

**By Vitor Freitas**

I'm a passionate software developer and researcher from Brazil, currently living in Finland. I write about Python, Django and Web Development on a weekly basis. [Read more.](#)



TUTORIAL

How to Implement Token Authentication using Django REST Framework

Nov 22, 2018 9 minutes read 22 comments 41,539 views



(Picture: <https://www.pexels.com/photo/door-handle-key-keyhole-279810/>)

In this tutorial you are going to learn how to implement Token-based authentication using Django REST Framework (DRF). The token authentication works by exchanging username and password for a token that will be used in all subsequent requests so to identify the user on the server side.

The specifics of how the authentication is handled on the client side vary **a lot** depending on the technology/language/framework you are working with. The client could be a mobile application using iOS or Android. It could be a desktop application using Python or C++. It could be a Web application using PHP or Ruby.

But once you understand the overall process, it's easier to find the necessary resources and documentation for your specific use case.

Token authentication is suitable for client-server applications, where the token is **safely stored**. You should never expose your token, as it would be (sort of) equivalent of handing out your username and password.

Table of Contents

- o [Setting Up The REST API Project](#) (*If you already know how to start a DRF project you can skip this*)
 - o [Implementing the Token Authentication](#)
 - o [User Requesting a Token](#)
 - o [Conclusions](#)
-

Setting Up The REST API Project

So let's start from the very beginning. Install Django and DRF:

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

Create a new Django project:

```
django-admin.py startproject myapi .
```

Navigate to the **myapi** folder:

```
cd myapi
```

Start a new app. I will call my app **core**:

```
django-admin.py startapp core
```

Here is what your project structure should look like:

```
myapi/
|-- core/
|   |-- migrations/
|   |-- __init__.py
|   |-- admin.py
|   |-- apps.py
|   |-- models.py
|   |-- tests.py
|   +-- views.py
|-- __init__.py
|-- settings.py
|-- urls.py
+-- wsgi.py
manage.py
```

Add the **core** app (you created) and the **rest_framework** app (you installed) to the **INSTALLED_APPS**, inside the **settings.py** module:

myapi/settings.py

```
INSTALLED_APPS = [
    # Django Apps
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third-Party Apps
    'rest_framework',

    # Local Apps (Your project's apps)
    'myapi.core',
]
```

Return to the project root (the folder where the **manage.py** script is), and migrate the database:

```
python manage.py migrate
```

Let's create our first API view just to test things out:

myapi/core/views.py

```
from rest_framework.views import APIView
from rest_framework.response import Response

class HelloView(APIView):
    def get(self, request):
        content = {'message': 'Hello, World!'}
        return Response(content)
```

Now register a path in the **urls.py** module:

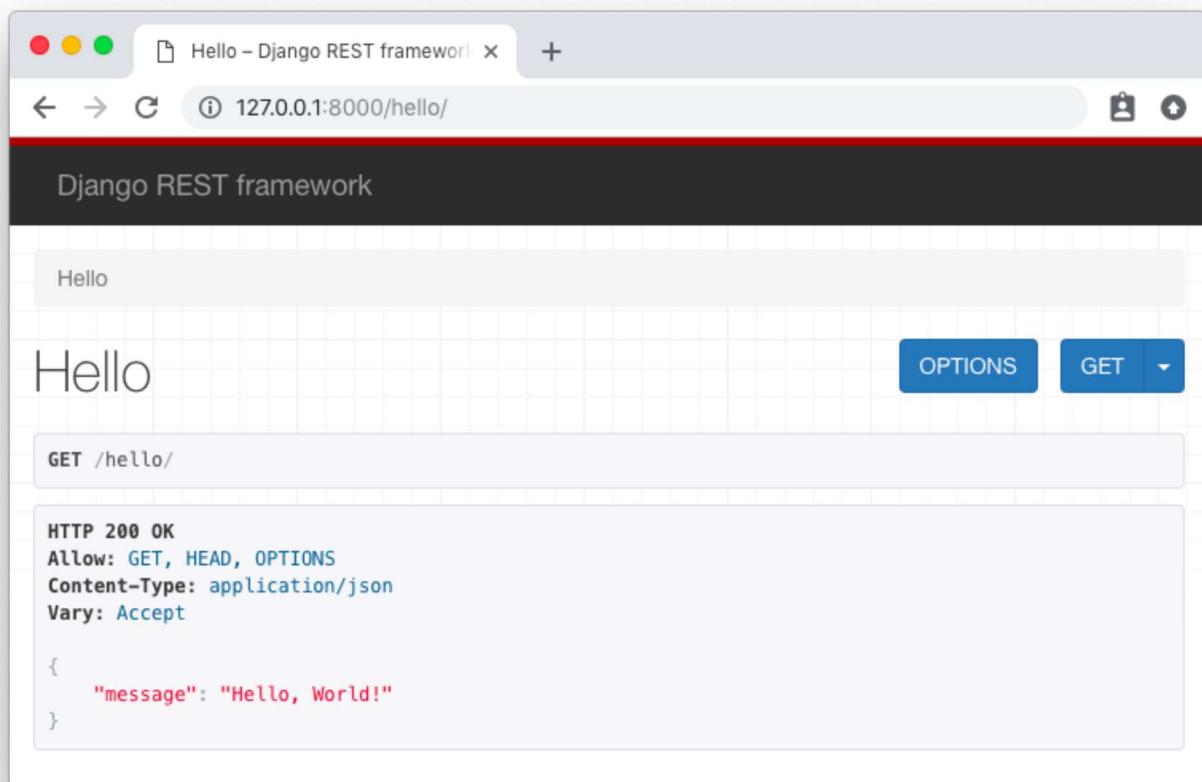
myapi/urls.py

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

urlpatterns = [
    path('hello/', views.HelloView.as_view(), name='hello'),
]
```

So now we have an API with just one endpoint `/hello/` that we can perform `GET` requests. We can use the browser to consume this endpoint, just by accessing the URL

`http://127.0.0.1:8000/hello/`:

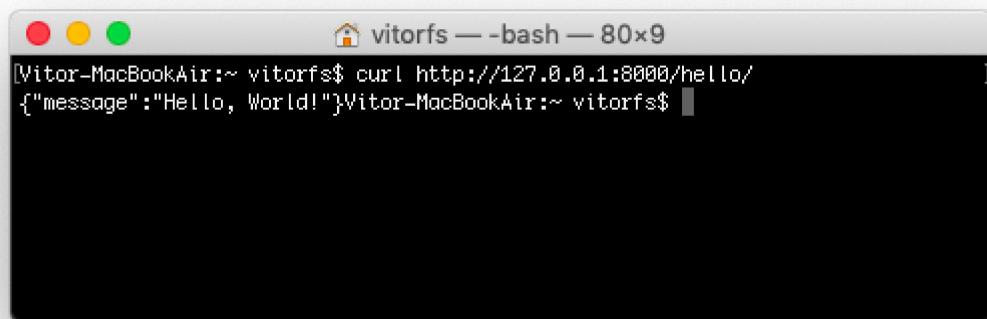


We can also ask to receive the response as plain JSON data by passing the `format` parameter in the querystring like `http://127.0.0.1:8000/hello/?format=json`:



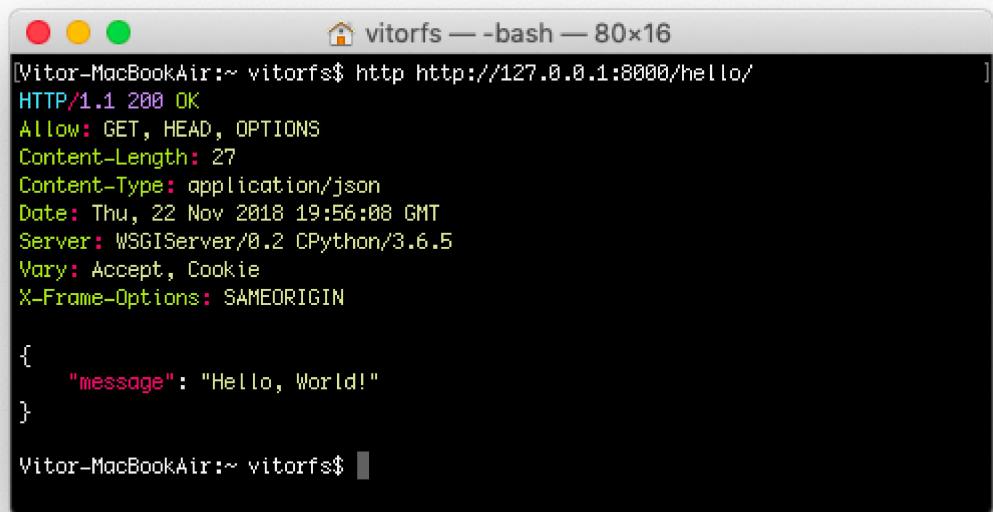
Both methods are fine to try out a DRF API, but sometimes a command line tool is more handy as we can play more easily with the requests headers. You can use [cURL](#), which is widely available on all major Linux/macOS distributions:

```
curl http://127.0.0.1:8000/hello/
```



But usually I prefer to use [HTTPie](#), which is a pretty awesome Python command line tool:

```
http http://127.0.0.1:8000/hello/
```



```
Vitor-MacBookAir:~ vitorfs$ http http://127.0.0.1:8000/hello/
HTTP/1.1 200 OK
Allow: GET, HEAD, OPTIONS
Content-Length: 27
Content-Type: application/json
Date: Thu, 22 Nov 2018 19:56:08 GMT
Server: WSGIServer/0.2 CPython/3.6.5
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "message": "Hello, World!"
}

Vitor-MacBookAir:~ vitorfs$
```

Now let's protect this API endpoint so we can implement the token authentication:

myapi/core/views.py

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated # <-- Here

class HelloView(APIView):
    permission_classes = (IsAuthenticated,) # <-- And here

    def get(self, request):
        content = {'message': 'Hello, World!'}
        return Response(content)
```

Try again to access the API endpoint:

```
http http://127.0.0.1:8000/hello/
```

```
[Vitor-MacBookAir:~ vitorfs$ http http://127.0.0.1:8000/hello/
HTTP/1.1 403 Forbidden
Allow: GET, HEAD, OPTIONS
Content-Length: 58
Content-Type: application/json
Date: Thu, 22 Nov 2018 20:03:09 GMT
Server: WSGIServer/0.2 CPython/3.6.5
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "detail": "Authentication credentials were not provided."
}

Vitor-MacBookAir:~ vitorfs$ ]
```

And now we get an HTTP 403 Forbidden error. Now let's implement the token authentication so we can access this endpoint.

Implementing the Token Authentication

We need to add two pieces of information in our `settings.py` module. First include `rest_framework.authtoken` to your `INSTALLED_APPS` and include the `TokenAuthentication` to `REST_FRAMEWORK`:

`myapi/settings.py`

```
INSTALLED_APPS = [
    # Django Apps
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third-Party Apps
    'rest_framework',
    'rest_framework.authtoken', # <-- Here

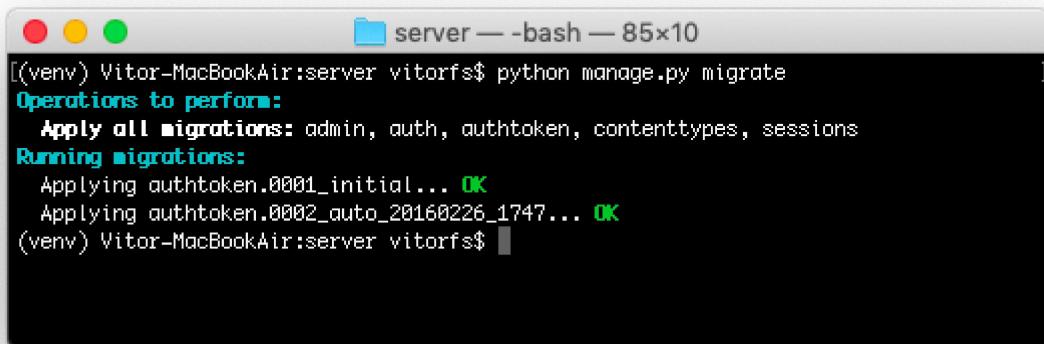
    # Local Apps (Your project's apps)
    'myapi.core',
]

REST_FRAMEWORK = {
```

```
'DEFAULT_AUTHENTICATION_CLASSES': [
    'rest_framework.authentication.TokenAuthentication', # <-- And here
],  
}
```

Migrate the database to create the table that will store the authentication tokens:

```
python manage.py migrate
```



A terminal window titled "server — -bash — 85x10". The command "python manage.py migrate" is run, followed by the output of the migration process. It shows the operations to perform, applying migrations for admin, auth, auth_token, contenttypes, and sessions, and then running migrations for auth_token. Two migrations are applied successfully: "auth_token.0001_initial" and "auth_token.0002_auto_20160226_1747". The terminal ends with "(venv) Vitor-MacBookAir:server vitorfs\$".

