

Django

TomerBu

נושאים להיום:

utils לפרויקטים

העלאת תמונה פרופיל

Users Permissions Groups

Authorization

Authentication

Token Authentication

שימוש במודול לפעולות עזר:

views.py

```
from core.utils import try_parse_int
from .serializers import *
from .models import Comment, Post, UserProfile, PostUserLikes
from rest_framework.viewsets import ModelViewSet
from rest_framework.response import Response

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

    def create(self, request, *args, **kwargs):
        data = request.data
        reply_to = data.get('reply_to')
        post_id = try_parse_int(data.get('post'))

        if reply_to:
            replied = Comment.objects.get(id=reply_to)
            print(replied)
            if (
                replied and replied.post.id != post_id
            ):
                return Response(
                    {"error": "Reply must be on the same post"},
                    status=400
                )

        return super().create(request, *args, **kwargs)
```

בשיעור הקודם - בדקנו
בדיקה שלילית - ולא עליינו
על הצורך בparse

זה ממחיש את הצורך
לבצע בדיקה שלילית
ובדיקה חיובית

מודול לפעולות עזר:

```
✓ LEC4
  > .vscode
  > blog
  ✓ core
    utils.py
  > drf-venv
```

```
def try_parse_int(value: str, default=None):
    try:
        return int(value)
    except (ValueError, TypeError):
        return default
```

UX Anti Pattern

Please enter all information as you would like it to appear in race materials

First Name: *	Emilio
Middle Name:	
Last Name: *	Lizardo
Gender: *	<input checked="" type="radio"/> Male <input type="radio"/> Female
Birthdate: *	01 / 01 / 1981
Email: * ?	ez@example.com
Enter Email Again: *	ez@example.com
Day Phone: *	312-555-5555 ext.
Evening Phone:	
Address Line 1: *	123 Yoyodyne Street
Address Line 2:	
City: *	Whippany
Country: *	United States
State: *	NJ
Zip/Postal Code: *	07981
Your age on race day, April 7, 2013. *	31
Running Shirt Size	

user

userprofile

תמונה פרופיל:

Django REST framework!!!

User Profile List

```
GET /api/v1/user-profile/  
  
HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept  
  
[  
    {  
        "id": 1,  
        "bio": "blogger blogs",  
        "profile_pic": null,  
        "birth_date": "1990-10-10",  
        "created_at": "2025-02-23T08:26:02.924244Z",  
        "updated_at": "2025-02-23T08:26:02.924263Z",  
        "user": 1  
    },  
    {  
        "id": 2,  
        "bio": "site mod",  
        "profile_pic": null,  
        "birth_date": "1990-10-10",  
        "created_at": "2025-02-23T08:28:40.183062Z",  
        "updated_at": "2025-02-23T08:28:40.183073Z",  
        "user": 2  
    },  
    {  
        "id": 3,  
        "bio": "likes to read",  
        "profile_pic": null,  
        "birth_date": "1989-02-02",  
        "created_at": "2025-02-23T08:28:59.003396Z",  
        "updated_at": "2025-02-23T08:28:59.003407Z",  
        "user": 3  
    }  
]
```

Bio

Profile pic Choose File No file chosen

Birth date dd/mm/yyyy

User moderator

תמונה פרופיל:

settings.py

```
import os

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

lec4/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings
from lec4.settings import MEDIA_ROOT, MEDIA_URL

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/', include('blog.urls')),
]

# https://docs.djangoproject.com/en/5.1/howto/static-files/#serving-static-files-during-development
urlpatterns += static(
    settings.MEDIA_URL,
    document_root=settings.MEDIA_ROOT
)
```

תמונה פרופיל:

