Simplifying CRUD with ModelViewSet and Routers in Django REST Framework

📦 Simplifying CRUD with ModelViewSet and Routers in Django REST Framework

In the previous module, you learned how to create full CRUD functionality using class-based views (`APIView`). While this approach gives you full control, it can also become repetitive, especially when you're building many APIs. Django online courses

In this module, you'll learn about one of the most powerful tools in Django REST Framework (DRF) – the **ModelViewSet** and **Router**. These two work together to reduce boilerplate code and make your CRUD API development much faster and cleaner.

🚀 What is a ViewSet?

A ViewSet is a class that combines logic for multiple HTTP methods (GET, POST, PUT, DELETE) into a single class. It automatically maps your model to views without needing you to write each method manually.

DRF provides different types of viewsets, but the most commonly used is **ModelViewSet**, which supports all CRUD operations out of the box.

```
from rest_framework import viewsets
from .models import Task
from .serializers import TaskSerializer

class TaskViewSet(viewsets.ModelViewSet):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer
```

That's it! With just this class, you get:

GET /tasks/ → List all tasks

POST /tasks/ → Create a new task

GET /tasks/{id}/ → Retrieve one task

PUT /tasks/{id}/ → Update a task

DELETE /tasks/{id}/ → Delete a task

But how does DRF know which URL should go to which action? That's where **Routers** come in.

🔗 What are Routers in DRF?

Routers automatically generate URL patterns for your ViewSets. You don't have to manually define each route like we did earlier.

```
# core/urls.py
from rest_framework.routers import DefaultRouter
from .views import TaskViewSet

router = DefaultRouter()
router.register('tasks', TaskViewSet, basename='task')

urlpatterns = router.urls
```

In your project's main `urls.py`:

```
# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('core.urls')),
]
```

This setup creates RESTful API endpoints instantly without having to manually define each URL.

🧪 Testing Your API

Open Postman or your browser and try the following requests:
GET `http://localhost:8000/api/tasks/` – list all tasks
POST with JSON body to create a task
GET a single task: `/api/tasks/1/`
PUT or PATCH to update
DELETE to remove a task
With ViewSet + Router, all these are instantly available and use the serializer and model you defined.

📌 Why Use ViewSets and Routers?

Less code: You don't need to define each method like in APIView
Faster development: One class can handle all CRUD operations
Consistency: DRF creates a standard pattern for your API URLs
Easier scaling: As your app grows, you simply register more viewsets

🛡️ Customizing ViewSets

You can still customize the behavior. For example, you can override `create()`, `retrieve()`, `update()`, etc.

```
class TaskViewSet(viewsets.ModelViewSet):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer

    def perform_create(self, serializer):
        print("Creating task...")
        serializer.save()
```

You can also apply filters, pagination, permissions, and search easily with ViewSets.

🧠 Extra Tip: Use `basename` Properly

The `basename` argument in `router.register()` is required if your queryset is defined later (like in a mixin). It helps the router name the URLs correctly.

📋 Best Practices

Use ViewSet + Router for fast prototyping and clean APIs
Use APIView if you need full custom behavior
Use Mixins if you need only a few operations (like ReadOnlyModelViewSet)
Keep your serializers simple and validate data properly

✅ Summary

ViewSets and Routers help you write less and build more. With just a few lines, you get a full REST API ready to use. This pattern is widely used in real-world Django apps, especially in startups and large platforms alike.
In the next module, you'll learn how to add **Authentication** (login, logout, and protected routes), and how to secure your APIs using **permissions**.
🚀 Practice Challenge: Create a new model called `Note` and implement full CRUD API using ViewSet and Router.