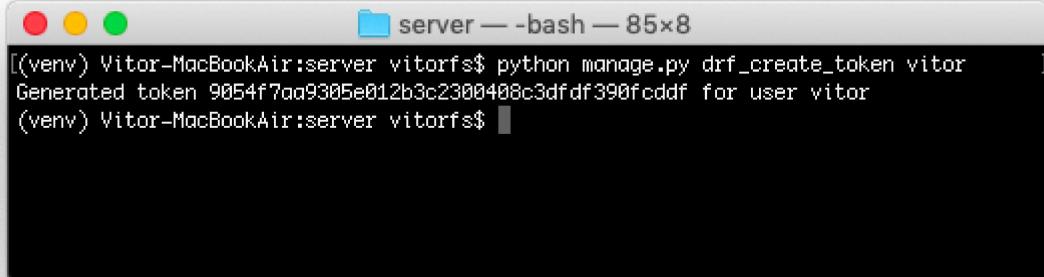
```
[(venv) Vitor-MacBookAir:server vitorfs$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, auth_token, contenttypes, sessions
Running migrations:
  Applying auth_token.0001_initial... OK
  Applying auth_token.0002_auto_20160226_1747... OK
(venv) Vitor-MacBookAir:server vitorfs$ ]
```

Now we need a user account. Let's just create one using the `manage.py` command line utility:

```
python manage.py createsuperuser --username vitor --email vitor@example.com
```

The easiest way to generate a token, just for testing purpose, is using the command line utility again:

```
python manage.py drf_create_token vitor
```



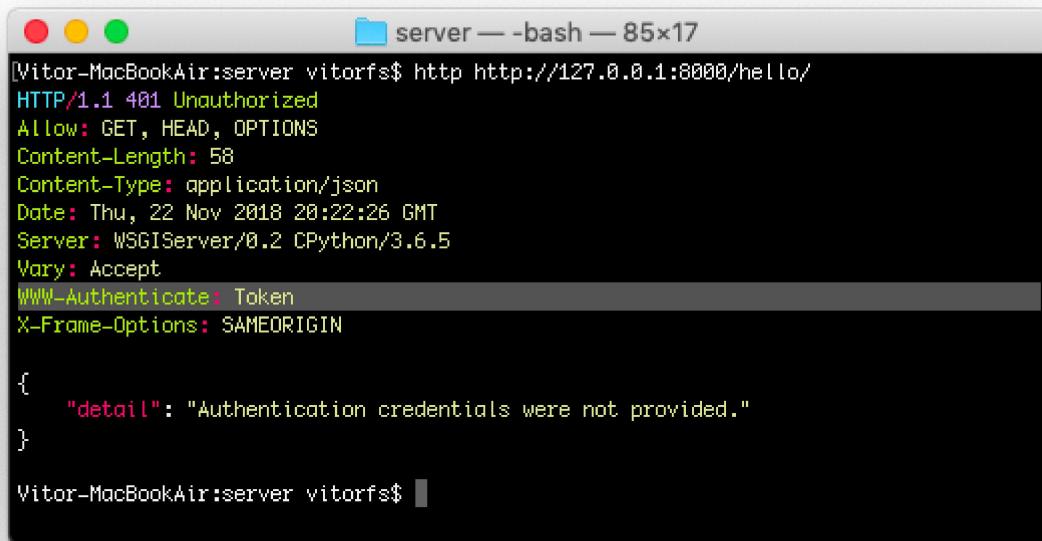
A terminal window titled "server — -bash — 85x8". The command "python manage.py drf_create_token vitor" is run, which generates a token for the user "vitor". The output shows the generated token: "Generated token 9054f7ad9305e012b3c2300408c3dfdf390fcddf for user vitor". The terminal ends with "(venv) Vitor-MacBookAir:server vitorfs\$".

```
[(venv) Vitor-MacBookAir:server vitorfs$ python manage.py drf_create_token vitor
Generated token 9054f7ad9305e012b3c2300408c3dfdf390fcddf for user vitor
(venv) Vitor-MacBookAir:server vitorfs$ ]
```

This piece of information, the random string `9054f7aa9305e012b3c2300408c3dfdf390fcddf` is what we are going to use next to authenticate.

