**Different types of routing in DRF**

## **Introduction**

The Django REST framework provides different ways of URL mapping or routing in an API project. Besides the traditional style of routing, there are other routing techniques that can save you time while developing. In this reading, you are going to learn about both traditional and other techniques.

**Note:** All the routings should be done in the **urls.py** file in your Django app.

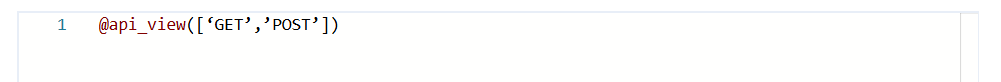
## **Regular routes**

The code below maps a function from a **views.py** file to an API endpoint. Don’t forget that you have to import the path function from the **django.urls** module first.

A close-up of a person

Description automatically generated

This URL pattern maps the books function to the **/api/books** endpoint. You already know how to specify which HTTP methods a function view can serve by supplying them in the **api\_decorator** function. The following code allows any function view to serve both HTTP **GET** and **POST** methods.



## **Routing to a class method**

If you map a specific method from a class, then you need to declare that method as a **@staticmethod** first. After that, you can map it in the **urls.py** file. Here’s an example of a class in the **views.py** file.

A close-up of a computer screen

Description automatically generated

You can also map this **listOrders** method in the **urls.py** file as follows.

A close-up of a computer screen

Description automatically generated

## **Routing class-based views**

You can save a lot of time in DRF by mapping a class that extends the **APIview**. You don’t need to individually map every method of such classes. In an upcoming video, Function and class-based views, you will learn that such classes have HTTP verb-specific methods inside them. When a class extends **APIview** or generic views, you can simply map those classes in the **urls.py** file.

Here’s the code of the class that extends the **APIView**.

A screenshot of a computer program

Description automatically generated

Now you can make HTTP, **GET** and **PUT** calls to the **/api/books/{bookId}** endpoint without issues. If the class has **post()**, **delete()** and **patch()** methods, it will work with HTTP **POST**, **DELETE** and **PATCH** methods too.

## **Routing classes that extend viewsets**

You can have classes that extend the different types of **ViewSets** in your API project. Just like the classes that extend **APIView**, these classes also have specific methods used to respond to different types of HTTP requests. Here’s an example of a typical class that extends the **viewset.Viewset** class.

A screenshot of a computer code

Description automatically generated

 You can map this class in the **urls.py** file in your Django app as follows.

A screenshot of a computer code

Description automatically generated

Notice how the HTTP verbs are mapped with each method in this class. Also, note that both the **list()** and **retrieve()** methods are present. The **list()** method is used to display all books, and the **retrieve()** method is used to display a single book.

After this mapping, you can access the **http://127.0.0.1:8000/api/books** endpoint with **GET** and **POST** methods. While you can access the **http://127.0.0.1:8000/api/books/1** endpoint with **GET**, **PUT**, **PATCH** and **DELETE**.

## **Routing with SimpleRouter class in DRF**

If you have a class that extends **ViewSets** then you can use different types of built-in routers to map those classes in your **urls.py** file. Doing things this way means you don’t have to map the individual methods as you did in the previous section. Initiate a **SimpleRouter** object and map the class in the **urls.py** file in your Django app as follows.

A close-up of a computer screen

Description automatically generated

After mapping, you can access the **api/books** and **api/books/1** endpoints with the same methods as in the previous example.

Did you notice that the argument **trailing\_slash=False** was passed, instantiating the **SimpleRouter** object? Without this argument, your API endpoints will have a trailing slash. And, since you don’t want a trailing slash at the end of your API endpoints, you have to pass this argument.