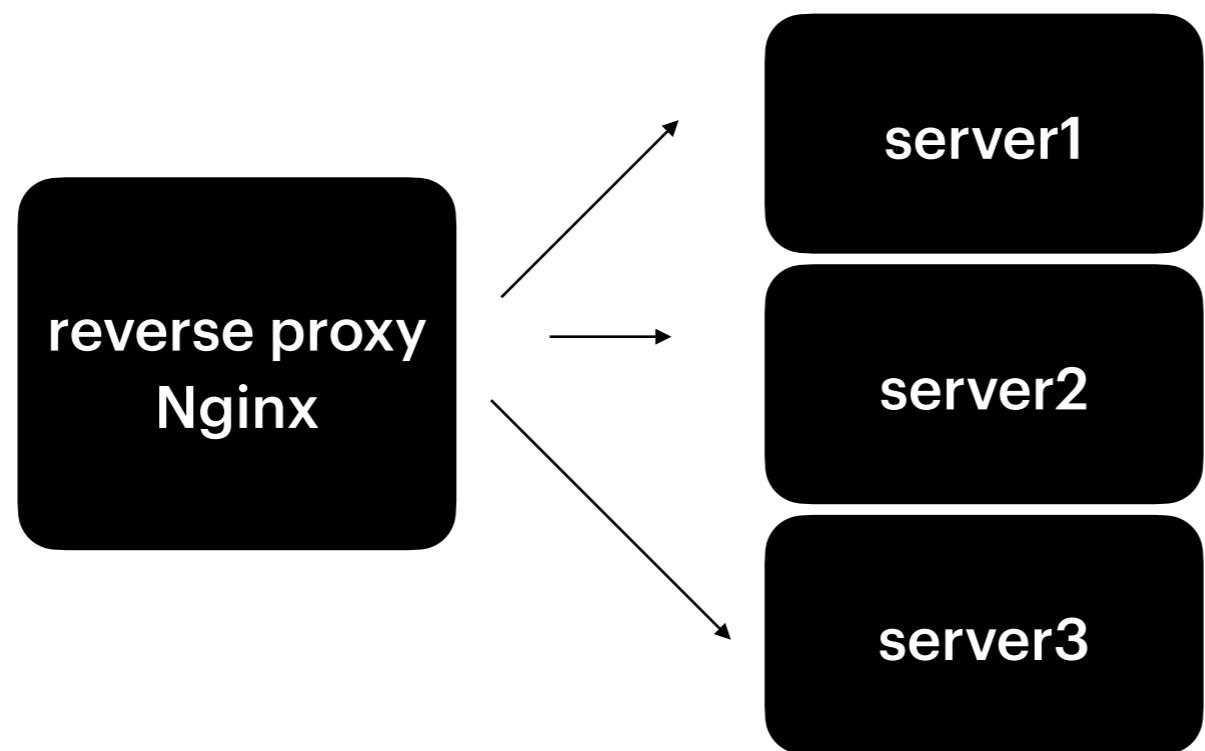
בפרויקטים גדולים:

משתמשים בשרת קבצים ייעודי
משתמשים בCDN

Content Delivery Network

s3 buckets:

שירות שמציאות חברות ענן - דוגמא AMAZON
שמאחסנים קבצים עבור הפרויקט ברשת CDN



שליחת קובץ בJavascript

<https://developer.mozilla.org/en-US/docs/Web/API/FormData/FormData>

שולח בקשה Multipart שולח את הקובץ בחלקים
שהשרת של Django ידע לפענה

תצרכו לבנות טופס רגיל

מהטופס לבנות אובייקט של FormData(form)

את זה תצרפו בfetch או בaxios

Django Permissions

blog/urls.py

הוספה כפторה Login/Logout

```
from django.urls import path, include
from .views import CommentsViewSet, PostsViewSet, UserProfileViewSet, LikesViewSet

from rest_framework.routers import DefaultRouter

router = DefaultRouter()

router.register('comments', CommentsViewSet, basename='comment')
router.register('posts', PostsViewSet, basename='posts')
router.register('user-profile', UserProfileViewSet, basename='user-profile')
router.register('likes', LikesViewSet, basename='likes')

urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls'))
]
```

The screenshot shows the Django REST framework's API browser interface. At the top, there's a navigation bar with the text "Django REST framework!!!". On the right side of this bar is a user dropdown menu labeled "itomers" with a dropdown arrow, and a "Log out" button. Below the navigation bar, the main content area has a title "Api Root" and a subtitle "The default basic root view for DefaultRouter". There are two blue buttons at the bottom right: "OPTIONS" and "GET". To the left of these buttons, there's a link "Api Root". At the very bottom, there's a status bar with the text "HTTP 200 OK" and "Allow: GET, HEAD, OPTIONS".

