Django

TomerBu

נושאים להיום:

חזרה על נושאים משיעורים קודמים -
ModelSerializer
APIViews

Class Based Views:
Generic Api Views: mixins

Built In Classes:
ListCreateApiView
RetrieveUpdateDestroyAPIView

ViewSets

ModelViewSets

שיעורי בית:

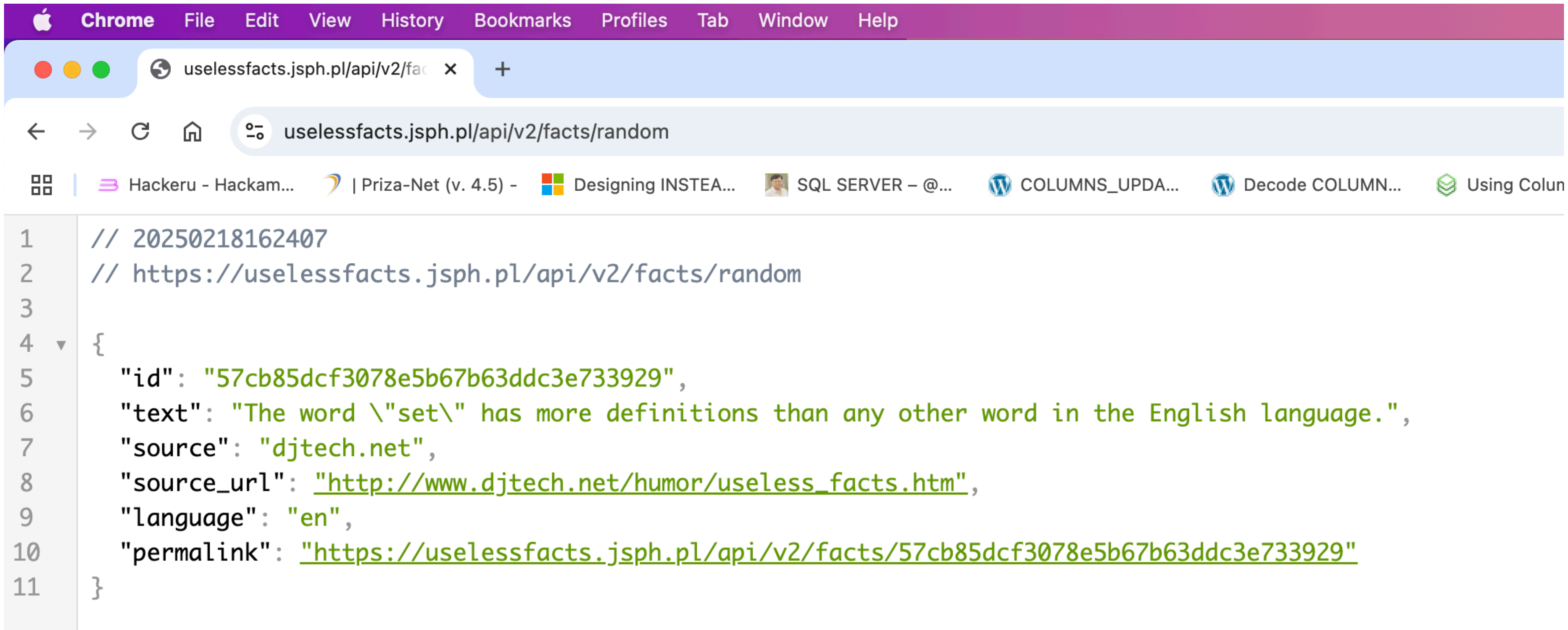
צרו API שמחזיר מידע על Fun Facts:

מודל:
FunFact:
תכונות:
text, source, source_url, language, permalink

יש לממש את כל פעולות הCRUD
כפי שמימשנו בכיתה

דוגמא לAPI כזה:

https://uselessfacts.jsph.pl/api/v2/facts/random


Chrome File Edit View History Bookmarks Profiles Tab Window Help
uselessfacts.jsph.pl/api/v2/fa
uselessfacts.jsph.pl/api/v2/facts/random
Hackeru - Hackam... | Priza-Net (v. 4.5) - Designing INSTEA... SQL SERVER – @... COLUMNS_UPDA... Decode COLUMN... Using Colun

```
// 20250218162407
// https://uselessfacts.jsph.pl/api/v2/facts/random

{
  "id": "57cb85dcf3078e5b67b63ddc3e733929",
  "text": "The word \"set\" has more definitions than any other word in the English language.",
  "source": "djtech.net",
  "source_url": "http://www.djtech.net/humor/useless_facts.htm",
  "language": "en",
  "permalink": "https://uselessfacts.jsph.pl/api/v2/facts/57cb85dcf3078e5b67b63ddc3e733929"
}
```

שיעורי בית:

```
django-admin startproject lec4
```

יצירת venv:

```
python -m venv drf-venv
```

```
./drf-venv/scripts/activate
```

