dj-extensions Documentation

Release 0.1.6

Jahan Balasubramaniam

February 07, 2016

Contents

Yet another Django extension with set of generic reusable, pluggable mixins

It has just started, there will be additions going forward.

Contributions are always welcome. Find the source code at GithubRepo See Contributions:

CHAPTER 1

Get started:

`dj-extensions` can be found in pypi repository and can be installed via pip

```
pip install dj-extensions
```

CHAPTER 2

Contents:

2.1 Installation

The extension is expected to work on all currently supported versions of django (>=1.7)

Stable releases can be installed from pypi(CheeseShop).

```
pip install dj-extensions
```

To install latest version(bleeding edge) install directly from Github repo

```
pip install --upgrade https://github.com/jahan01/dj-extensions/tree/master
```

2.2 View Mixins

These mixins provide additional fuunctionality to django class based views.

Currently following mixins are supported. This list will be expanding over the time. Kindly keep check on it.

Mixin names explain roughly what they do.

Contents

2.2.1 PermissionRequiredMixin

Checks for permission to access a view. If a user is unauthorised, request is redirected to login page.

Note: For versions >=1.9, django provides in-built permission check mixin.

Usage:

```
from django.views.generic import ListView
from dj_extensions.views import PermissionsRequiredMixin

class SomeView(PermissionsRequiredMixin, ListView):
    model                = YourModel
    required_permissions = ('app.permission1')
```

You can check for more than one permission too

```
from django.views.generic import ListView
from dj_extensions.views import PermissionsRequiredMixin

class SomeView(PermissionsRequiredMixin, ListView):
    model                = YourModel
    required_permissions = ('app.permission1',
                            'app.permission2',
                           )
```

The variable `required_permissions` is required.

Note: No need to check for do login required check if you are doing permissions checks. In either case if the check fails, the user will be redirected to login page

2.2.2 AjaxOnlyMixin

This mixin allows only the ajax requests and returns `HttpForbiddenError` if the request is not ajax.

Usage:

```
from django.views.generic import TemplateView
from dj_extensions.views import AjaxOnlyMixin

class SomeView(AjaxOnlyMixin, TemplateView):
    # your custom code
```

2.2.3 PaginationMixin

Google style paginations for your django list views.

This mixin provides list of links to previous and next `n_list` number of pages, in addition to just previous and next links provided by default.

Usage:

```
from django.views.generic import ListView
from dj_extensions.views import PaginationMixin

class SomeView(PaginationMixin, ListView):
    model                = YourModel
    paginate_by          = 10
    n_list               = 5
```

Default values are

```
paginate_by = 5
n_list      = 4
```

In your template for this view, add the following lines:

```
<nav>
  <ul class="pagination">
    {% if page_obj.has_previous %}
      <li><a href="?page={{ page_obj.previous_page_number }}" aria-label="Previous"><span aria-hidden
      {% for i in page_obj.paginator.page_range|slice:page_dict.prev %}
        <li><a href="?page={{ i }}">{{ i }}</a></li>
      {% endfor %}
    {% else %}
      <li><a href="javascript:;" aria-label="Previous"><span aria-hidden="true">&laquo;</span></a></l
    {% endif %}
    <li class="active"><a href="javascript:;"> {{ page_obj.number }} <span class="sr-only">(current)<
    {% if page_obj.has_next %}
      {% for i in page_obj.paginator.page_range|slice:page_dict.next %}
        <li><a href="?page={{ i }}">{{ i }}</a></li>
      {% endfor %}
      <li><a href="?page={{ page_obj.next_page_number }}"><span aria-hidden="true">&raquo;</span></a
    {% else %}
      <li><a href="javascript:;" aria-label="Next"><span aria-hidden="true">&raquo;</span></a></li>
    {% endif %}
  </ul>
</nav>
```

Note: This mixin only works with List views.

2.2.4 FilterMixin

This mixin is used filter your list view based on query strings from http requests

Usage:

```
from django.views.generic import ListView
from dj_extensions.views import FilterMixin

class SomeView(FilterMixin, ListView):
    model               = YourModel
    allowed_filters     = {
                           'name': 'emp_name__icontains',
                           'age' : 'age_exact',
                          }
```

The key of the `allowed_filters` dict is the query string and value is the django ORM filter opertation.

For example, the request `http://localhost:8000/some_view?name=foo&age=21` will perform

```
YourModel.objects.filter(emp_name_icontains='foo').filter(age_exact=21)
```

Note: This mixin only works with List views.

2.2.5 Combine Multiple Mixins

You can combine multiple mixins if required.

For example, there may be a use-case where you want your list view to be paginated, check for permissions and support filtering as well. For this case your view class will be:

```
from dj_extensions.views import PermissionsRequiredMixin, FilterMixin, PaginationMixin

class SomeView(PermissionsRequiredMixin, FilterMixin, PaginationMixin, ListView):
    model                = YourModel
    paginate_by          = 10
    n_list               = 5
    required_permissions = (
                            'app.permission1',
                            'app.permission2',
                           )
    allowed_filters      = {
                            'name': 'emp_name__icontains',
                            'age' : 'age_exact',
                           }
```

2.3 Contributions:

To contribute you don't necessarily need to code. Any one of the following is considered as a contribution.

- **Filing a bug**
  - If you bug at code/documentation
  - Please raise the issue at github repo issue page
- **Correcting documentation or documenting undocumented parts** Please find the docs folder in the repo
- **Request a new feature** Please be specific and tell the use case
- **Contibuting code for an issue or new feature** Please give a pull request
- **Contributing to testing** Please write tests and give us a pull request

CHAPTER 3

Indices and tables

- genindex
- modindex
- search