Created and maintained by

Tom Christie

PyStok

TJ Software

Who We Are ?

Rafał Selewońko - TJ.Software Lead Developer

Seler ?

Tomasz Roszko - CEO / Python & Django Developer

He got his start as a web developer in 2007 working for “Netstation” and writing internet applications for one of the largest publishing companies in Poland. During this time he worked extensively with Django, Pylons, and the RED DOT CMS. In 2009 he started his own company creating a team of passionate Python/Django developers. He is currently the leader of this development team based in Bialystok Poland. He is also CTO of Opentopic and Learnicious. Previously he has worked with a number of industry leaders such as RevSquare, NowWeComply, BlueIce, FlowLabs and MediaPolis.

PyStok TJ.Software

Reasons you might want to use Django REST framework

- Web browseable API
- Out of the box authentication
- Serialization with ORM and non-ORM data
- Highly customizable
- Extensive documentation
- Used by large companies

Installation

console

```
# Install django-rest-framework
pip install djangorestframework

# Install additional libraries for drf
pip install markdown       # Markdown support for the browsable API.
pip install django-filter  # Filtering support
```

settings.py

```
INSTALLED_APPS = (
    ...
    # add rest framework app
    'rest_framework',
)

REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',
    )
}
```

Views and Viewsets

api.py

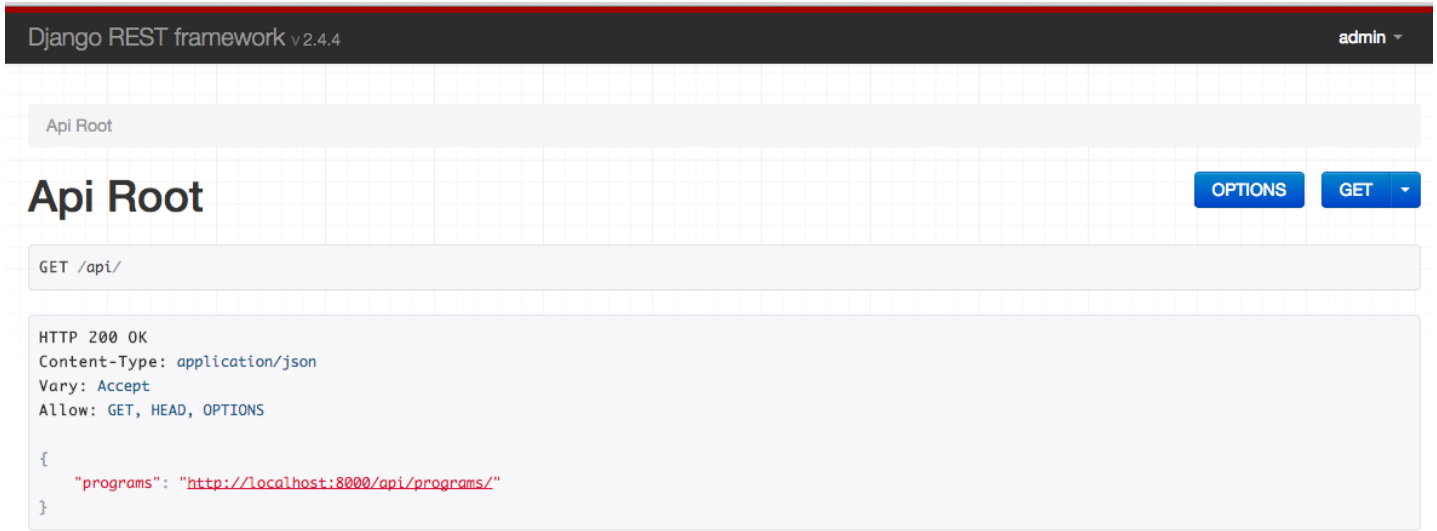
```python
from rest_framework import viewsets, routers
from .models import Program


class ProgramViewSet(viewsets.ModelViewSet):
    """
    Basic API view for programs
    """
    queryset = Program.objects.all()


# Routers provide an easy way of automatically
determining the URL conf.
router = routers.DefaultRouter()
router.register(r'programs', ProgramViewSet)
```