התקנת ספריות:

```
pip install django
pip install djangorestframework
```

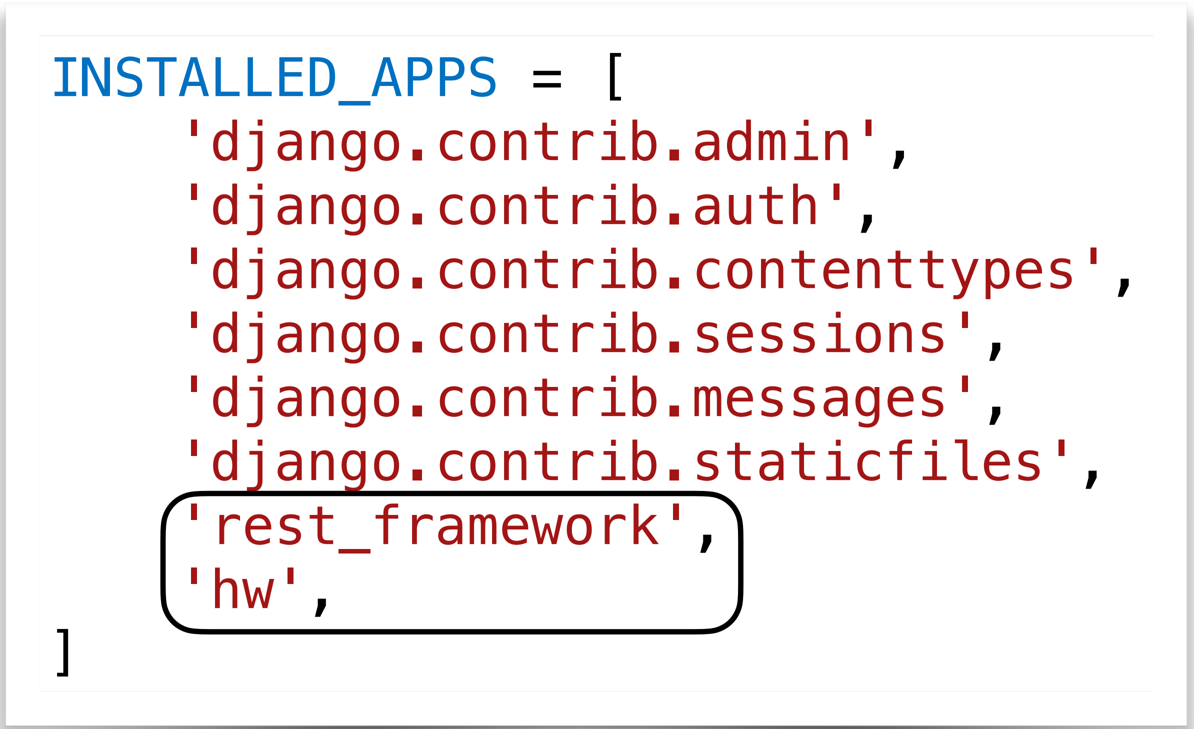
יצירת app:

```
python manage.py startapp hw
```

הרצת השרת:

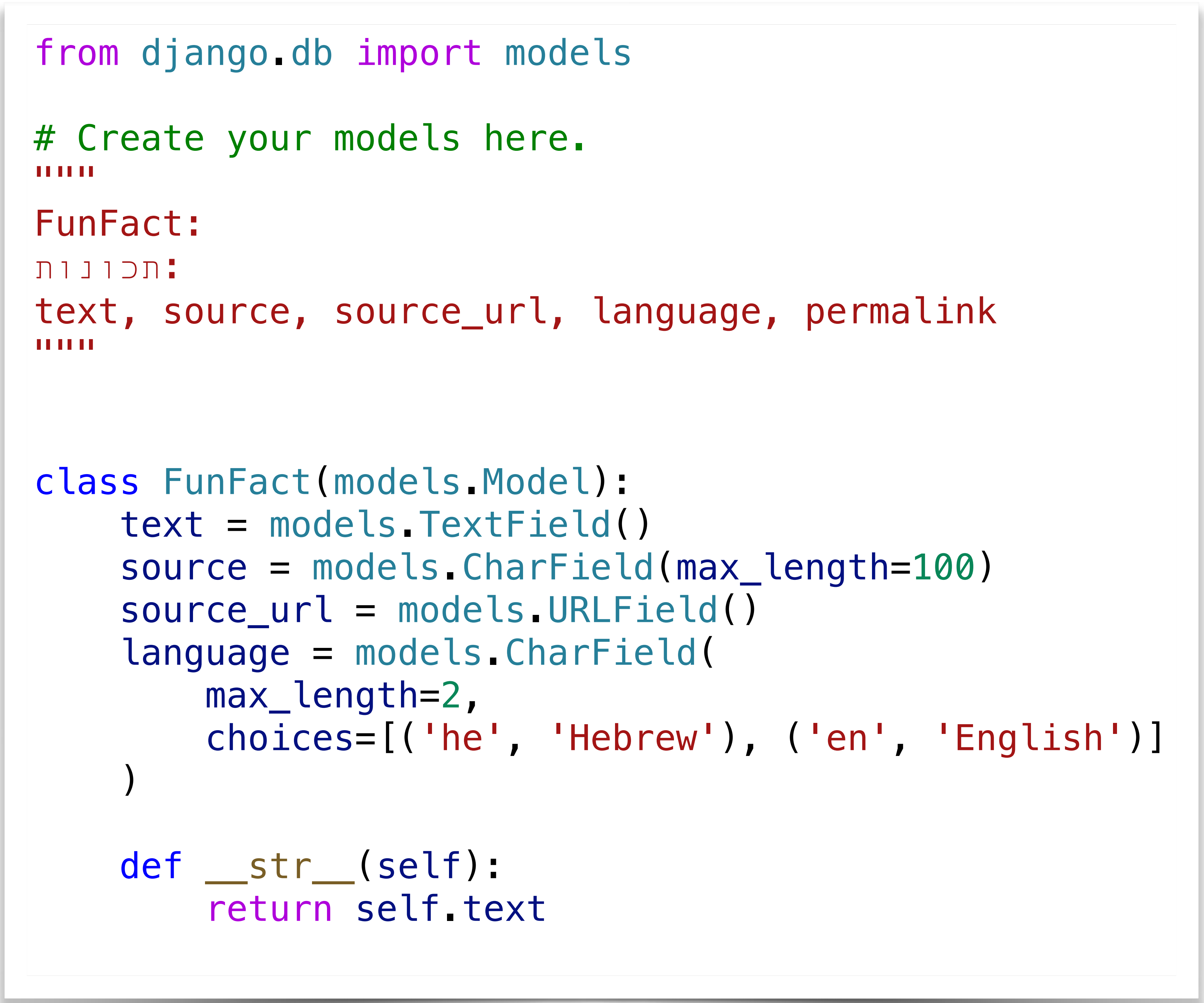
```
python manage.py runserver
```

שיעורי בית:



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'hw',
]
```

שיעורי בית:



```python
from django.db import models

# Create your models here.
"""
FunFact:
תכונות:
text, source, source_url, language, permalink
"""



class FunFact(models.Model):
    text = models.TextField()
    source = models.CharField(max_length=100)
    source_url = models.URLField()
    language = models.CharField(
        max_length=2,
        choices=[('he', 'Hebrew'), ('en', 'English')]
    )

    def __str__(self):
        return self.text
```

```
python manage.py makemigrations hw
```

```
python manage.py migrate
```

שיעורי בית:

```python
from django.contrib import admin

from hw.models import FunFact

# Register your models here.

admin.site.register([FunFact])


# python manage.py createsuperuser
```

```
python manage.py createsuperuser
```

שיעורי בית:

hw/urls.py

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

urlpatterns = [
    path('facts/', FunFactsView.as_view(), name='fun-facts'),
]
```

lec4/urls.py

```python
from django.contrib import admin
from django.urls import path, include

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

# api/v1/facts
```

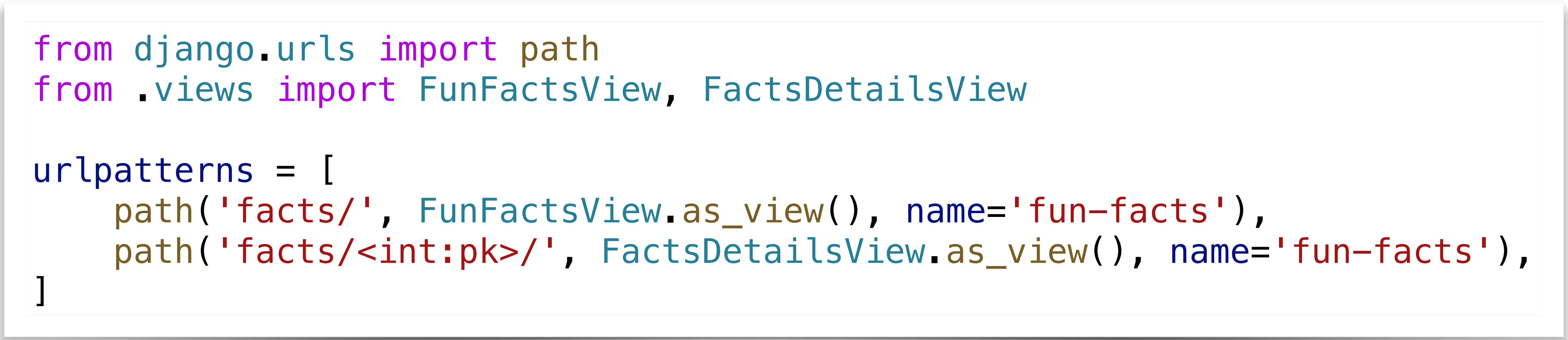
http://localhost:8000/api/facts/

שיעורי בית:

```python
from rest_framework.serializers import ModelSerializer
from .models import FunFact

class FunFactsSerializer(ModelSerializer):
    class Meta:
        model = FunFact
        # fields = ['id', 'fact', 'source']
        fields = '__all__'
        # exclude = ['password']
```

