



CS548
Web Services Techniques and REST Technologies

Module 5

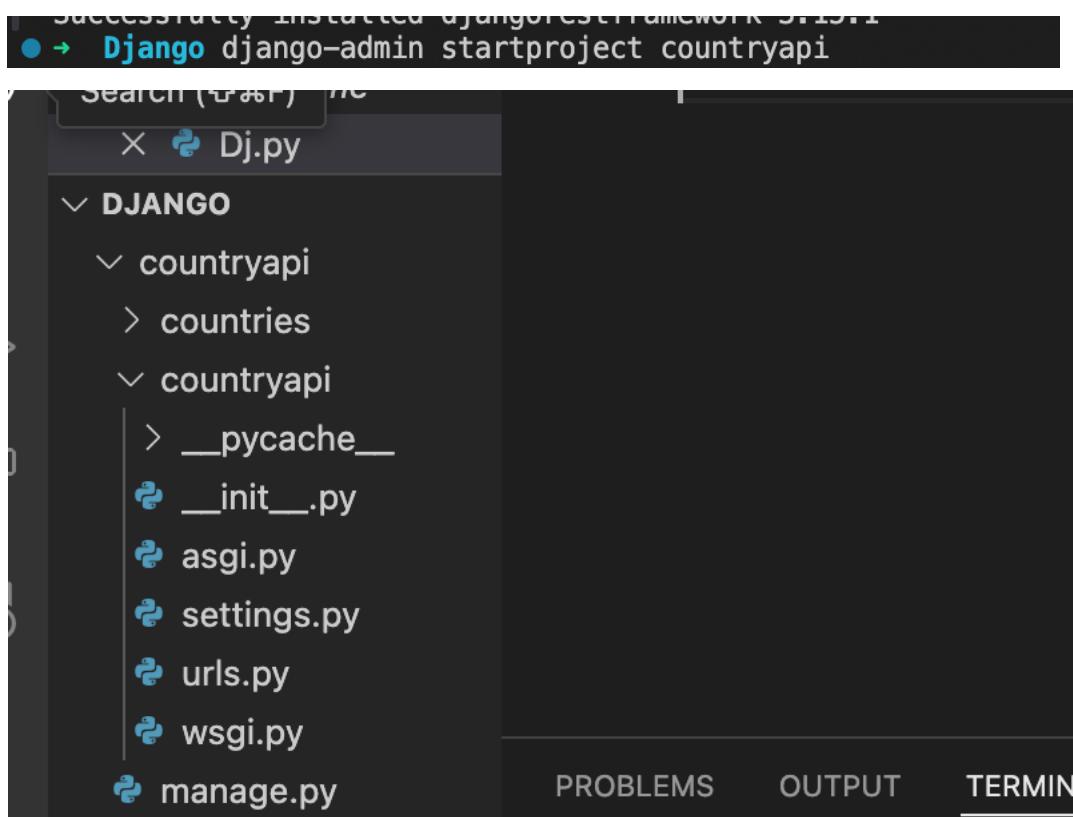
Belsabel Woldemichael

ID - 20052

1. Build a Django project and add in Django REST framework.

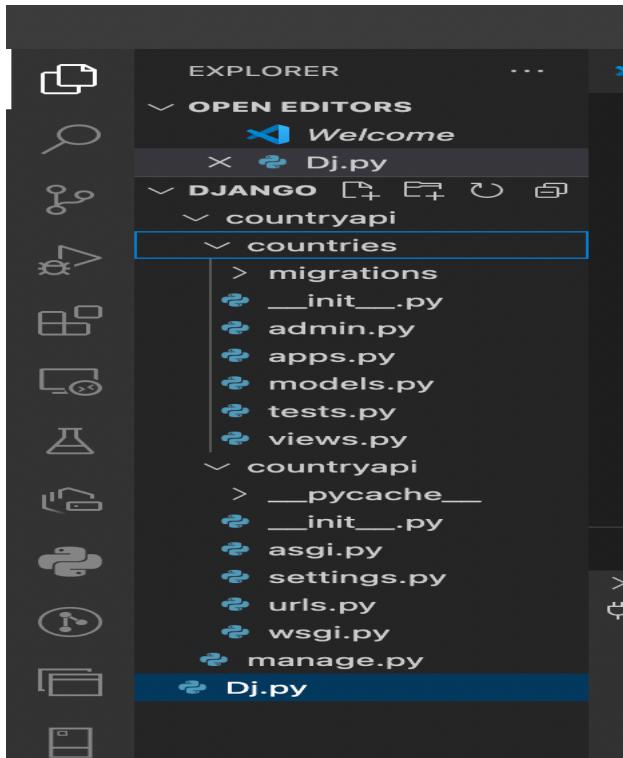
```
Requirement already satisfied: pyctz in /System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python (from django) (2015.7)
• -> Django python3 -m pip install Django djangorestframework
Requirement already satisfied: Django in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (5.0.6)
Collecting djangorestframework
  Downloading djangorestframework-3.15.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: asgiref<4,>=3.7.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from Django) (3.8.1)
Requirement already satisfied: sqlparse>=0.3.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from Django) (0.5.0)
Requirement already satisfied: djangorestframework-3.15.1-py3-none-any.whl (1.1 MB)
  1.1/1.1 MB 10.7 MB/s eta 0:00:00
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.15.1
• -> Django
```