Urls

urls.py

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
from eduprograms.api import router

urlpatterns = patterns(
    '',
    url(r'^admin/', include(admin.site.urls)),

    # include api links
    url(r'^api/', include(router.urls)),
    url(r'^api-auth/', include('rest_framework.urls',

namespace='rest_framework'))
)
```

How our api looks like ?



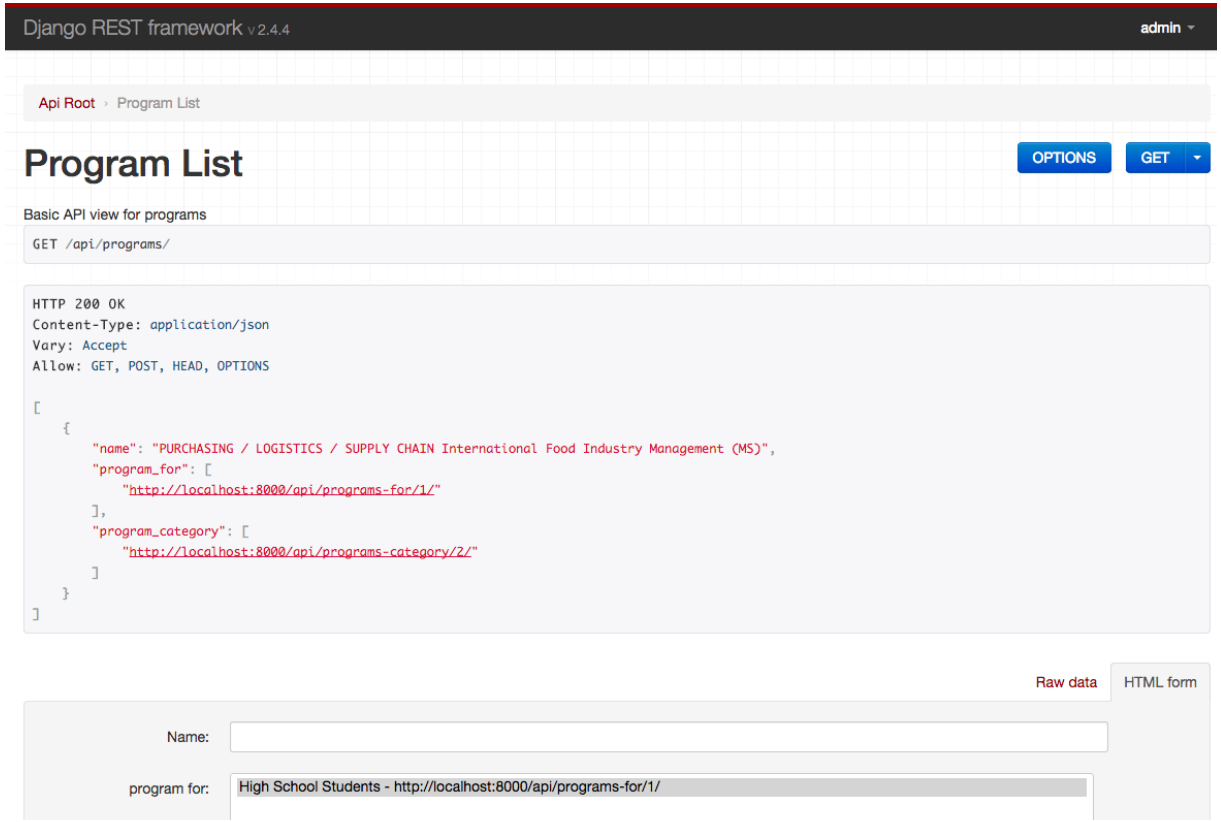
PyStok TJ Software

Serializers

```
from rest_framework import viewsets, routers, serializers
from .models import Program, ProgramFor, ProgramCategory


# Serializers define the API representation.
class ProgramSerializer(serializers.
HyperlinkedModelSerializer):
    class Meta:
        model = Program
        fields = ('name', 'program_for', 'program_category')


class ProgramViewSet(viewsets.ModelViewSet):
    """
    Basic API view for programs
    """
    queryset = Program.objects.all()
    serializer_class = ProgramSerializer
```