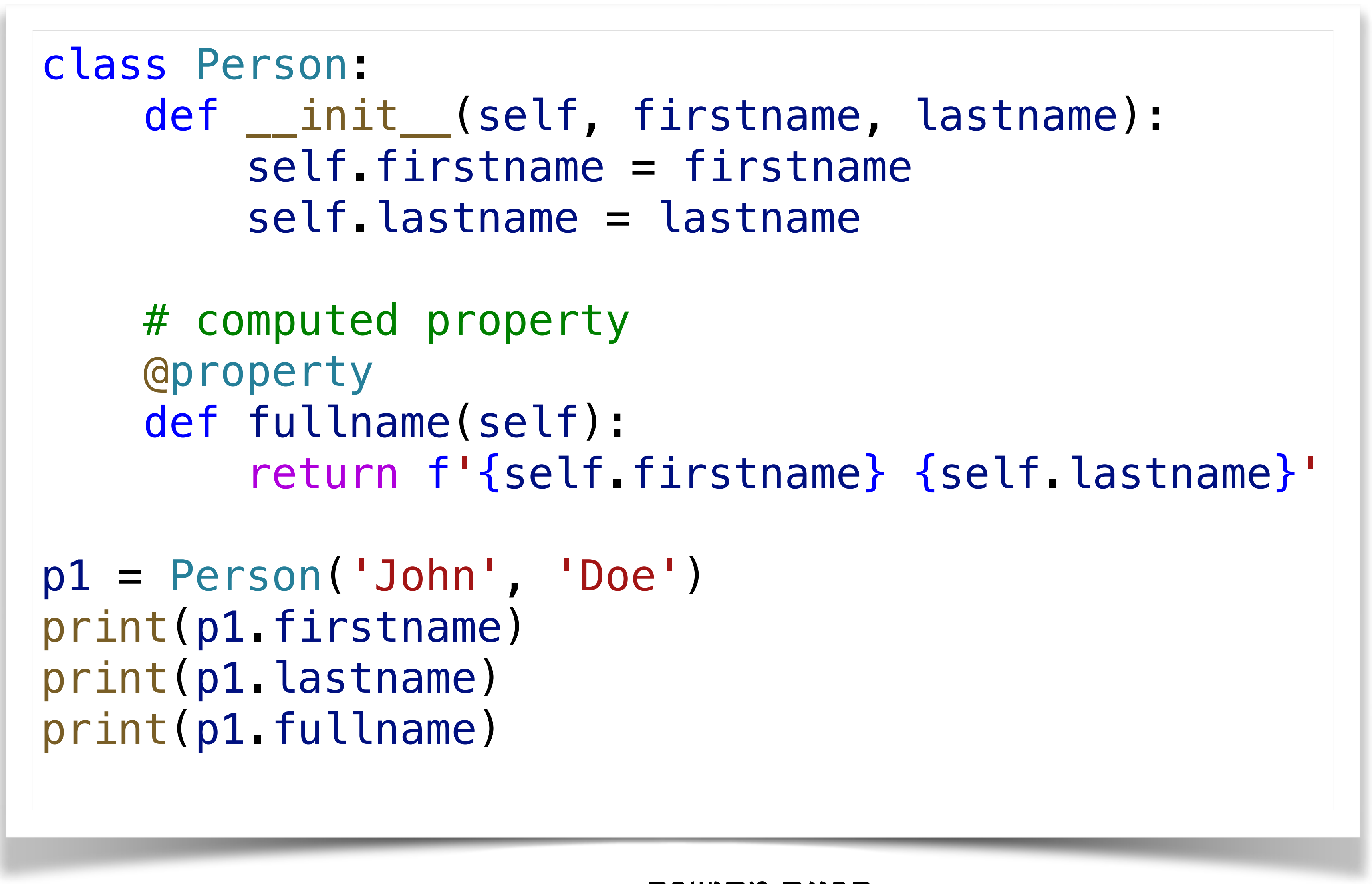
שיעורי בית:



```python
from django.urls import path
from .views import FunFactsView, FactsDetailsView

urlpatterns = [
    path('facts/', FunFactsView.as_view(), name='fun-facts'),
    path('facts/<int:pk>/', FactsDetailsView.as_view(), name='fun-facts'),
]
```

תכונות Properties:



```python
class Person:
    def __init__(self, firstname, lastname):
        self.firstname = firstname
        self.lastname = lastname

    # computed property
    @property
    def fullname(self):
        return f'{self.firstname} {self.lastname}'

p1 = Person('John', 'Doe')
print(p1.firstname)
print(p1.lastname)
print(p1.fullname)
```

תכונה מחושבת:

שמור שם פרטי
שמור שם משפחה

ניצור תכונה ״נגזרת״
תכונה שמחושבת מערכי תכונות אחרות

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get

ניהול תגיות

many to many

תגית

כתבה

tags must be unique

יעילות חיפוש לפי תגיות

כלול בממשק הניהול

פילטרים בממשק ניהול
לחיפוש לפי תגיות

אפליקציה חדשה לשיעור:

```
python manage.py startapp blog
```

```
pip install django-taggit
```

ספריה לניהול תגיות

```
pip install Pillow
```

ספריה לעבודה עם קבצי תמונה

settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'taggit',
    'hw',
    'blog',
]
```

מודלים לבלוג:

lec4/blog/models.py

```python
from django.db import models
from django.contrib.auth.models import User

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, unique=True)
    bio = models.TextField(blank=True, max_length=1000)
    profile_pic = models.ImageField(upload_to='profile_pics', blank=True)
    birth_date = models.DateField(null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    @property
    def username(self):
        return self.user.username

    def __str__(self):
        return f'{self.user.username}'
```

מודלים לבלוג:

lec4/blog/models.py

```python
class Post(models.Model):
    author = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    title = models.CharField(max_length=100, unique=True, validators=[
        MinLengthValidator(5),
        MaxLengthValidator(100),
        RegexValidator(
            regex = '^[a-zA-Z].*$',
            message = "Title must start with a letter")
    ])
    text = models.TextField(
        validators=[
            MinLengthValidator(10)
        ]
    )
    tags = TaggableManager()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f'{self.title} by {self.author.username}'
```

הספריה taggit
בונה טבלת tags
מיישמת עבורנו רבים לרבים

תוספת סטטוס:
כתבה יכולה להיות בסטטוס:
draft, published, archived

מודלים לבלוג:

lec4/blog/models.py

```python
STATUS_CHOICES = [
    ('draft', 'Draft'),
    ('published', 'Published'),
    ('archived', 'Archived')
]

class Post(models.Model):
    author = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    title = models.CharField(max_length=100, unique=True, validators=[
        MinLengthValidator(5),
        MaxLengthValidator(100),
        RegexValidator(
            regex='^[a-zA-Z].*$',
            message="Title must start with a letter")
    ])
    text = models.TextField(
        validators=[
            MinLengthValidator(10)
        ]
    )
    tags = TaggableManager()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    status = models.CharField(
        max_length=10,
        choices=STATUS_CHOICES,
        default="draft"
    )

    def __str__(self):
        return f'{self.title} by {self.author.username}'
```

מודלים לבלוג:

lec4/blog/models.py

