urljects Documentation

Release

Visgean Skeloru

December 26, 2016

Contents

CHAPTER 1

Installation

URLjects are supported on Python 2.7 and 3.4, the tests are also passing on pypy.

Easy installation:

```
$ easy_install urljects
```

PIP:

```
$ pip install urljects
```

You don't have to add urljects to INSTALLED_APPS

1.1 Testing

Run tests with:

```
$ python setup.py test
```

CHAPTER 2

Using URLjects

Under URLjects there are three main ways of urls. You should stick to one and avoid mixing them.

2.1 Using in urls.py

This is pretty standard way of dealing with routing, you put your urls into file called `urls.py` and register that file with django. Nice and tidy.

```
from urljects import url, U, slug
from views import main_view, ClassView

urlpatterns = ('',
    url(U / 'main', main_view),
    url(U / slug / 'classy', ClassView)
)
```

Note that you have to import `urljects.url` and not django version.

2.2 Using view_include

If you are tired of having so many `urls.py` files you can use `view_include` to directly import views to main urlconf. So your `urlconf` will look like this:

```
urlpatterns = [
    url(U / 'eshop', view_include(eshop_views)),
    url(U / 'blog', view_include(blog_views, namespace='named'))
]
```

Note that views have to either decorated with `url_view` function or for class based views they will have to inherit from `URLView` class.

Class based views:

```
class DetailView(View, URLView):
    url = U / 'detail' / slug
    url_name = 'detail_view'
    url_priority = -1
```

Function based views:

```
@url_view(U / 'detail' / slug, priority=-1)
def detail_view(request):
    return render()
```

If you don't specify url name for functional views it will be derived from function name.

2.3 Using the RouteMap

An alternative to `URLView` and `@url_view` is the RouteMap.

The RouteMap is an object that maps URLs to routes. Usually, one is created in every view module, and named "route". With class and callable views, it's used as a decorator:

```
from urljects import RouteMap
route = RouteMap()

@route(U / 'post')
def post_view():
    return render()

@route(U / 'detail', name='detail_view')
class DetailView(View):
    pass
```

It can also be used as a function, typically for view names as strings:

```
route(U / 'profile', 'users_app.views.profile_view')
```

In urls.py, use the RouteMap's `include` method:

```
from my_app.blog.views import route as blog_routemap
urlpatterns = [
    blog_routemap.include(U / 'blog')
]
```

The `@route` decorator may be used multiple times on a single view. The URLs it records are included in the order the views are defined in, or they can be given a priority:

```
@route(U / slug, priority=-1)
def post_view():
    """ A catch-all view with low priority """
    return render()
```

CHAPTER 3

U object

`U` object is `URLPattern` instance. It is used as replacement over regular expression. It is similar to .. _Pathlib.Path: https://docs.python.org/3/library/pathlib.html

You should combine U object with RE patterns, you can do it like this:

```
from urljects import U, slug, url, view_include

choice = r'(?P<choice>YES|NO)')

urlpatterns = [
    url(U / 'detail' / slug, views.DetailView),    # -> r'^detail/(?P<slug>[\w-]+)$'
    url(U / choice, views.ChoiceView)              # -> r'^(?P<choice>YES|NO)')'

    url(U / 'eshop', view_include(eshop_views))    # -> r'^eshop/' + included urls
]
```

CHAPTER 4

View name handling

If you don't specify name to `urljects.url` it will be derived automatically.

4.1 Functional views

```
url(U, view=views.test_view)
```

here `view.__name__` will be used.

4.2 String views

```
url(U / 'test', view='views.test_view')
```

the last part of the string will be used.

4.3 Class Views

```
url(U / 'class_view', view=views.ClassTestView)
```

`ClassTestView.url_name` will be used.

4.4 Namespace

So far there is no support for auto-guessing namespaces based on app.