```
class ProgramForViewSet(viewsets.ModelViewSet):
    """
    Basic API view for ProgramFor
    """
    queryset = ProgramFor.objects.all()


class ProgramCategoryViewSet(viewsets.ModelViewSet):
    """
    Basic API view for ProgramFor
    """
    queryset = ProgramCategory.objects.all()


# Routers provide an easy way of automatically determining the URL conf.
router = routers.DefaultRouter()
router.register(r'programs', ProgramViewSet)
router.register(r'programs-for', ProgramForViewSet)
router.register(r'programs-category', ProgramCategoryViewSet)
```

Works :)



PyStok

TJ Software

Display related data

```
class ProgramChapterSerializer(serializers.HyperlinkedModelSerializer):

    class Meta:
        model = ProgramChapter


class ProgramForSerializer(serializers.HyperlinkedModelSerializer):

    class Meta:
        model = ProgramFor


class ProgramCategorySerializer(serializers.HyperlinkedModelSerializer):

    class Meta:
        model = ProgramCategory


# Serializers define the API representation.
class ProgramSerializer(serializers.HyperlinkedModelSerializer):

    program_for = ProgramForSerializer(many=True)
    program_category = ProgramCategorySerializer(many=True)
    programchapter_set = ProgramChapterSerializer(many=True)

    class Meta:
        model = Program
        fields = ('name', 'program_for', 'program_category',
                  'programchapter_set')
```

Hmmmm Looks good :)

Django REST framework v2.4.4 admin

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

[
    {
        "name": "PURCHASING / LOGISTICS / SUPPLY CHAIN International Food Industry Management (MS)",
        "program_for": [
            {
                "url": "http://localhost:8000/api/programs-for/1/",
                "name": "High School Students"
            }
        ],
        "program_category": [
            {
                "url": "http://localhost:8000/api/programs-category/2/",
                "name": "Advanced Masters MS/MSC\u2019S"
            }
        ],
        "programchapter_set": [
            {
                "url": "http://localhost:8000/api/programs-chapter/1/",
                "title": "Chapter 1",
                "priority": 0,
                "program": "http://localhost:8000/api/programs/1/",
                "subtitle_1": "SubTtitle",
                "subtext_1": "SubText1"
            },
            {
                "url": "http://localhost:8000/api/programs-chapter/2/",
                "title": "Chapter 2",
                "priority": 0,
                "program": "http://localhost:8000/api/programs/1/",
                "subtitle_1": "SubTitle 2",
                "subtext_1": "Sub Text"
            }
        ]
    }
]
```