```python
class Comment(models.Model):
    author = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    text = models.TextField(
        validators = [MinLengthValidator(2)]
    )
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    reply_to = models.ForeignKey(
        'self',
        on_delete=models.CASCADE,
        default=None,
        null=True,
        blank=True
    )

    def __str__(self):
        return f'{self.text} by {self.author.username}'
```

מודלים לבלוג:

lec4/blog/models.py

```
class PostUserLikes(models.Model):
    user = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    like_type = models.CharField(
        choices=LIKE_CHOICES,
        max_length=10,
        default='like'
    )
    created_at = models.DateTimeField(auto_now_add=True)

    # prevent multiple likes/dislikes per post
    class Meta:
        unique_together = ['user', 'post']

    def __str__(self):
        return f'{self.user.username} {self.like_type}d {self.post.title}'
```

שלבי עבודה:

1) models + migrations + admin

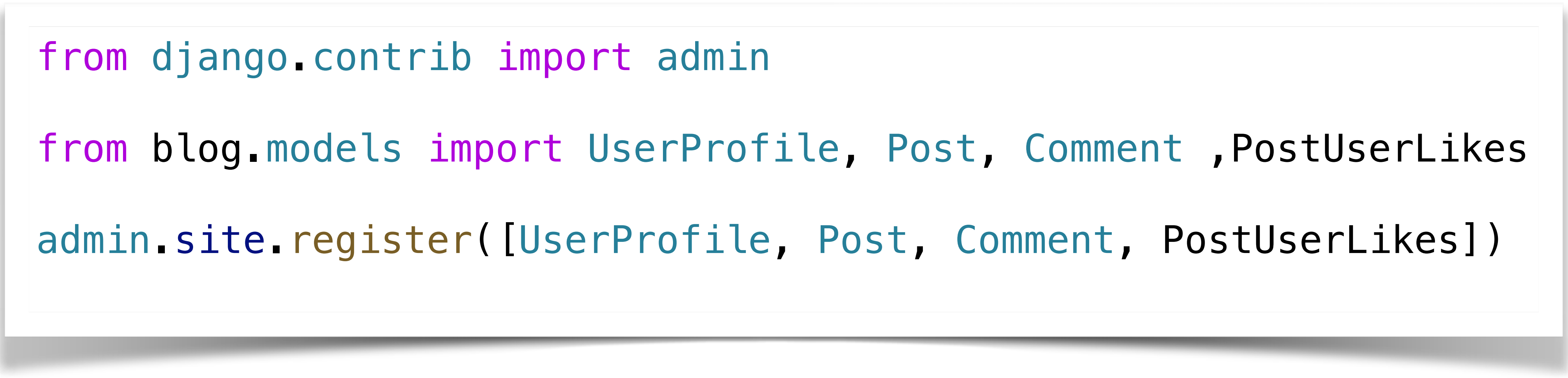
2) serializers - for json

3) views

4) urls

1

```
python manage.py makemigrations blog
python manage.py migrate
```



```
from django.contrib import admin

from blog.models import UserProfile, Post, Comment ,PostUserLikes

admin.site.register([UserProfile, Post, Comment, PostUserLikes])
```

Json Serializers

blog/serializers.py

```
from .models import Post, Comment, UserProfile, PostUserLikes
from rest_framework.serializers import ModelSerializer

class PostSerializer(ModelSerializer):
    class Meta:
        model = Post
        fields = '__all__'

class CommentSerializer(ModelSerializer):
    class Meta:
        model = Comment
        fields = '__all__'

class UserProfileSerializer(ModelSerializer):
    class Meta:
        model = UserProfile
        fields = '__all__'

class PostUserLikesSerializer(ModelSerializer):
    class Meta:
        model = PostUserLikes
        fields = '__all__'
```

Generic Views:

blog/views.py

```python
from rest_framework.generics import GenericAPIView
from rest_framework.mixins import *
from .serializers import *
from .models import *

class PostsView(GenericAPIView, ListModelMixin ,CreateModelMixin):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def get(self, request):
        # use ListModelMixin's list method
        return self.list(request)

    def post(self, request):
        # use CreateModelMixin's create method
        return self.create(request)
```

Generic Views:

blog/urls.py

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

urlpatterns = [
    path('posts/', PostsView.as_view(), name='posts')
]
```

lec4/urls.py

```
from django.contrib import admin
from django.urls import path, include

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

http://localhost:8000/api/blog/posts/

Generic Views:

View לפעולות עריכה, מחיקה ועדכון

lec4/blog/views.py

```
class PostActions(GenericAPIView, UpdateModelMixin, DestroyModelMixin, RetrieveModelMixin):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def get(self, request, pk):
        return self.retrieve(request, pk)

    def put(self, request, pk):
        return self.update(request, pk)

    def delete(self, request, pk):
        return self.destroy(request, pk)
```

lec4/blog/urls.py

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

urlpatterns = [
    path('posts/', PostsView.as_view(), name='posts'),
    path('posts/<int:pk>', PostActions.as_view(), name='post-actions'),
]
```

עבודה עם ListCreateApiView

ListCreateAPIView

מחלקה שמבוססת על GenericAPIView
מגיעה מוכנה עם כל הפעולות להוספת מידע ולהצגת רשימה:

```python
from rest_framework.generics import ListCreateAPIView, RetrieveUpdateDestroyAPIView

from .serializers import PostSerializer
from .models import *


class PostsView(ListCreateAPIView):
    serializer_class = PostSerializer
    queryset = Post.objects.all()

class PostActions(RetrieveUpdateDestroyAPIView):
    serializer_class = PostSerializer
    queryset = Post.objects.all()
```

