Welcome to Day12

Python Django

Pre-requisites

- ☐ pip install Django

```
python -m venv myEnv
source venv/bin/activate
venv\Scripts\activate.bat (for windows)
```

Create eComm API

Create project

```
django-admin startproject my_ecom_api
```

```
my_ecom_api/
├── manage.py
└── my_ecom-api/
    ├── __init__.py
    └── asgi.py
    └── settings.py
    └── urls.py
    └── wsgi.py
```

Launch project

```
cd my_ecom_api
python manage.py runserver
```

Create module

```
python manage.py startapp products
```

Create new View

```
python manage.py startapp products
```

my_ecom_api/products/views.py

```python
from django.shortcuts import render
from django.http import HttpResponse

def products(request):
    return HttpResponse("Hello world! Welcome to products")
```

Update URL Navigation

my_ecom_api/products/product_urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('products/', views.products, name='products'),
]
```

Update Overall URL

my_ecom_api/products/product_urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('products/', views.products, name='products'),
]
```

my_ecom_api/my_ecom_api/urls.py

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('', include('products.product_urls')),
    path('admin/', admin.site.urls),
]
```

Hello world view

```
cd ..
my_ecom_api> python manage.py runserver
```

Response as json

json response

my_ecom_api/my_ecom_api/products/views.py

```python
def product(request):
    ...
def product_json(request):
    # In a real application, you would fetch product details from a databas
    product_details = [
        {"id": 1, "name": "Banana Cake", "price": 10.99,
            "description": "A delicious cake made with ripe bananas."},
        {"id": 2, "name": "Coffee Cake", "price": 19.99,
            "description": "A rich coffee cake with a hint of chocolate."},
        {"id": 3, "name": "Vanilla Cake", "price": 5.99,
            "description": "A classic vanilla cake."},
    ]
    return JsonResponse(product_details)
```

Settings

my_ecom_api/my_ecom_api/settings.py

```
INSTALLED_APPS = [
    ...
    'products'
]
```

include URL

my_ecom_api/products/product_urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('products/', views.products, name='products'),
    path('products/json/', views.product_json, name='product_json'),
]
```

Individual product

my_ecom_api/my_ecom_api/products/views.py

```python
def product(request):
    ...
def product_json(request):
    ...
def product_json_detail(request, product_id):
    product_details = [
        {"id": 1, "name": "Banana Cake", "price": 10.99, "description": "with ripe bananas."},
        {"id": 2, "name": "Coffee Cake", "price": 19.99, "description": "Rich coffee cake"},
        {"id": 3, "name": "Vanilla Cake", "price": 5.99, "description": "classic vanilla cake"},
    ]
    product = product_details[product_id]
    return JsonResponse(product,safe=False)
```

include URL

my_ecom_api/products/product_urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('products/', views.products, name='products'),
    path('products/json/', views.product_json, name='product_json'),
    path('products/json/<int:product_id>/', views.product_json_detail,
             name='product_json_detail'),
]
```

Error handling

Handling error

my_ecom_api/my_ecom_api/products/views.py

```python
def product_json_detail(request, product_id):
    product_details = [
        ...
    ]
    if product_id >= len(product_details):
        return JsonResponse({"error": "Product not found"}, status=404)
    product = product_details[product_id]
    return JsonResponse(product)
```

Model in Django

Model

my_ecom_api/products/models.py

```python
from django.db import models


class Product(models.Model):
  Name = models.CharField(max_length=55)
  Price = models.FloatField()
  Desc = models.CharField(max_length=100)
```

commands

- step 1

```
python manage.py makemigrations products
```

- step 2

```
python manage.py migrate
```

- (Optional) step 3

```
(optional) python manage.py sqlmigrate products 0001
```

Add Records

```
python manage.py shell
```

Insert the data

```
>>> from products.models import Product
>>> Product.objects.all()
>>> vanilla =  Product(Name="Vanilla Cake", Price = 3, Desc="Fresh Cake")
>>> vanilla.save()
>>> Product.objects.all()
```

Display model in page

fetch Models

- views.py