Django Permissions

המודלים החשובים ביותר הם:

User

: ישות שמתארת את המשתמש: שם משתמש, סיסמה אימייל ועוד.

Permission

: ישות שמתארת יכולת ספציפית:
add_post, change_post, delete_post
add_comment, change_comment, delete_comment

(הרשאות הללו נוצרות אוטומטית ע"י Django כאשרנו מוסיפים למשק הניהול).

Group

: Group

קבוצת של אנשים
עם הרשאות מוגדרות

דג'נו מגדירה את המודלים + יחסינו גומלין
+ מערכת לבדיקה

לדוגמא: ניצור קבוצה users
ונעניק לה ביצוע:

אנחנו נלמד לשימוש בהם

view_post
view_comment
change_comment
delete_comment
add_comment

Django Permissions

Add group

Name:

Permissions:

Available permissions ?

Authentication and Authorization | user | Can view user

Blog | comment | Can add comment

Blog | comment | Can change comment

Blog | comment | Can delete comment

Blog | comment | Can view comment

Blog | post | Can add post

Blog | post | Can change post

Blog | post | Can delete post

Blog | post | Can view post

Blog | post user likes | Can add post user likes

Blog | post user likes | Can change post user likes

Blog | post user likes | Can delete post user likes

Blog | post user likes | Can view post user likes

Blog | user profile | Can add user profile

Hold down "Control", or "Command" on a Mac, to select more than one.

Choose all

Hold down "Control", or "Command" on a Mac, to select more than one.

Django Permissions

Add group

Name:

moderators

Permissions:

Available permissions ?



Filter

- Blog | comment | Can delete comment
- Blog | comment | Can view comment
- Blog | post | Can add post
- Blog | post | Can change post
- Blog | post | Can delete post
- Blog | post | Can view post
- Blog | post user likes | Can add post user likes
- Blog | post user likes | Can change post user likes
- Blog | post user likes | Can delete post user likes
- Blog | post user likes | Can view post user likes
- Blog | user profile | Can add user profile
- Blog | user profile | Can change user profile
- Blog | user profile | Can delete user profile

[Choose all ?](#)

Hold down "Control", or "Command" on a Mac, to select more than one.

[SAVE](#)

[Save and add another](#)

[Save and continue editing](#)

Global Level Security

```
REST_FRAMEWORK = {
    'DEFAULT_RENDERER_CLASSES': [
        'rest_framework.renderers.JSONRenderer',
        'rest_framework.renderers.BrowsableAPIRenderer',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

משתמש חייב להיות מחובר כדי לבצע את כל הפעולות

```
'DEFAULT_PERMISSION_CLASSES': [
    'rest_framework.permissions.AllowAny',
]
```



ברירת מחדל

Authentication VS Authorization

Authentication זיהוי של המשתמש

קיימת מערכת זיהוי מובנית `login` כבר קיימת

Authorization האם מותר למשתמש לבצע פעולה

נctrוך להגדרת הגדרות משלנו

Django Permissions

core/permissions.py

```
from rest_framework import permissions
from rest_framework.permissions import BasePermission

class IsAdminOrModerator(BasePermission):
    # by default has_object_permission returns True
    def has_permission(self, request, view):
        is_admin = (
            request.user and
            request.user.is_authenticated and
            request.user.is_superuser
        )
        if is_admin:
            return True

        in_moderators_group = (
            request.user and
            request.user.is_authenticated and
            request.user.groups.filter(name='moderators').exists()
        )
        return in_moderators_group
```

Django Permissions

blog/views.py

```
from core.utils import try_parse_int
from .serializers import *
from .models import Comment, Post, UserProfile, PostUserLikes
from rest_framework.viewsets import ModelViewSet
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated
from core.permissions import IsAdminOrModerator