## Routing with DefaultRouter class in DRF

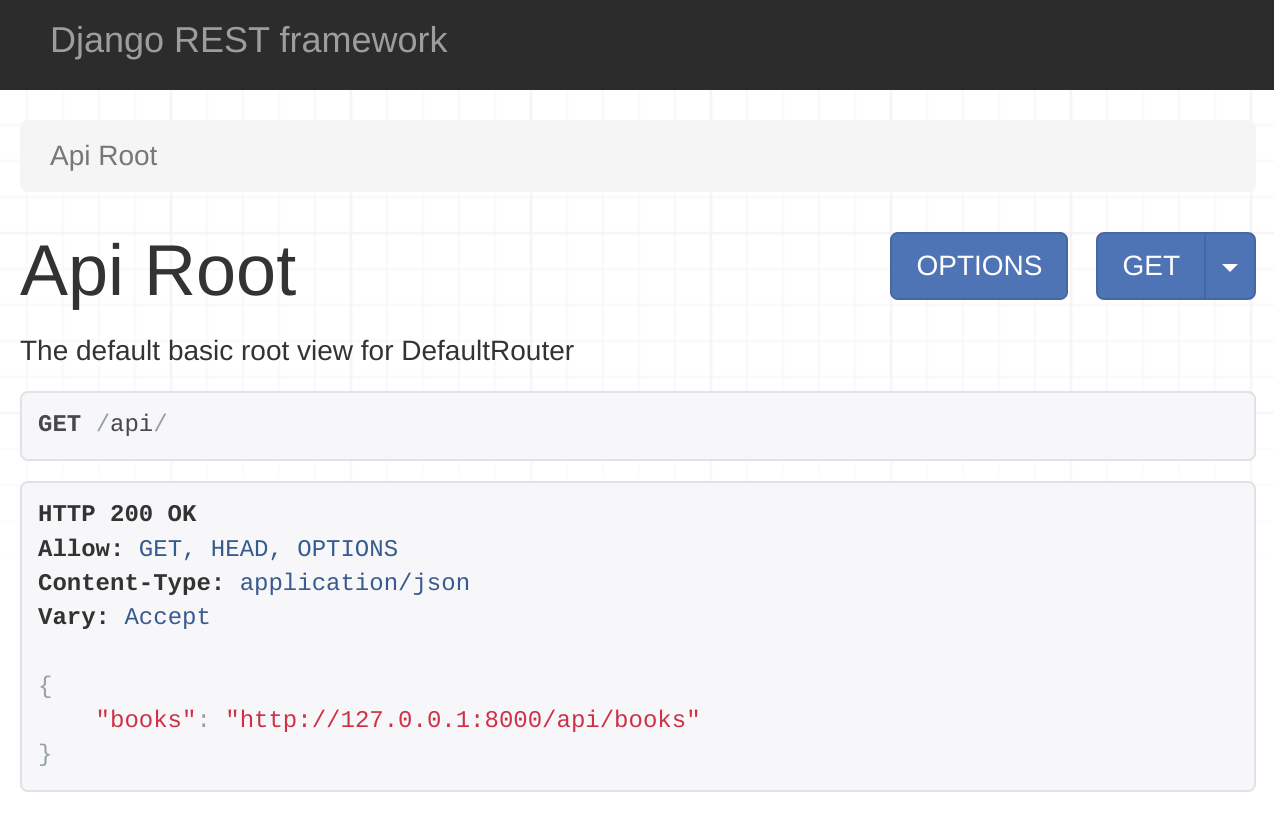
There is another type of router called **DefaultRouter** which provides an extra benefit over the **SimpleRouter**. It creates an API root endpoint with a trailing slash that displays all your API endpoints in one place. You can use it this way in the **urls.py** file.

A screen shot of a computer

Description automatically generated

Again, after mapping, you can access the **api/books** and **api/books/1** endpoints with the same methods as in the previous examples.

Additionally, you can access the API root view when you visit the **http://127.0.0.1:8000/api/** endpoint. This will display all the available endpoints as in the screenshot below.



You can read about all of these routers and more in the DRF documentation shared in the additional resources of this lesson.

## Conclusion

In this reading, you learned about different types of URL routing for your Django REST framework-based API project.