```python
...
from .models import Product
def product_json(request):
    allProducts = Product.objects.all().values()
    return JsonResponse(list(allProducts), safe=False)
def product_json_detail(request, product_id):
    allProducts = Product.objects.all().values()
    if product_id >= len(allProducts):
        return JsonResponse({"error": "Product not found"}, status=404)
    allProductsList = list(allProducts)
    product = allProductsList[product_id]
    return JsonResponse(product)
```

Fetch/filter data

```python
def product_json_detail(request, product_id):
    thisProduct = Product.objects.filter(id=product_id).values()
    if not thisProduct:
        return JsonResponse({"error": "Product not found"}, status=404)
    return JsonResponse(thisProduct[0], safe=False)
```

Filter by Name

filter name

my_ecom_api/products/product_urls.py

```python
urlpatterns = [
    path('products/', views.products, name='products'),
    path('products/json/', views.product_json, name='product_json'),
    path('products/json/<int:product_id>/', views.product_json_detail, name='product_json_detail'),
    path('products/json/<str:name>/', views.prod_name, name='prod_name'),
]
```

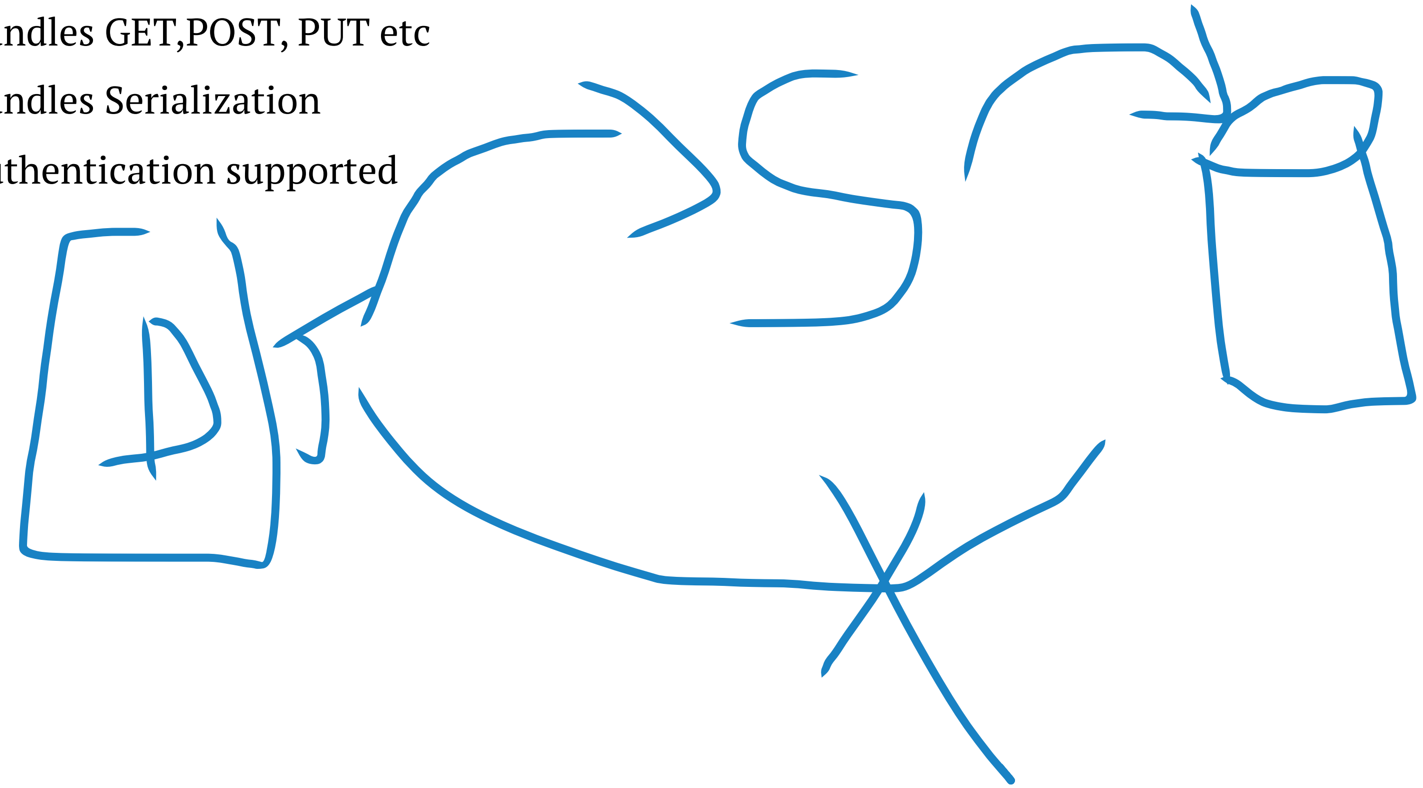
my_ecom_api/products/views.py

