Authentication & Permissions in Django REST Framework: Securing Your APIs

🔐 Authentication & Permissions in Django REST Framework: Securing Your APIs

As a backend developer, building APIs is just the beginning. It's equally important to protect them. You don't want unauthorized users accessing sensitive data or modifying resources they shouldn't. That's where **authentication** and **permissions** come into play. Django online courses
In this module, you'll learn the difference between authentication and permission, explore DRF's built-in authentication systems, and implement user login/logout and protected routes using tokens.

🔑 What is Authentication?

Authentication is the process of verifying who the user is. In APIs, this is typically done using a token, session, or credentials sent with the request.
Token Authentication: Each user gets a unique token. They send it in the header of every request.
Session Authentication: Similar to traditional login systems where the session is stored on the server (works well with web apps).
In DRF, token authentication is simple and popular, especially for APIs used by mobile or frontend apps (React, Flutter, etc.).

🚧 Step 1: Setup Token Authentication

```
# settings.py
INSTALLED_APPS = [
    ...
    'rest_framework',
    'rest_framework.authtoken',
]
```

Then run the following to generate the necessary token tables:

```
python manage.py migrate
```

Next, update your REST framework settings:

```
# settings.py
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ]
}
```

Now, let's create a token for each user automatically upon registration.

```
# signals.py
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from rest_framework.authtoken.models import Token

@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_auth_token(sender, instance=None, created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)
```

Connect the signal in your app's `apps.py`:

```
def ready(self):
    import core.signals
```

📄 Step 2: Token Login Endpoint

DRF already provides a ready-to-use login view:

```
# core/urls.py
from rest_framework.authtoken.views import obtain_auth_token

urlpatterns = [
    ...
    path('login/', obtain_auth_token),
]
```

You can test this by sending a POST request with username and password to `/api/login/`. You'll receive a token like:

```
{
  "token": "e1d4e2f123eabc..."
}
```

Save this token and send it with every request as a header:

```
Authorization: Token e1d4e2f123eabc...
```

🔒 What are Permissions?

While authentication confirms who the user is, **permissions** decide what they can do. For example:
Can any logged-in user delete any task?

Should users be allowed to update only their own data?

🧱 **Default DRF Permissions:**

```
# settings.py
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

Now, all API endpoints require a valid token unless you allow them explicitly.

👫 **Common DRF Permission Classes:**

`AllowAny` – Anyone can access the view

`IsAuthenticated` – Only authenticated users allowed

`IsAdminUser` – Only admin users allowed

`IsAuthenticatedOrReadOnly` – Read allowed for all, write allowed for logged-in users

You can set permissions at the view level too:

```
from rest_framework.permissions import IsAuthenticated

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

🛠️ Creating Custom Permissions

Suppose you want only the creator of a task to update or delete it. You can create a custom permission.

```
# core/permissions.py
from rest_framework import permissions

class IsOwner(permissions.BasePermission):
    def has_object_permission(self, request, view, obj):
        return obj.owner == request.user
```

And then apply it to your viewset:

```
from .permissions import IsOwner

class TaskViewSet(viewsets.ModelViewSet):
    ...
    permission_classes = [IsAuthenticated, IsOwner]
```

🔐 Logout (Optional)

To log out, simply delete the token on the client side or use a custom logout view to invalidate it from the server.

✅ Summary

Authentication confirms **who** the user is
Permissions decide **what** the user can do
Token authentication is great for APIs used by mobile/web apps
DRF makes it easy to secure your endpoints with just a few lines
Security is one of the most important aspects of backend development. By securing your API properly, you prevent data leaks, unauthorized access, and abuse of your app's resources.

🚀 Next up, you'll learn how to apply **Filtering, Searching, and Pagination** to make your APIs more flexible and scalable!