RetrieveUpdateDestroyAPIView

מחלקה שמבוססת על GenericAPIView
מגיעה מוכנה עם כל הפעולות לעדכון, מחיקה והצגת פרטים

הצגת עמוד ראשי עם קישורים:

views.py

```
from rest_framework.reverse import reverse
from rest_framework.views import APIView
from rest_framework.response import Response
class APIMap(APIView):
   """My Blog Map"""
   def get(self, request):
    return Response({
        "posts": reverse('posts', request=request),
        "post-details":  reverse('post-actions',kwargs = {"pk":1}, request=request),
    })
```

urls.py

```
from django.urls import path
from .views import PostsView2, PostActions2, APIMap

urlpatterns = [
    path('', APIMap.as_view(), name='map'),
    path('posts/', PostsView2.as_view(), name='posts'),
    path('posts/<int:pk>', PostActions2.as_view(), name='post-actions'),
]
```

View Sets:

עוד אלמנט מאוד נוח מבית DjangoRestFramework:

`ViewSets`

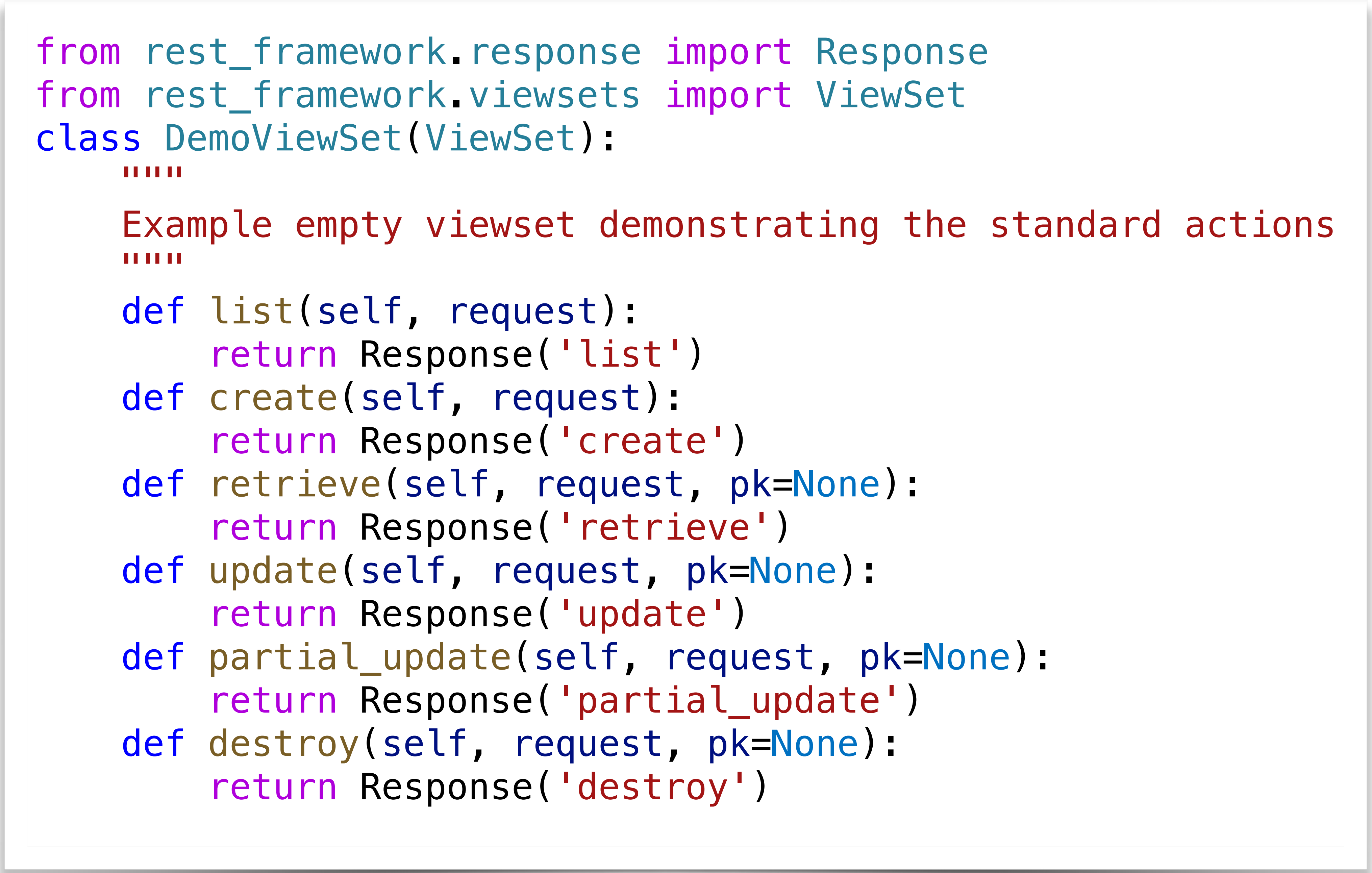
איחוד בין הפעולות של List
לבין הפעולות של Actions

נכתוב את כל הפעולות במחלקה אחת
ונשתמש בRouter כדי למפות את הViews לURLS

View Sets:

כל הפעולות במקום אחד:

views.py



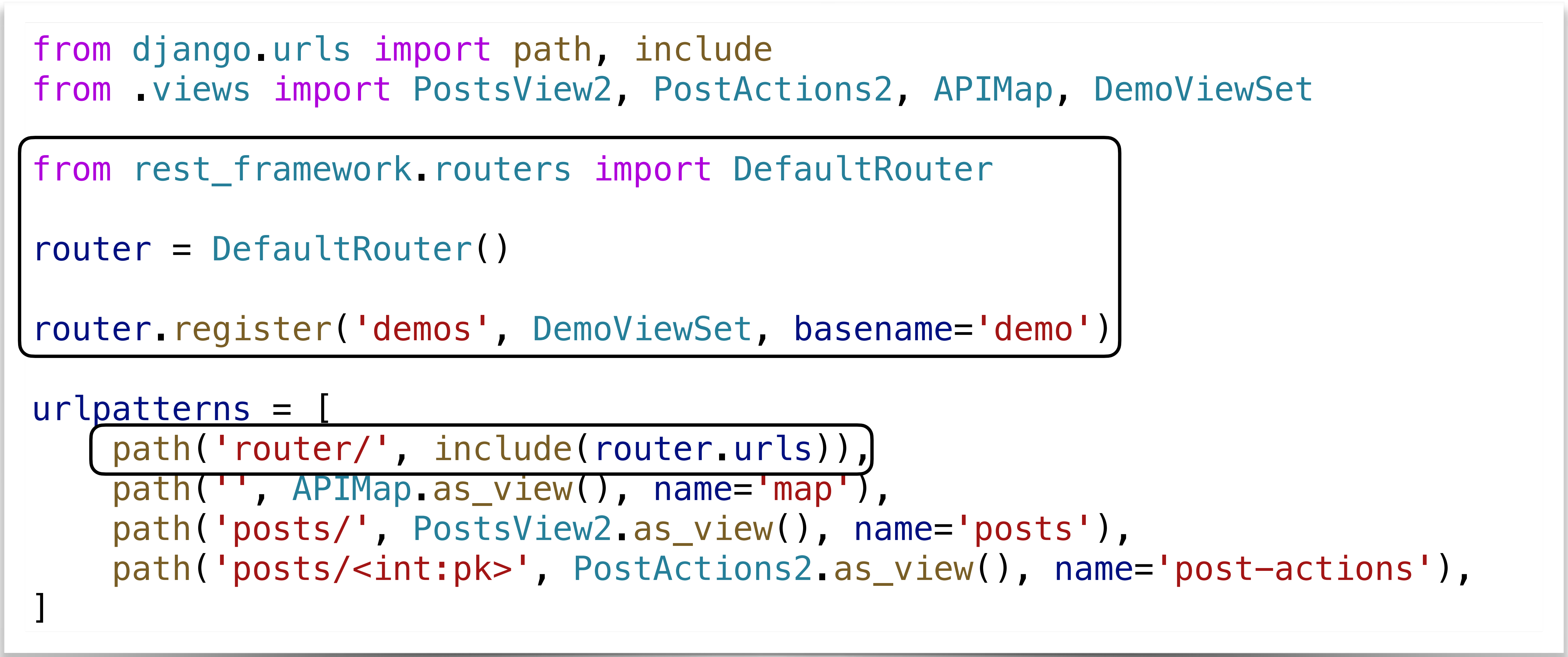
```python
from rest_framework.response import Response
from rest_framework.viewsets import ViewSet
class DemoViewSet(ViewSet):
    """
    Example empty viewset demonstrating the standard actions
    """
    def list(self, request):
        return Response('list')
    def create(self, request):
        return Response('create')
    def retrieve(self, request, pk=None):
        return Response('retrieve')
    def update(self, request, pk=None):
        return Response('update')
    def partial_update(self, request, pk=None):
        return Response('partial_update')
    def destroy(self, request, pk=None):
        return Response('destroy')
```

https://www.django-rest-framework.org/api-guide/viewsets/#viewset-actions

View Sets:

ראוטרים מאחדים את פעולות הdetails
עם פעולות הlist

blog/urls.py



```python
from django.urls import path, include
from .views import PostsView2, PostActions2, APIMap, DemoViewSet

from rest_framework.routers import DefaultRouter

router = DefaultRouter()

router.register('demos', DemoViewSet, basename='demo')

urlpatterns = [
    path('router/', include(router.urls)),
    path('', APIMap.as_view(), name='map'),
    path('posts/', PostsView2.as_view(), name='posts'),
    path('posts/<int:pk>', PostActions2.as_view(), name='post-actions'),
]
```