```python
def prod_name(request, name):
    thisProduct = Product.objects.filter(Name=name).values()
    if not thisProduct:
        return JsonResponse({"error": "Product not found"}, status=404)
    return JsonResponse(thisProduct[0], safe=False)
```

Rest Framework

Rest Framework?

- It has all the REST features packaged together
- Handles GET,POST, PUT etc
- Handles Serialization
- Authentication supported



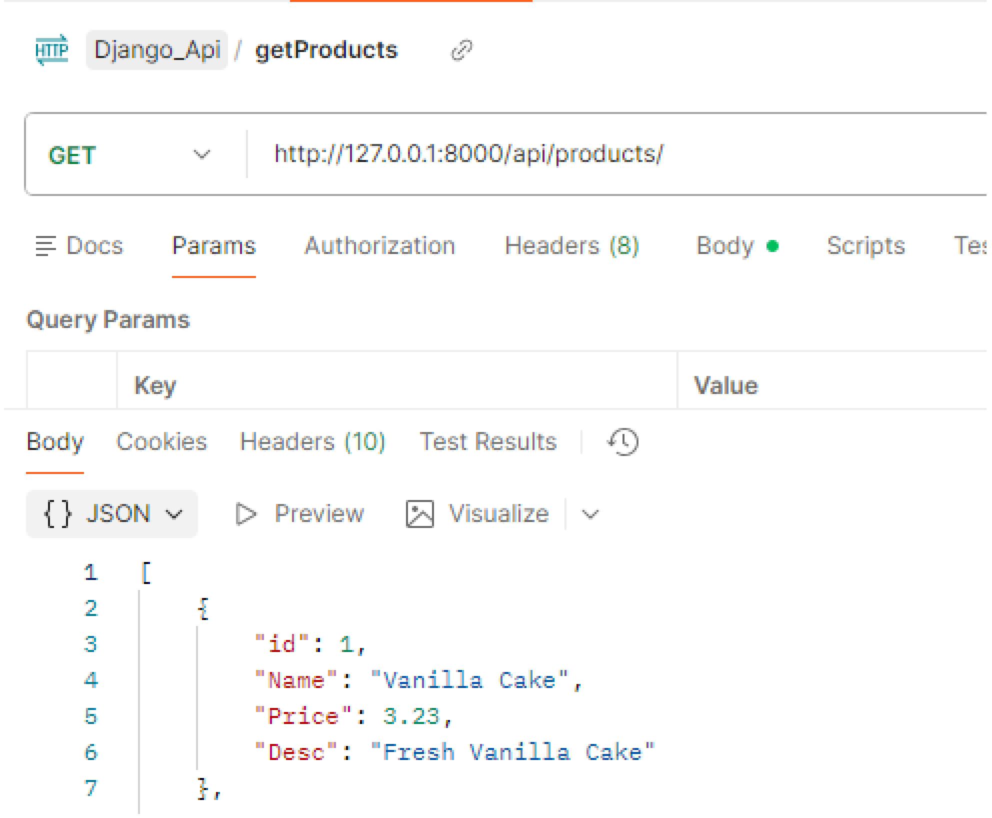
Rest Framework?