2. **Create the Django Project:** Make sure you are in the desired directory where you want to create your Django project. Run:



3. **Navigate to the Project Directory:** Change to the directory where manage.py is located:
4. **Create the Django App:** Run the following command within the countryapi directory:

```
directory
• -> Django cd countryapi
• -> countryapi python3 manage.py startapp countries
```



5. Open up the settings.py file that's inside the countryapi folder. Add the last two lines to INSTALLED_APPS to tell Django about the countries application and Django REST framework:

```
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     "rest_framework",
41     "countries",
42 ]
```

6. Inside of the countries application, update models.py with the following code:

```
countryapi > countries > models.py > Country
1  from django.db import models
2
3  # Create your models here.
4  class Country(models.Model):
5      name = models.CharField(max_length=100)
6      capital = models.CharField(max_length=100)
7      area = models.IntegerField(help_text="(in square kilometers)")
```

7. Run the following commands to have Django update the database based on this model:

```
● → countryapi python3 manage.py startapp countries
● → countryapi python3 manage.py makemigrations
  Migrations for 'countries':
    countries/migrations/0001_initial.py
      - Create model Country
● → countryapi python3 manage.py migrate
  Operations to perform:
    Apply all migrations: admin, auth, contenttypes, countries, sessions
  Running migrations:
    Applying contenttypes.0001_initial... OK
    Applying auth.0001_initial... OK
    Applying admin.0001_initial... OK
    Applying admin.0002_logentry_remove_auto_add... OK
    Applying admin.0003_logentry_add_action_flag_choices... OK
    Applying contenttypes.0002_remove_content_type_name... OK
    Applying auth.0002_alter_permission_name_max_length... OK
    Applying auth.0003_alter_user_email_max_length... OK
    Applying auth.0004_alter_user_username_opts... OK
    Applying auth.0005_alter_user_last_login_null... OK
    Applying auth.0006_require_contenttypes_0002... OK
    Applying auth.0007_alter_validators_add_error_messages... OK
    Applying auth.0008_alter_user_username_max_length... OK
    Applying auth.0009_alter_user_last_name_max_length... OK
    Applying auth.0010_alter_group_name_max_length... OK
    Applying auth.0011_update_proxy_permissions... OK
    Applying auth.0012_alter_user_first_name_max_length... OK
    Applying countries.0001_initial... OK
    Applying sessions.0001_initial... OK
```