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsAdminOrModerator]
```

Django Permissions

core/permissions.py

```
from rest_framework.permissions import BasePermission

class IsOwnerOrReadOnly(BasePermission):
    def has_object_permission(self, request, view, obj):
        # GET is allowed
        if request.method in permissions.SAFE_METHODS:
            return True
        # posts and comments has attribute author
        if hasattr(obj, 'author'):
            return obj.author.user == request.user
        return False
```

Django Permissions

blog/views.py

```
from core.permissions import IsAdminOrModerator, IsOwnerOrReadOnly
from rest_framework.viewsets import ModelViewSet

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrReadOnly]
```

Django Permissions

אפשר להגדיר Permission Classes
פר מетодה:

blog/views.py

```
from rest_framework.permissions import IsAuthenticated
from core.permissions import IsAdminOrModerator, IsOwnerOrReadOnly
from rest_framework.viewsets import ModelViewSet
class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrReadOnly]

    def get_permissions(self):
        if self.actions in ["create", "update", "partial_update", "destroy"]:
            return [IsAdminOrModerator()]
        return [IsAuthenticated()]
```

Django Model Permissions

מגדירים במשק ניהול את הרשאות פר-קובוצה
(ראיינו במשק ניהול איך מגדירים והרשאות)

```
from rest_framework.viewsets import ModelViewSet
from rest_framework.permissions import DjangoModelPermissions

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [DjangoModelPermissions]
```

נכון לעריכת:

- מנהלים יכולים להוסיף וערוך כתבות.
- אדמין יכול לערוך הרשות למשתמשים.
- משתמשים יכולים להציג לערוך תגובות ולמחוק תגובות
- משתמשים יכולים לצפות בposts
- משתמשים יכולים לערוך ולצפות בפרופילים
- משתמשים יכולים לערוך ולצפות בלייקים

Django Model Permissions

core/permissions.py

```
from rest_framework import permissions
from rest_framework.permissions import DjangoModelPermissions, BasePermission

class IsOwnerOrModelPermissions(DjangoModelPermissions):
    # we already have has_permission method in DjangoModelPermissions
    # so we need to override has_object_permission

    def has_object_permission(self, request, view, obj):
        if (
            request.method in permissions.SAFE_METHODS
            and super().has_permission(request, view)
        ):
            return True

        return (
            obj.author.user == request.user
            or request.user.is_superuser
            or request.user.groups.filter(name='moderators').exists()
        )
```

Django Model Permissions

views.py

```
from core.permissions import IsOwnerOrModelPermissions

class CommentsViewSet(ModelViewSet):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer
    permission_classes = [IsOwnerOrModelPermissions]
```

תוכנית עבודה:

1) כל משתמש שנוצר -> יש להוסיף אוטומטית לקובץ המשתמשים

2) משתמש לא יכול לשלוח בבקשת `user_id`
הזהה של המשתמש חייב להכיל מפרטי זהיהו

:Authentication (3

לلمוד מה ברירת המחדל Base Auth

לעבור לשימוש בTokens

לעבור לשימוש בJWTokens

Register/Login/Logout Endpoints

תוכנית עבודה:

כל משתמש שנוצר -> יש להוסיף אוטומטית לקבוצת המשתמשים

האתגר:

משתמש יכול להיווצר ב-
ModelViewSet
או ב-Admin Panel

סטרטגיה לפתרון:
Django Signals

האזנה לאירועים:

<https://docs.djangoproject.com/en/5.1/topics/signals/#connecting-to-signals-sent-by-specific-senders>

<https://docs.djangoproject.com/en/5.1/ref/signals/#post-save>

```
from django.apps import AppConfig
from django.db.models.signals import post_save
from django.dispatch import receiver

# settings for our blog app
class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

    @receiver(post_save, sender='auth.User')
    def perform_add_user_to_users_group(sender, instance, created, **kwargs):
        # the import is nested inside the function
        from django.contrib.auth.models import Group, User
        from blog.models import UserProfile

        if not created:
            return
        group, _ = Group.objects.get_or_create(name='users')
        instance.groups.add(group)
        instance.save()

