There are 5 total steps to have our REST API fully working:

Install and setup Django and DRF

Set Up Django Models

Set Up DRF Serializers

Set Up Views and URLs

Start Using the API!

Install and Set Up Django and DRF

Let’s start by creating a Python virtual environment and activating it in the desired working directory. For this example I used pyenv to manage my virtual environment.

python -m venv LMS

After the virtual environment is created and activated,

let’s install django and djangorestframework which are the

necessary Python libraries.

(LMS) $ pip3 install django

Successfully installed asgiref-3.3.1 django-3.1.5 pytz-

2020.5 sqlparse-0.4.1

(LMS) $ pip3 install djangorestframework

Successfully installed djangorestframework-3.12.2

After installing the necessary requirements let’s create a

Django project and app.

(LMS) $ django-admin startproject teaching\_blog

(LMS) $ cd my\_awesome\_django\_project

(LMS) $ django-admin startapp app\_users

(LMS) $ django-admin startapp app\_users

(LMS) $ ls

After the app is created, let’s register the app by adding

the path to the app config in the teaching\_blog/settings.py.

We should also add rest\_api\_framework to this list.

INSTALLED\_APPS = [

...

‘rest\_framework’,

'app\_users'

'curriculam'

]

Let’s make sure the app is up and running by using the

Django runserver command.

(LMS) $ python3 manage.py runserver

(0 silenced).

July 25, 2022 - 12:11:43

Django version 3.2, using settings 'teaching\_blog.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

Set Up Django Models

At this point we will be mostly editing the files in the

app\_users directory. Let’s start by adding Person and

Species into app\_users/models.py.

from django.db import models

class Contact(models.Model):

name = models.CharField(max\_length=150)

email = models.CharField(max\_length=150)

feedback = models.TextField()

def \_\_str\_\_(self):

return self.name

def get\_absolute\_url(self):

return reverse('index')

After the models have been added, let’s run the migrations

to let Django know that we are going to add table to the

database.

The first step is to create a new migration by running the

makemigrations command.

(LMS) $ python manage.py makemigrations

After the migrations files are created, we can run the

migrate command.

(LMS) $ python manage.py migrate

Set Up DRF Serializers

Now that we have added the models and created the tables, it

is time to tell DRF how to serialize the models. The

serializers will convert the Person model and Species model

into JSON that will be used by the API to return the data to

the user. We will add the serializers by creating a new file

app\_users/serializers.py.

from rest\_framework import serializers

from .models import Contact

class ContactSerializers(serializers.ModelSerializer):

class Meta:

model = Contact

fields = ['name', 'email', 'feedback']

Set Up Routers and Create API URLs

After the serializers are created we need to create a view

to the API and connect it to the Django URLs. Let’s start by

adding 2 viewsets for each of the models we created in a new

file app\_users/views.py. Viewsets provide the advantage of

combining multiple sets of logic into a single class.

from .serializers import ContactSerializers

from .models import \*

from rest\_framework import viewsets

class ContactViewSet(viewsets.ModelViewSet):

queryset = Contact.objects.all()

serializer\_class = ContactSerializers

After the viewsets are defined we can now use the router

functionality provided by DRF to route a desired API

endpoint to the given viewset. Let’s create a new file

app\_users/urls.py and add the router configuration as shown

below.

from django.urls import include, path

from rest\_framework import routers

from .views import ContactSerializers, ContactViewSet

router = routers.DefaultRouter()

router.register(r'contact',ContactViewSet)

urlpatterns = [

path('school/',include(router.urls)),

]

Finally let’s connect the main Django URL at

teaching\_blog/urls.py to point to the app’s URL file.

from django.urls import path,include

from django.conf.urls import include

import app\_users

import curriculum

urlpatterns = [

path('', include('app\_users.urls')),

path('curriculum/',include('curriculum.urls')),

path('admin/', admin.site.urls),

]

The URLs generated by the routers will have all the desired

API methods (GET, POST and PUT) we want. This is because the

router automatically maps the URLs to the right handlers

.get(), .list(), and .create() in the viewset. Let’s test

out the API and make sure everything is working as expected.

Start Using the API

If we go to the http://127.0.0.1:8000/school/?format=api URL

now, we should be able to use the browsable api to post

Contact Detail and Feedbck.