8. Copy and save the following JSON data into a file called countries.json and save it inside the countries directory:

The screenshot shows a code editor interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Explorer:** Shows the project structure:
 - OPEN EDIT... (highlighted)
 - DJ.py
 - settings.py
 - models.py
 - countries.json (highlighted)
 - serializers.py
 - views.py
 - urls.py
 - urls.py
- Django:** Shows the countryapi application structure:
 - countryapi
 - countries
 - __pycache__
 - __init__.cpython-31...
 - admin.cpython-3...
 - apps.cpython-3...
 - models.cpython-3...
 - serializers.cpython-3...
 - urls.cpython-31...
 - views.cpython-3...
 - fixtures
 - countries.json (highlighted)
 - migrations
 - __pycache__
 - __init__.py
 - 0001_initial.py
 - __init__.py
 - admin.py
- Editor:** Shows the contents of countries.json:

```
1  [
2   {
3     "model": "countries.country",
4     "pk": 1,
5     "fields": {
6       "name": "Thailand",
7       "capital": "Bangkok",
8       "area": 513120
9     }
10  },
11  {
12    "model": "countries.country",
13    "pk": 2,
14    "fields": {
15      "name": "Australia",
16      "capital": "Canberra",
17      "area": 7617930
18    }
19  },
20  {
21    "model": "countries.country",
22    "pk": 3,
23    "fields": {
24      "name": "Egypt",
25      "capital": "Cairo",
26      "area": 1010408
27    }
28  }
29]
```

9. Call the following command to load this data in the database:

```
cd ..
● → countryapi python3 manage.py loaddata countries
Installed 3 object(s) from 1 fixture(s)
```

10. Start by creating a file called serializers.py inside of the countries application

The screenshot shows a code editor interface with a dark theme. The menu bar includes Apple, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, and Help. The title bar says "serializers.py — Django". The left sidebar is titled "OPEN EDITORS" and lists files: DJ.py, settings.py, models.py, countries.json, and serializers.py. The "DJANGO" section shows the "countryapi" and "countries" applications, along with their __pycache__ directory and various Python files like __init__.py, admin.py, and models.py. The main editor area displays the following code:

```
countryapi > countries > serializers.py > CountrySerializer > Meta
1 # countries/serializers.py
2 from rest_framework import serializers
3 from .models import Country
4
5 class CountrySerializer(serializers.ModelSerializer):
6     class Meta:
7         model = Country
8         fields = ["id", "name", "capital", "area"]
```

11. Below is the code for a ModelViewSet subclass called CountryViewSet. This class will generate the views needed to manage Country data. Add the following code to views.py inside the countries application:

The screenshot shows a code editor interface with a dark theme. The menu bar includes Apple, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, and Help. The title bar says "views.py — Django". The left sidebar is titled "OPEN EDITORS" and lists files: DJ.py, settings.py, models.py, countries.json, serializers.py, and views.py. The "DJANGO" section shows the "countryapi" and "countries" applications, along with their __pycache__ directory. The main editor area displays the following code:

```
countryapi > countries > views.py > CountryViewSet
1 # countries/views.py
2 from rest_framework import viewsets
3
4 from .models import Country
5 from .serializers import CountrySerializer
6
7 class CountryViewSet(viewsets.ModelViewSet):
8     serializer_class = CountrySerializer
9     queryset = Country.objects.all()
```

12. Create a urls.py file in the countries application and add the following code to the file:

The screenshot shows a code editor interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Toolbar:** Standard Mac OS X style toolbar with red, green, and blue buttons.
- Explorer:** Shows the project structure:
 - OPEN EDITORS:** DJ.py, settings.py, models.py, countries.json, serializers.py, views.py, urls.py (highlighted).
 - DJANGO:** countryapi, countries, __pycache__, __init__.cpython-38, admin.cpython-38.
- Code Editor:** The 'urls.py' file content is displayed:

```
1 # countries/urls.py
2 from django.urls import path, include
3 from rest_framework.routers import DefaultRouter
4
5 from .views import CountryViewSet
6
7 router = DefaultRouter()
8 router.register(r"countries", CountryViewSet)
9
10 urlpatterns = [
11     path("", include(router.urls))
12 ]
```

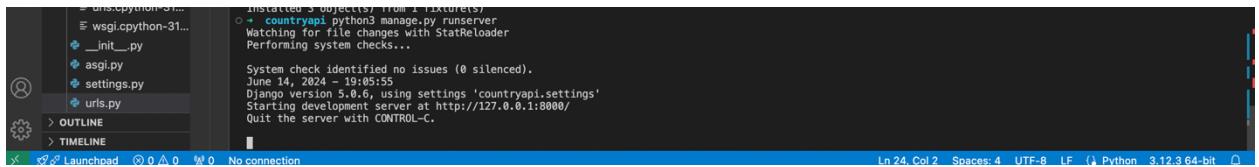
13. Finally, you need to update the project's base urls.py file to include all the countries URLs in the project. Update the urls.py file inside of the countryapi folder with the following code:

The screenshot shows a code editor interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Toolbar:** Standard Mac OS X style toolbar with red, green, and blue buttons.
- Explorer:** Shows the project structure:
 - OPEN EDITORS:** DJ.py, settings.py, models.py, countries.json, serializers.py, urls.py (highlighted).
 - DJANGO:** countryapi, countries, migrations, 0001_initial.py, __init__.py, admin.py, apps.py, models.py, serializers.py, tests.py, urls.py, views.py.
- Code Editor:** The 'urls.py' file content is displayed:

```
1 """
2 URL configuration for countryapi project.
3
4 The `urlpatterns` list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/5.0/topics/http/urls/
6 Examples:
7 Function views
8     1. Add an import: from my_app import views
9     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 # countryapi/urls.py
18 from django.contrib import admin
19 from django.urls import path, include
20
21 urlpatterns = [
22     path("admin/", admin.site.urls),
23     path("", include("countries.urls")),
24 ]
```

14. Run the following command in the root countryapi directory to start the Django development server:



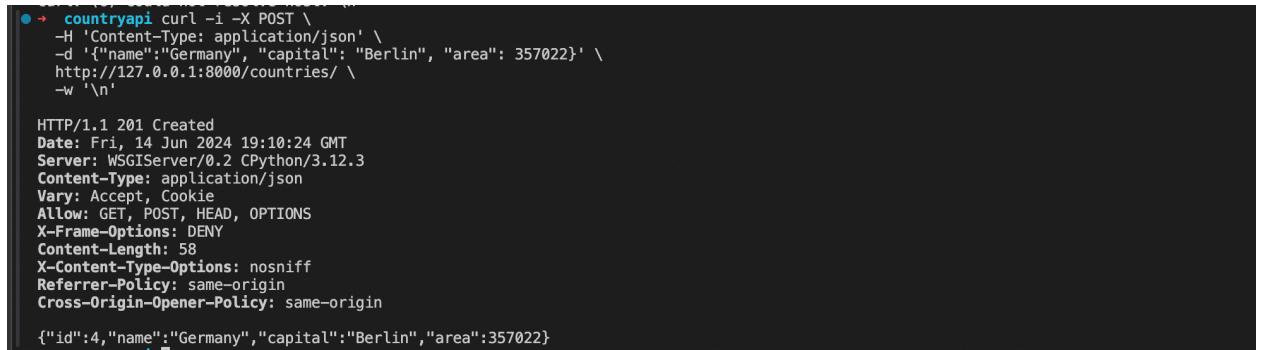
```
= drscipython-31...  
= wsgicpython-31...  
+ __init__.py  
+ asgi.py  
+ settings.py  
+ urls.py  
> OUTLINE  
> TIMELINE  
INSPECTED 0 DIRECTORIES & 1 FIXTURES  
o - countryapi python3 manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
System check identified no issues (0 silenced).  
June 14, 2024 - 19:05:55  
Django version 5.0.6, using settings 'countryapi.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CONTROL-C.
```

15. The development server is now running. Go ahead and send a GET request to /countries/ to get a list of all the countries in your Django project:



```
PROBLEMS OUTPUT TERMINAL  
> > ✎ TERMINAL  
↳ ✎ ● → Django cd countryapi  
● → countryapi curl -i http://127.0.0.1:8000/countries/ -w '\n'  
HTTP/1.1 200 OK  
Date: Fri, 14 Jun 2024 19:06:39 GMT  
Server: WSGIServer/0.2 CPython/3.12.3  
Content-Type: application/json  
Vary: Accept, Cookie  
Allow: GET, POST, HEAD, OPTIONS  
X-Frame-Options: DENY  
Content-Length: 183  
X-Content-Type-Options: nosniff  
Referer-Policy: same-origin  
Cross-Origin-Opener-Policy: same-origin  
[{"id":1,"name":"Thailand","capital":"Bangkok","area":513120},{"id":2,"name":"Australia","capital":"Canberra","area":7617930}, {"id":3,"name":"Egypt","capital":"Cairo","area":1010408}]  
o → countryapi
```