        UserProfile.objects.get_or_create(user=instance)
        print(f'User {instance.username} added to group {group.name}')
```

Authentication

ברירת המחדל של DRF עבור

היא Basic Authentication

הסיבה - מנגנון פשוט שלא דורש הגדרות.

נכיר קודם Basic Authentication

Basic Authentication

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home' and 'Workspaces'. Below it, a 'Overview' section shows three recent requests. The main area displays a request to 'http://127.0.0.1:8000/api/v1/posts/' using the 'HTTP' method (indicated by a blue arrow icon). The 'Headers' tab is selected, showing a table with one row: 'Authorization' with value 'basic bW9zaGU6YXBhc3N3b3JkYUE=' (the base64 encoded string 'Basic dXNlcjpwYXNzd29yZQ=='). Other tabs like 'Params', 'Body', and 'Cookies' are also visible. At the bottom, there's a JSON preview of the response body.

HTTP <http://127.0.0.1:8000/api/v1/posts/>

GET http://127.0.0.1:8000/api/v1/posts/

Headers (9)

Key	Value
Authorization	basic bW9zaGU6YXBhc3N3b3JkYUE=

Body (10) Test Results

{ } JSON ▾ Preview Visualize

```
1 [  
2 | .{  
3 | | "id": 1,
```

חסרונו של Basic
שלוח את שם המשתמש והסיסמה!

כלומר בצד לקוח יצרכו לשמור
את שם המשתמש והסיסמה

ב Local Storage

ולכן חובה להשתמש ב Token

Token Authentication

```
'DEFAULT_AUTHENTICATION_CLASSES': [  
    'rest_framework.authentication.SessionAuthentication',  
    'rest_framework.authentication.BasicAuthentication',  
    'rest_framework.authentication.TokenAuthentication',  
],
```

1

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

2

