form submitted return True when the request method is GET and have data in request.
- Adding form_empty method.

0.6.1 (2014-02-21)

- Missing files.

0.6 (2014-02-21)

- Fix order_by and order_reverse.
- Fix is_form_submitted method.
- Refactoring.

0.5 (2014-02-06)

- FilteredForm now use the mixins.
- Improve documentation
- Implement the get_qs_filter method
- FilterFields list view

0.4 (2013-10-01)

- Search for each terms instead of searching for the full string in order.
- Let use redefine the All value
- Allow html in choice labels
- Fix import path in documentation.

0.3 (2013-09-03)

- Added README.rst in the manifest file.

0.2 (2013-09-03)

- Fix AssertRaises to work with 2.6

0.1 (2013-09-02)

- Introduced generic mixin and FilteredView
- Introduced demo app
- Initialized project

4.5 Contributing to the project

This document provides guidelines for people who want to contribute to the project.

4.5.1 Create tickets

Please use the bugtracker [1] **before** starting some work:

- check if the bug or feature request has already been filed. It may have been answered too!
- else create a new ticket.
- if you plan to contribute, tell us, so that we are given an opportunity to give feedback as soon as possible.
- Then, in your commit messages, reference the ticket with some `refs #TICKET-ID` syntax.

4.5.2 Fork and branch

- Work in forks and branches.
- Prefix your branch with the ticket ID corresponding to the issue. As an example, if you are working on ticket #23 which is about contribute documentation, name your branch like `23-contribute-doc`.
- If you work in a development branch and want to refresh it with changes from master, please rebase [2] or merge-based rebase [3], i.e. don't merge master.

4.5.3 Setup a development environment

System requirements:

- Python [4] version 2.6 or 2.7, available as `python` command.

  Note: You may use Virtualenv [5] to make sure the active `python` is the right one.

- make and wget to use the provided `Makefile`.

Execute:

```
git clone git@github.com:novapost/django-generic-filters.git
cd django-generic-filters/
make develop
```

If you cannot execute the Makefile, read it and adapt the few commands it contains to your needs.

4.5.4 The Makefile

A `Makefile` is provided to ease development. Use it to:

- setup the development environment: `make develop`
- update it, as an example, after a pull: `make update`
- run tests: `make test`
- build documentation: `make documentation`

The `Makefile` is intended to be a live reference for the development environment.

[1] https://github.com/novapost/django-generic-filters/issues
[2] http://git-scm.com/book/en/Git-Branching-Rebasing
[3] http://tech.novapost.fr/psycho-rebasing-en.html
[4] http://python.org
[5] http://virtualenv.org

4.5.5 Documentation

Follow style guide for Sphinx-based documentations [6] when editing the documentation.

4.5.6 Test and build

Use the Makefile.

4.5.7 Demo project included

The *Demo project* is part of the tests. Maintain it along with code and documentation.

4.5.8 References

[6] http://documentation-style-guide-sphinx.readthedocs.org/