Authentication

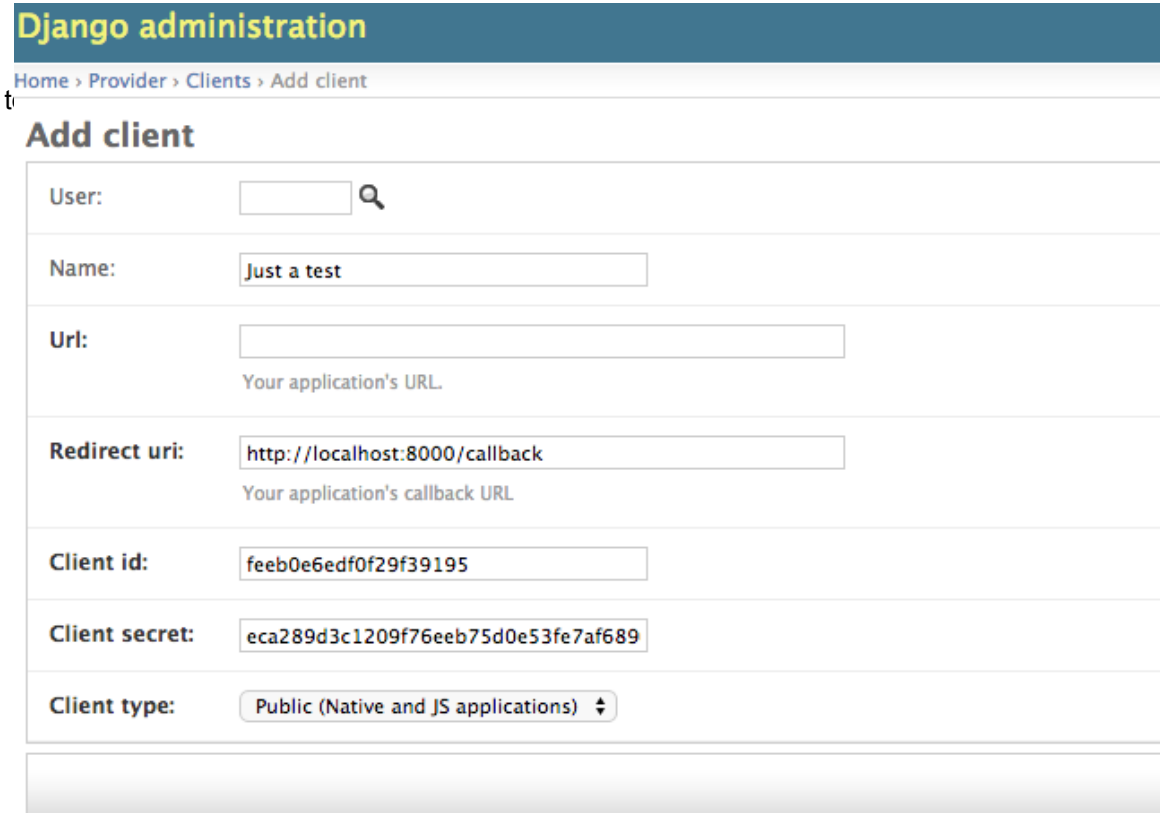
- BasicAuthentication
- TokenAuthentication
- SessionAuthentication
- OAuthAuthentication
- OAuth2Authentication

Auth Settings

```
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.BasicAuthentication',  # Basic HTTP Auth, realm="api"
        'rest_framework.authentication.SessionAuthentication',  # For Browseable API
        'rest_framework.authentication.OAuth2Authentication',  # Standard OAuth2
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',
    )
}
```

OAuth2

```
# Install django-oauth2-provider
pip install django-oauth2-provider # !!!! do not work
# with django 1.7.x (waiting for merge pull requests t
# master)
INSTALLED_APPS = (
    ...
    'provider',
    'provider.oauth2',
`
from django.conf.urls import patterns, include, url
from django.contrib import admin
from eduprograms.api import router

urlpatterns = patterns(
    '',
    url(r'^admin/', include(admin.site.urls)),

    # include api links
    url(r'^api/', include(router.urls)),
    url(r'^api-auth/', include('rest_framework.urls',

namespace='rest_framework')),
    url(r'^auth2/', include('provider.oauth2.urls',
                        namespace=oauth2))
)
```



PyStok TJ Software

Works

$ curl -X POST -d "client_id=feeb0e6edf0f29f39195&client_secret=eca289d3c1209f76eeb75d0e53fe7af689676427&grant_type=password&username=admin&password=admin" http://localhost:8000/oauth2/access_token/

{"access_token": "c6bfba6b3d2fea1e7be022a43efca8bdd02848f4", "token_type": "Bearer", "expires_in": 31535999, "refresh_token": "a032e7ce934aecfd26a3c1e922e8c161598eaa70", "scope": "read"}(ibm)

$ curl -H "Authorization: Bearer c6bfba6b3d2fea1e7be022a43efca8bdd02848f4" http://localhost:8000/api/

{"programs": "http://localhost:8000/api/programs/", "programs-for": "http://localhost:8000/api/programs-for/", "programs-category": "http://localhost:8000/api/programs-category/", "programs-chapter": "http://localhost:8000/api/programs-chapter/"}