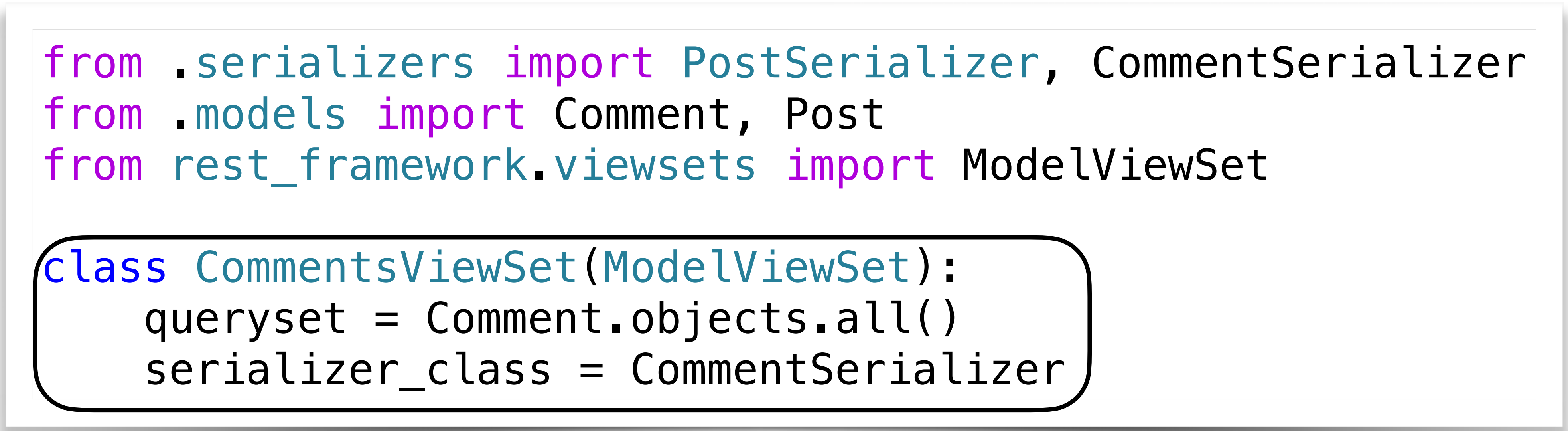
http://localhost:8000/api/blog/router/demos/

ModelViewSets

blog/urls.py

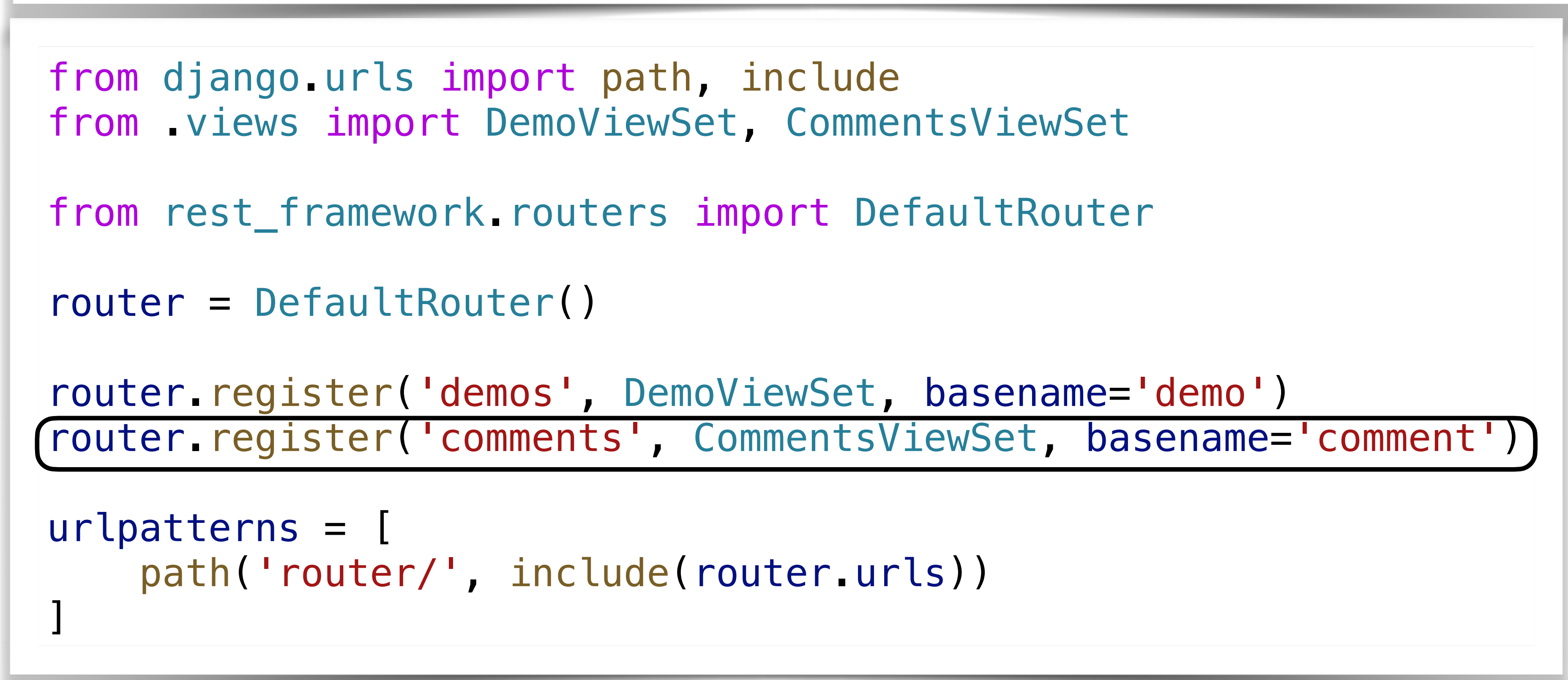
ModelViewSet
מימוש כל פעולות הCRUD

במקום אחד



```python
from .serializers import PostSerializer, CommentSerializer
from .models import Comment, Post
from rest_framework.viewsets import ModelViewSet

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
```



```python
from django.urls import path, include
from .views import DemoViewSet, CommentsViewSet

from rest_framework.routers import DefaultRouter

router = DefaultRouter()

router.register('demos', DemoViewSet, basename='demo')
router.register('comments', CommentsViewSet, basename='comment')

urlpatterns = [
    path('router/', include(router.urls))
]
```

שיעורי בית:

ממשו את כל פעולות הCRUD
למודלים בBlog App

באמצעות ModelViewSet

והנגישו את הURLS באמצעות הRouter