16. Send a POST request to /countries/ to create a new Country in your Django project:



```
● → countryapi curl -i -X POST \  
-H 'Content-Type: application/json' \  
-d '{"name":"Germany", "capital": "Berlin", "area": 357022}' \  
http://127.0.0.1:8000/countries/ \  
-w '\n'  
HTTP/1.1 201 Created  
Date: Fri, 14 Jun 2024 19:10:24 GMT  
Server: WSGIServer/0.2 CPython/3.12.3  
Content-Type: application/json  
Vary: Accept, Cookie  
Allow: GET, POST, HEAD, OPTIONS  
X-Frame-Options: DENY  
Content-Length: 58  
X-Content-Type-Options: nosniff  
Referer-Policy: same-origin  
Cross-Origin-Opener-Policy: same-origin  
{"id":4,"name":"Germany","capital":"Berlin","area":357022}
```

17. Run the following command to get the first Country:

```

● → countryapi curl -i http://127.0.0.1:8000/countries/1/ -w '\n'
HTTP/1.1 200 OK
Date: Fri, 14 Jun 2024 19:09:25 GMT
Server: WSGI Server/0.2 CPython/3.12.3
Content-Type: application/json
Vary: Accept, Cookie
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
X-Frame-Options: DENY
Content-Length: 60
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

{"id":1,"name":"Thailand","capital":"Bangkok","area":513120}
● → countryapi

```

18. Get Request by using Postman.

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:8000/countries/`. The response body is a JSON array containing four country objects:

```

[{"id": 1, "name": "Thailand", "capital": "Bangkok", "area": 513120}, {"id": 2, "name": "Australia", "capital": "Canberra", "area": 7617930}, {"id": 3, "name": "Egypt", "capital": "Cairo", "area": 1010408}, {"id": 4, "name": "Germany", "capital": "Berlin", "area": 357022}]

```

PART 1.B

Create client calls to the server you created, as we did in class.

1. POST a New Country

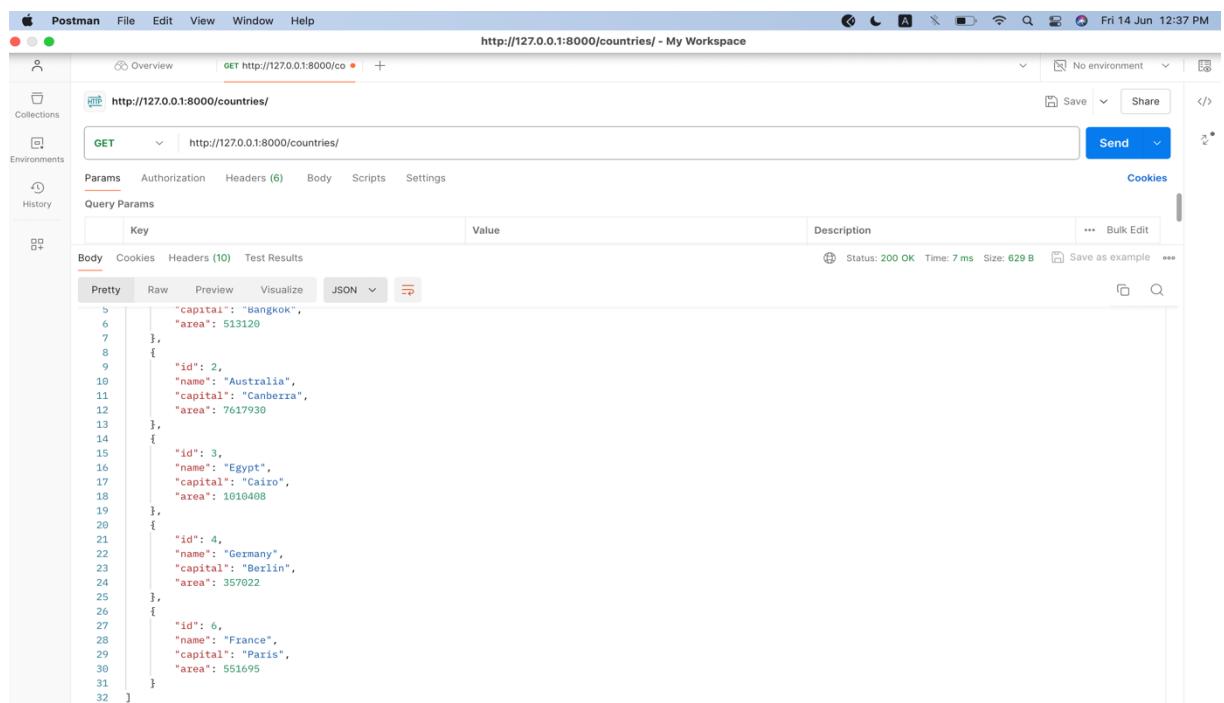
```
{"id":4,"name":"Germany","capital":"Berlin","area":357022}
● → countryapi curl -i -X POST \
-H 'Content-Type: application/json' \
-d '{"name":"France", "capital": "Paris", "area": 551695}' \
http://127.0.0.1:8000/countries/ \
-w '\n'

HTTP/1.1 201 Created
Date: Fri, 14 Jun 2024 19:34:32 GMT
Server: WSGIServer/0.2 CPython/3.12.3
Content-Type: application/json
Vary: Accept, Cookie
Allow: GET, POST, HEAD, OPTIONS
X-Frame-Options: DENY
Content-Length: 56
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

{"id":5,"name":"France","capital":"Paris","area":551695}
○ → countryapi
```