Works

$ curl -X OPTIONS -u admin:admin http://localhost:8000/api/programs-category/

{"name": "Program Category List", "description": "Basic API view for ProgramFor", "renders": ["application/json", "text/html"], "parses": ["application/json", "application/x-www-form-urlencoded", "multipart/form-data"], "actions": {"POST": {"url": {"type": "field", "required": false, "read_only": true}, "name": {"type": "string", "required": true, "read_only": false, "label": "Name", "max_length": 255}}}}

$ curl -H "Authorization: Bearer c6bfba6b3d2fea1e7be022a43efca8bdd02848f4" -X POST -d 'name=test' http://localhost:8000/api/programs-category/

{"url": "http://localhost:8000/api/programs-category/1/", "name": "test"}

$ curl -H "Authorization: Bearer c6bfba6b3d2fea1e7be022a43efca8bdd02848f4" http://localhost:8000/api/programs-category/1/

{"url": "http://localhost:8000/api/programs-category/1/", "name": "test"}

$ curl -H "Authorization: Bearer c6bfba6b3d2fea1e7be022a43efca8bdd02848f4" http://localhost:8000/api/programs-category/

[{"url": "http://localhost:8000/api/programs-category/1/", "name": "test"}]

ViewsSets and **@list_route** and **@detail_route** decorators

- /api/playlists/my/
- /api/playlists/public/
- /api/playlists/<id>/
- /api/playlists/<id>/tracks/

```
class PlaylistViewsSet(ModelViewSet):
    queryset = Playlist.objects.all()
    serializer_class = PlaylistSerializer

    @list_route
    def my(self, request):
        self.queryset = self.queryset.filter(user=request.user)
        return self.list(request)

    @list_route
    def public(self, request):
        self.queryset = self.queryset.filter(public=True)
        return self.list(request)

    @detail_route
    def tracks(self, request):
        playlist = self.get_object()
        tracks = playlist.track_set.objects.all()
        serializer = TrackSerializer(tracks)
        return Response(serializer.data)
```

More and more....

```
# it would be good to have diffrent image served for
# mobile / desktop
photo = serializers.SerializerMethodField('get_photo')

# get field value directly from object
expired =.serializers.Field('has_expired')

def get_photo(self, obj):
    request = self.context['request']._request
    MobileDetectionMiddleware.process_request(request)
    return get_image_path(obj.photo, request.mobile)
```

From Documentation: Generic View (http://www.django-rest-framework.org/api-guide/generic-views)

```
class UserList(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = (IsAdminUser,)
```

```
    def get_paginate_by(self):
        """
        Use smaller pagination for HTML representations.
        """
        if self.request.accepted_renderer.format == 'html':
            return 20
        return 100
```

```
    def list(self, request):
        # Note the use of `get_queryset()` instead of `self.queryset`
        queryset = self.get_queryset()
        serializer = UserSerializer(queryset, many=True)
        return Response(serializer.data)
```

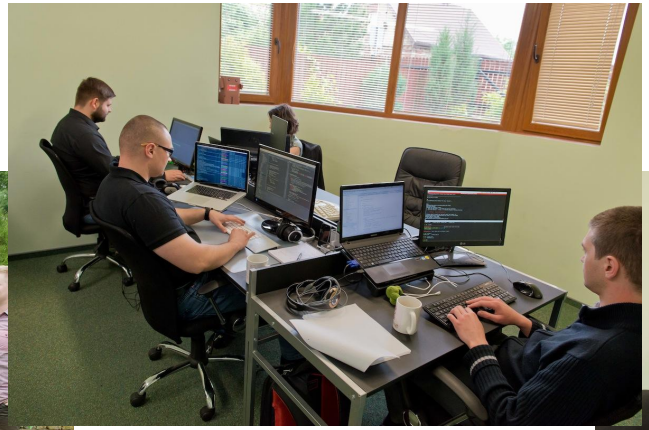
Django Rest Frameworks

- http://www.django-rest-framework.org/
- https://django-tastypie.readthedocs.org/en/latest/
- https://bitbucket.org/jespern/django-piston/wiki/Home

Presentation APP

https://github.com/tjsoftware-co/pystok2-drf

www.django-rest-framework.org



Work with US !!!

PyStok

TJ Software

Play with US !!! :)

PyStok

TJ Software