```
pip install djangorestframework
```

my_ecom_api/my_ecom_api/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    ...,
    'products',
    'rest_framework',
]
```

fetch request



GET http://127.0.0.1:8000/api/products/

Query Params

Key	Value

```
[
    {
        "id": 1,
        "Name": "Vanilla Cake",
        "Price": 3.23,
        "Desc": "Fresh Vanilla Cake"
    },
```

Serializer

my_ecom_api/products/serializers.py

```python
from rest_framework import serializers
from .models import Product

class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = '__all__'
```

Fetch product

my_ecom_api/products/views.py

```python
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .serializers import ProductSerializer

@api_view(['GET'])
def get_products(request):
    allProducts = Product.objects.all().values()
    serializer = ProductSerializer(Product.objects.all(), many=True)
    return Response(serializer.data)
```

my_ecom_api/products/product_urls.py

```python
urlpatterns = [
    ...
    path('api/products/', views.get_products, name='get_products'),
]
```

Admin panel

Create user

```
python manage.py createsuperuser
```

- models.py

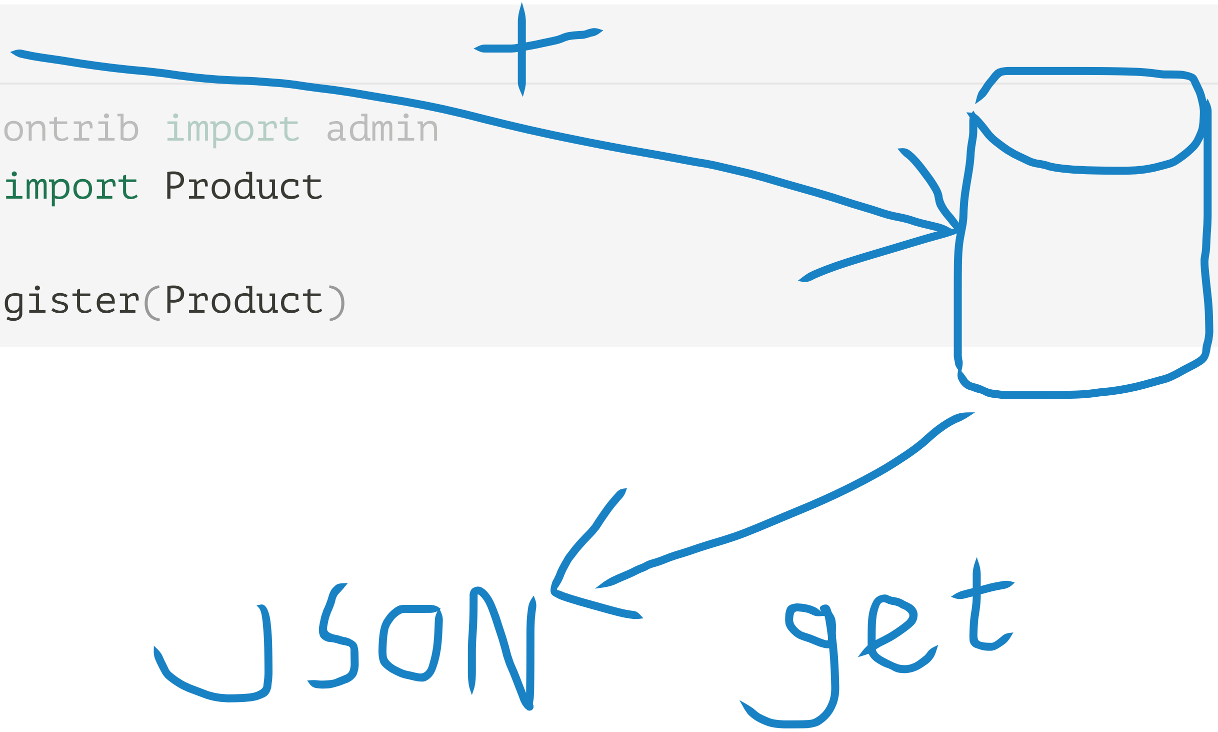
admin.py

```python
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```

models

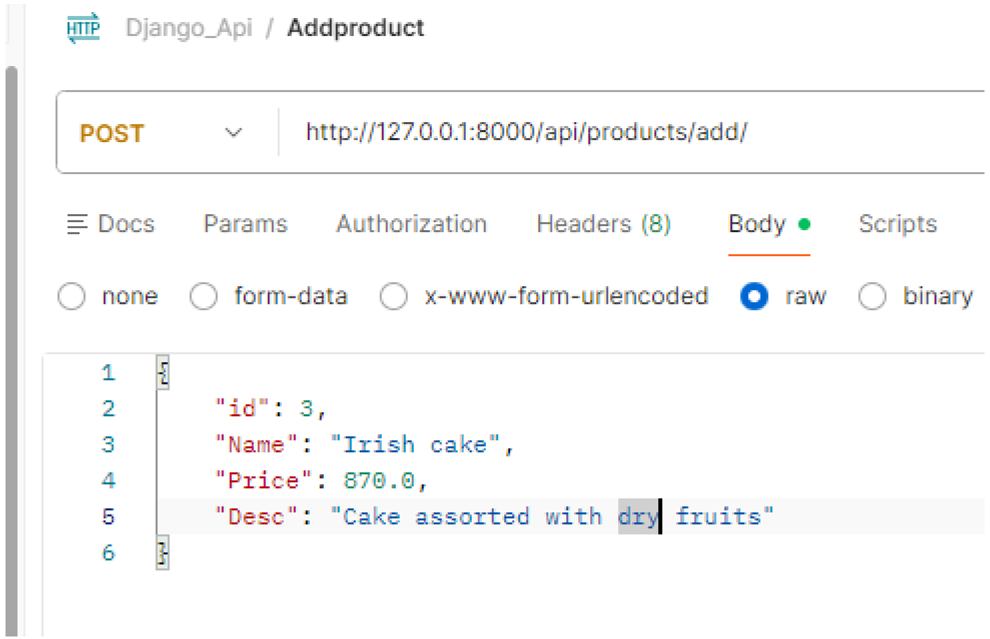
admin.py

```python
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```



Add new Product via request

post request



POST http://127.0.0.1:8000/api/products/add/

```
{
    "id": 3,
    "Name": "Irish cake",
    "Price": 870.0,
    "Desc": "Cake assorted with dry fruits"
}
```

Add new Product via request

views.py

```
from rest_framework import status

@api_view(['GET'])
def get_products(request):
    ...
@api_view(['POST'])
def add_product(request):
    serializer = ProductSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    else:
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

def products(request):
```

Serialzers.py

```
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = ['id', 'Name', 'Price', 'Desc']
```

product_urls.py

```


urlpatterns = [
    ...
    path('api/products/add/', views.add_product, name='add_product'),
]
```

Delete item

product_urls.py

```
urlpatterns = [
    ...
    path('api/products/remove/<int:product_id>', views.remove_product, name='remove_produc
]
```

views.py

```
@api_view(['DELETE'])
def remove_product(request, product_id):
    try:
        product = Product.objects.get(id=product_id)
        product.delete()
        return JsonResponse({"message": "Product removed successfully"})
    except Product.DoesNotExist:
        return JsonResponse({"error": "Product not found"}, status=404)
```