2. Get Request

- When I checked in post man, I have now France been added.



The screenshot shows the Postman application interface. A GET request is made to `http://127.0.0.1:8000/countries/`. The response body is a JSON array of countries, including France, displayed in a pretty-printed format. The array starts with:

```
[{"id": 1, "name": "Thailand", "capital": "Bangkok", "area": 513120}, {"id": 2, "name": "Australia", "capital": "Canberra", "area": 7617930}, {"id": 3, "name": "Egypt", "capital": "Cairo", "area": 1010408}, {"id": 4, "name": "Germany", "capital": "Berlin", "area": 357022}, {"id": 5, "name": "France", "capital": "Paris", "area": 551695}]
```

3. PUT Update a Country

```
t id :5, name : France , capital : Paris , area :550000
● → countryapi curl -i -X PUT \
-H 'Content-Type: application/json' \
-d '{"name":"France", "capital": "Nice", "area": 550000}' \
http://127.0.0.1:8000/countries/5/ \
-w '\n'

HTTP/1.1 200 OK
Date: Fri, 14 Jun 2024 19:35:28 GMT
Server: WSGIServer/0.2 CPython/3.12.3
Content-Type: application/json
Vary: Accept, Cookie
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
X-Frame-Options: DENY
Content-Length: 55
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

{"id":5,"name":"France","capital":"Nice","area":550000}
○ → countryapi █
```

4. PATCH Update Specific Fields of a Country

```
{"id":5,"name":"France","capital":"Nice","area":550000}
● → countryapi curl -i -X PATCH \
-H 'Content-Type: application/json' \
-d '{"capital": "Lyon"}' \
http://127.0.0.1:8000/countries/5/ \
-w '\n'

HTTP/1.1 200 OK
Date: Fri, 14 Jun 2024 19:36:07 GMT
Server: WSGIServer/0.2 CPython/3.12.3
Content-Type: application/json
Vary: Accept, Cookie
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
X-Frame-Options: DENY
Content-Length: 55
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

{"id":5,"name":"France","capital":"Lyon","area":550000}
○ → countryapi █
```

5. DELETE a Country

```
t id :5, name : France , capital : Lyon , area :550000
● → countryapi curl -i -X DELETE \
http://127.0.0.1:8000/countries/5/ \
-w '\n'

HTTP/1.1 204 No Content
Date: Fri, 14 Jun 2024 19:36:37 GMT
Server: WSGIServer/0.2 CPython/3.12.3
Vary: Accept, Cookie
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
X-Frame-Options: DENY
Content-Length: 0
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

○ → countryapi █
```

Explanation of Work Completed

First, I created a new Django project and integrated Django REST framework. Within this project, I created a new Django app, added it to INSTALLED_APPS, and set up the initial model in models.py. After running migration commands to create database tables based on this model, I loaded initial data into the database from a JSON file. I then developed the API by creating serializers, views, and URL configurations to handle requests for the Country model. Following this, I started the Django development server to handle these requests. Finally, I performed client calls (Post, Get, Put, Patch and Delete) to the server I created.

- **“I have neither given nor received unauthorized aid in completing this work, nor have I presented someone else’s work as my own.”**