```
python manage.py migrate
```

3

Token Authentication

python manage.py runserver

Django administration

Home > Auth Token > Tokens

Start typing to filter...

AUTH TOKEN

Tokens **+ Add**

AUTHENTICATION AND AUTHORIZATION

Groups **+ Add**

Users **+ Add**

BLOG

Comments **+ Add**

Post user likess **+ Add**

Posts **+ Add**

User profiles **+ Add**

HW

Fun facts **+ Add**

TAGGIT

Tags **+ Add**

The Token "5c208a02d0e6265d6a67ce16122b8e3259f384e3" was added successfully.

Select Token to change

Search

Username

KEY

5c208a02d0e6265d6a67ce16122b8e3259f384e3

fc389682c887e46c25fdc865b78a1a7c56003204

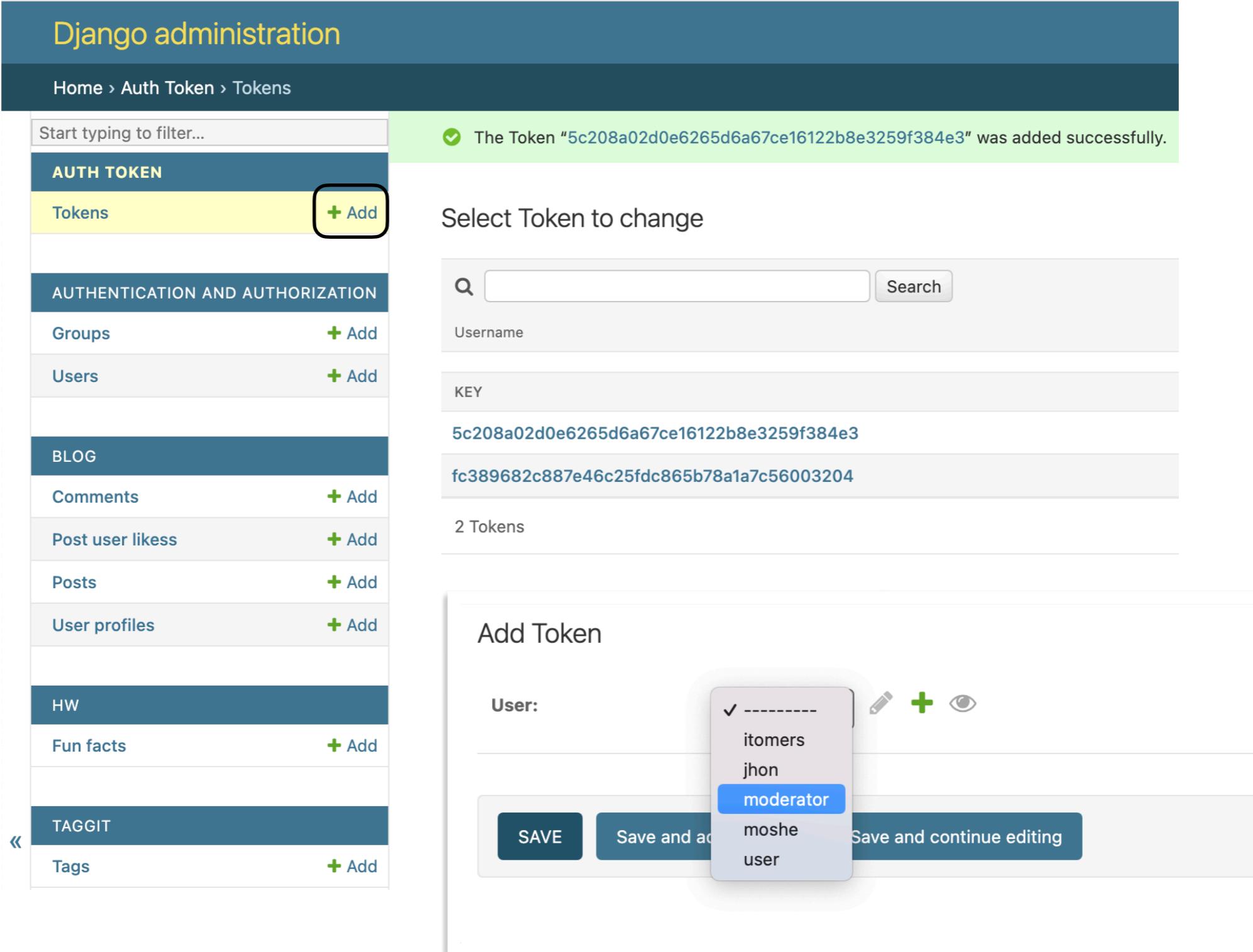
2 Tokens

Add Token

User:

✓ -----
itomers
jhon
moderator
moshe
user

SAVE Save and add Save and continue editing



Token Authentication

הוספה Token לכל המשתמשים שכבר קיימים:

```
python manage.py shell
```

```
from rest_framework.authtoken.models import Token
from django.contrib.auth.models import User

for user in User.objects.all():
    Token.objects.get_or_create(user=user)
```

```
exit()
```

Token Authentication

הוספה Token לכל המשתמשים שבדרך :-)

apps.py

blog > apps.py > BlogConfig > perform_add_user_to_users_group

```
from django.apps import AppConfig
from django.db.models.signals import post_save
from django.dispatch import receiver

# settings for our blog app
class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'

    @receiver(post_save, sender='auth.User')
    def perform_add_user_to_users_group(sender, instance, created, **kwargs):
        # the import is nested inside the function
        from django.contrib.auth.models import Group, User
        from blog.models import UserProfile
        from rest_framework.authtoken.models import Token

        if not created:
            return
        group, _ = Group.objects.get_or_create(name='users')
        instance.groups.add(group)
        instance.save()

        UserProfile.objects.get_or_create(user=instance)
        Token.objects.get_or_create(user=instance)
        print(f'User {instance.username} added to group {group.name}')
```

Token Authentication

הוספה נקודת קצה לLogin

blog/urls.py

```
from rest_framework.authtoken import views as auth_views
```

```
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls')),

    path('login/', auth_views.obtain_auth_token)
]
```

Token Authentication

הוספה נקודת קצה ל:Login

The screenshot shows a POST request to `http://localhost:8000/api/v1/login/`. The request method is `POST`. The body is set to `raw` and contains the following JSON payload:

```
1 {  
2   "username": "itomers",  
3   "password": "123456"  
4 }
```

The response body is shown as JSON, containing a single key-value pair:

```
1 {  
2   "token": "fc389682c887e46c25fdc865b78a1a7c56003204"  
3 }
```

Token Authentication

http://127.0.0.1:8000/api/v1/posts/

Overview

GET http://127.0.0.1:8000/...

GET http://127.0.0.1:8000/...

GET http://127.0.0.1:8000/...

+



http://127.0.0.1:8000/api/v1/posts/

GET



http://127.0.0.1:8000/api/v1/posts/

Params

Authorization

Headers (9)

Body

Scripts

Settings

Headers

8 hidden

	Key	Value	Description
<input checked="" type="checkbox"/>	Authorization	token 0eb5d7243a0d1ff7f08501f00e76d5...	
	Key	Value	Description

Body

Cookies

Headers (10)

Test Results



200 OK

המחבר של Comment או Post ילקח מפרטי זהה

https://www.djangoproject.com/en/2.2/ref/contrib/auth/#django.contrib.auth.models.User.is_authenticated

https://www.djangoproject.com/en/2.2/ref/contrib/auth/#django.contrib.auth.models.User.is_staff

HiddenField

A field class that does not take a value based on user input, but instead takes its value from a default value or **callable**.

Signature: `HiddenField()`

For example, to include a field that always provides the current time as part of the serializer validated data, you would use the following:

```
modified = serializers.HiddenField(default=timezone.now)
```

```
from rest_framework.serializers import CurrentUserDefault
from rest_framework.serializers import HiddenField

class PostSerializer(TaggitSerializer, ModelSerializer):
    tags = TagField(style={'base_template': 'textarea.html'})
    author = HiddenField(default = CurrentUserDefault())
```

המחבר של Post ילקח מפרטי הזיהוי Comment או

<https://www.djangoproject.com/en/1.11/topics/auth-api/#api-guide-fields-hiddenfield>

<https://www.djangoproject.com/en/1.11/topics/auth-api/#api-guide-authentication-currentuserdefault>

```
from rest_framework.serializers import HiddenField

class CurrentUserDefault():
    requires_context = True

    def __call__(self, serializer_field):
        request = serializer_field.context['request']
        return request.user.userprofile

class PostSerializer(TaggitSerializer, ModelSerializer):
    tags = TagField(style={'base_template': 'textarea.html'})
    author = HiddenField(default = CurrentUserDefault())

    class Meta:
        model = Post
        fields = '__all__'
```

אותו דבר ב

:CommentSerializer

```
class CommentSerializer(ModelSerializer):
    author = HiddenField(default = CurrentUserDefault())
    author_id = SerializerMethodField()

    class Meta:
        model = Comment
        fields = '__all__'

    def get_author_id(self, obj):
        return obj.author.id
```

הערה:

אפשר ליצור מחלקה שירשת `ModelSerializer`

עם התכונות:

```
author = HiddenField(default = CurrentUserDefault())
author_id = SerializerMethodField()
```

וחמתודה

ואז לרשת מהמזהה ב-2 serializers שיצרנו

יצירת משתמשים:

רישום משתמש חדש -> פעולה דטה-בייס

בדיקה שהמידע תקין

הצפנה הסיסמא

Serializer
View
Urls

יצירת משתמשים:

serializers.py

```
from rest_framework.serializers import ModelSerializer
from django.contrib.auth.models import User

class UserSerializer(ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email', 'password']
        extra_kwargs = {
            'password': {'write_only': False, 'required': True},
            'id': {'read_only': True},
            'email': {'required': True},
            'username': {'required': True, 'min_length': 3},
        }

    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        return user
```

views.py

```
class UsersViewSet(ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

נסים בשיעור הבא :-)

urls.py

```
from .views import UsersViewSet
router.register('auth', UsersViewSet, basename='auth')
```