Authentication

- BasicAuthentication
- SessionAuthentication
- TokenAuthentication
- RemoteUserAuthentication
- Custom Authentication

TokenAuthentication

This authentication scheme uses a simple token-based HTTP Authentication scheme. Token authentication is appropriate for client-server setups, such as native desktop and mobile clients.

To use the TokenAuthentication scheme you'll need to configure the authentication classes to include TokenAuthentication, and additionally include rest_framework.authtoken in your INSTALLED_APPS setting:

```
INSTALLED_APPS = [
    ...
    'rest_framework.authtoken’
]
```

Note: Make sure to run manage.py migrate after changing your settings. The rest_framework.authtoken app provides Django database migrations.

TokenAuthentication

You'll also need to create tokens for your users.

```
from rest_framework.authtoken.models import Token
token = Token.objects.create(user=...)
print(token.key)
```

For clients to authenticate, the token key should be included in the Authorization HTTP header. The key should be prefixed by the string literal "Token", with whitespace separating the two strings. For example:

```
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

TokenAuthentication

If successfully authenticated, TokenAuthentication provides the following credentials.

request.user will be a Django User instance.

request.auth will be a rest_framework.authtoken.models.Token instance.

Unauthenticated responses that are denied permission will result in an HTTP 401 Unauthorized response with an appropriate WWW-Authenticate header. For example:

WWW-Authenticate: Token

The http command line tool may be useful for testing token authenticated APIs. For example:

http http://127.0.0.1:8000/studentapi/ 'Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b'

TokenAuthentication

Note: If you use TokenAuthentication in production you must ensure that your API is only available over https.

Generate Token

- Using Admin Application
- Using Django manage.py command

python manage.py drf_create_token <username> - This command will return API Token for the given user or Creates a Token if token doesn't exist for user.

- By exposing an API endpoint
- Using Signals

How Client can Ask/Create Token

When using TokenAuthentication, you may want to provide a mechanism for clients to obtain a token given the username and password.

REST framework provides a built-in view to provide this behavior. To use it, add the obtain_auth_token view to your URLconf:

```
from rest_framework.authtoken.views import obtain_auth_token

urlpatterns = [
    path(‘gettoken/', obtain_auth_token)
]
```

The obtain_auth_token view will return a JSON response when valid username and password fields are POSTed to the view using form data or JSON:

```
http POST http://127.0.0.1:8000/gettoken/ username=“name” password=“pass”

{ 'token' : '9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b’ }
```

How Client can Ask/Create Token

It also generates token if the token is not generated for the provided user.

Custom Auth Token

We can customize auth token to provide additional details to the client.

```
from rest_framework.authtoken.views import ObtainAuthToken
from rest_framework.authtoken.models import Token
from rest_framework.response import Response
class CustomAuthToken(ObtainAuthToken):
    def post(self, request, *args, **kwargs):
        serializer = self.serializer_class(data=request.data, context={'request': request})
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        token, created = Token.objects.get_or_create(user=user)
        return Response({
            'token': token.key,
            'user_id': user.pk,
            'email': user.email
        })
```

Generate Token by Signals

If you want every user to have an automatically generated Token, you can simply catch the User's post_save signal.

```
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)
```

Permission Classes

Permissions in REST framework are always defined as a list of permission classes.

- AllowAny
- IsAuthenticated
- IsAdminUser
- IsAuthenticatedOrReadOnly
- DjangoModelPermissions
- DjangoModelPermissionsOrAnonReadOnly
- DjangoObjectPermissions
- Custom Permissions

httpie

HTTPie (pronounced aitch-tee-tee-pie) is a command line HTTP client. Its goal is to make CLI interaction with web services as human-friendly as possible. It provides a simple http command that allows for sending arbitrary HTTP requests using a simple and natural syntax, and displays colorized output. HTTPie can be used for testing, debugging, and generally interacting with HTTP servers.

Syntax:- http [flags] [METHOD] URL [ITEM [ITEM]]

How to Install

pip install httpie

Use httpie

GET Request

http http://127.0.0.1:8000/studentapi/

GET Request with Auth

http http://127.0.0.1:8000/studentapi/ 'Authorization:Token 621cdf999d9151f9aea8e52f00eb436aa680fa24’

POST Request/ Submitting Form

http -f POST http://127.0.0.1:8000/studentapi/ name=Jay roll=104 city=Dhanbad 'Authorization:Token 621cdf999d9151f9aea8e52f00eb436aa680fa24'

Use httpie

PUT Request

http PUT http://127.0.0.1:8000/studentapi/4/ name=Kunal roll=109 city=Bokaro 'Authorization:Token 621cdf999d9151f9aea8e52f00eb436aa680fa24’

Delete Request

http DELETE http://127.0.0.1:8000/studentapi/4/ 'Authorization:Token 621cdf999d9151f9aea8e52f00eb436aa680fa24’