一、简介

官网地址： https://www.django-rest-framework.org/

> Django Rest Framework是一个开源的，由**合作资助的项目**。如果在商业上使用REST框架，建议**注册付费计划**，使之可以持续开发，健康发展。

Django Rest Framework是一个强大且灵活的工具包，主要用以构建RESTful风格的Web API。

Django REST Framework（以后简称DRF）可以在Django的基础上迅速实现API，并且自身还带有基于WEB的测试和浏览页面，可以方便的测试自己的API。DRF几乎是Django生态中进行前后端分离开发的默认库。

Django REST Framework具有以下功能和特性：

- 自带基于Web的可浏览的API，对于开发者非常有帮助
- 支持OAuth1a 和OAuth2认证策略
- 支持ORM或非ORM数据源的序列化
- 高可自定制性，多种视图类型可选
- 自动生成符合 RESTful 规范的 API
- 支持 OPTION、HEAD、POST、GET、PATCH、PUT、DELETE等HTTP方法
- 根据 Content-Type 来动态的返回数据类型（如HTML、json）
- 细粒度的权限管理（可到对象级别）
- 丰富的文档和强大的社区支持
- Mozilla、Red Hat、 Heroku和Eventbrite等知名公司正在使用

二、安装依赖

当前时间2019年5月，DRF版本3.9.2，依赖及支持如下:

- Python (2.7, 3.4, 3.5, 3.6, 3.7)
- Django (1.11, 2.0, 2.1, 2.2)

通常我们都是使用最新版本的Python和Django，比如当下的Django2.2和Python3.7.3。

以下软件包是可选的:

- coreapi （1.32.0+） - 支持模式生成和coreapi命令行工具。
- Markdown （2.1.0+） - 可浏览API的Markdown支持。
- django-filter （1.0.1+） - 过滤支持。
- django-crispy-forms - 改进的HTML显示过滤。
- django-guardian （1.1.1+） - 对象级别的权限支持。

三、安装方法

直接使用pip就可以安装:

```
pip install djangorestframework
pip install markdown       # Markdown
pip install django-filter    # 可选
pip install coreapi            # 可选
```

或者通过git下载:

```
git clone https://github.com/encode/django-rest-framework

```

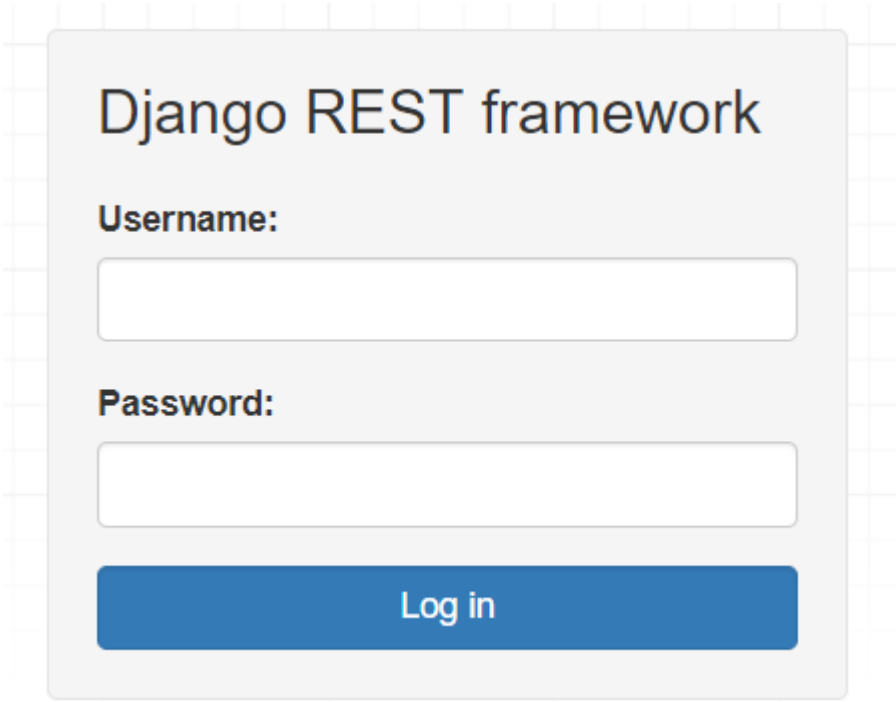
安装完毕，在项目配置文件中，注册app:

```
INSTALLED_APPS = (
    ...
    'rest_framework',
)
```

如果你想使用基于浏览器的可视化的API目录，并且希望获得一个登录登出功能，那么可以在根路由下添加下面的路由，这个功能类似Django自带的admin后台:

```
urlpatterns = [
    ...
    path('api-auth/', include('rest_framework.urls'))
]
```

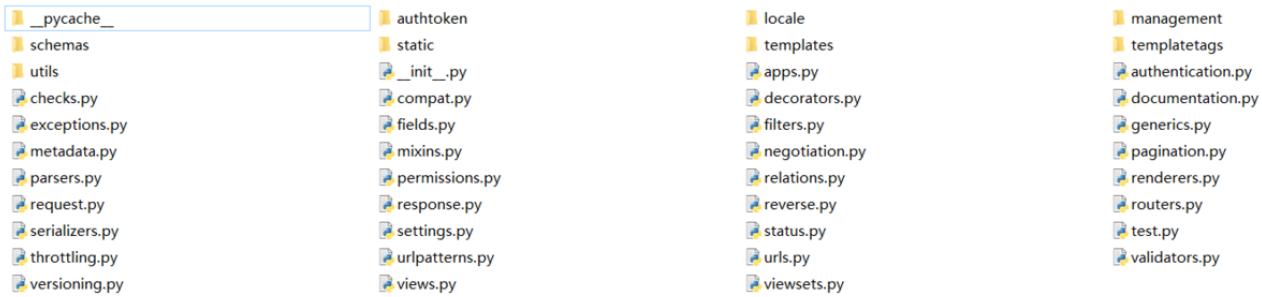
`'api-auth/'` 可以随意指定。



为了登录操作，也许你还要生成数据表，创建超级用户。这个步骤可选：

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

在Windows中，DRF的目录结构如下：



四、简单的使用

在项目的根路由urls.py文件中，写入下面的代码：

```
from django.urls import path, include
from django.contrib.auth.models import User
```

```python
from rest_framework import routers, serializers, viewsets

# Serializers define the API representation.
class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'email', 'is_staff')

# ViewSets define the view behavior.
class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer

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

# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls',
namespace='rest_framework'))
]
```

这个简单，又高度集成的例子，实际上是把DRF的序列化器、视图集、路由类和路由表都写在一个文件里了。真实情况下肯定不能这么做。

启动开发服务器，访问 `127.0.0.1:8000/users/` ，查看效果吧。