But now that we have the `TokenAuthentication` in place, let's try to make another request to our `/hello/` endpoint:

```
http http://127.0.0.1:8000/hello/
```



```
[Vitor-MacBookAir:server vitorfs$ http http://127.0.0.1:8000/hello/
HTTP/1.1 401 Unauthorized
Allow: GET, HEAD, OPTIONS
Content-Length: 58
Content-Type: application/json
Date: Thu, 22 Nov 2018 20:22:26 GMT
Server: WSGIServer/0.2 CPython/3.6.5
Vary: Accept
WWW-Authenticate: Token
X-Frame-Options: SAMEORIGIN

{
    "detail": "Authentication credentials were not provided."
}
Vitor-MacBookAir:server vitorfs$ ]
```

Notice how our API is now providing some extra information to the client on the required authentication method.

So finally, let's use our token!

```
http http://127.0.0.1:8000/hello/ 'Authorization: Token 9054f7aa9305e012b3c2300408c3dfdf390fcddf'
```



```
[Vitor-MacBookAir:server vitorfs$ http http://127.0.0.1:8000/hello/ 'Authorization: Token 9054f7aa9305e012b3c2300408c3dfdf390fcddf'
HTTP/1.1 200 OK
Allow: GET, HEAD, OPTIONS
Content-Length: 27
Content-Type: application/json
Date: Thu, 22 Nov 2018 20:28:52 GMT
Server: WSGIServer/0.2 CPython/3.6.5
Vary: Accept
X-Frame-Options: SAMEORIGIN

{
    "message": "Hello, World!"
}
Vitor-MacBookAir:server vitorfs$ ]
```

And that's pretty much it. For now on, on all subsequent request you should include the header

`Authorization: Token 9054f7aa9305e012b3c2300408c3dfdf390fcddf`.

The formatting looks weird and usually it is a point of confusion on how to set this header. It will depend on the client and how to set the HTTP request header.

For example, if we were using cURL, the command would be something like this:

```
curl http://127.0.0.1:8000/hello/ -H 'Authorization: Token 9054f7aa9305e012b3c2300408c3dfdf390fcddf'
```

Or if it was a Python [requests](#) call:

```
import requests

url = 'http://127.0.0.1:8000/hello/'
headers = {'Authorization': 'Token 9054f7aa9305e012b3c2300408c3dfdf390fcddf'}
r = requests.get(url, headers=headers)
```

Or if we were using Angular, you could implement an [HttpInterceptor](#) and set a header:

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const user = JSON.parse(localStorage.getItem('user'));
    if (user && user.token) {
      request = request.clone({
        setHeaders: {
          Authorization: `Token ${user.accessToken}`
        }
      });
    }
    return next.handle(request);
  }
}
```

```

        }
    });
}

return next.handle(request);
}
}

```

User Requesting a Token

The DRF provide an endpoint for the users to request an authentication token using their username and password.

Include the following route to the `urls.py` module:

myapi/urls.py

```

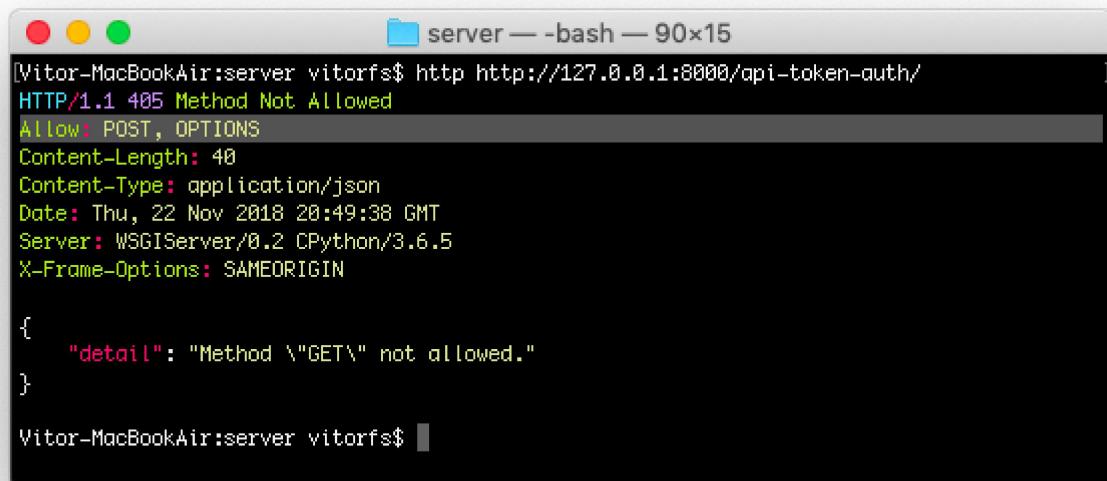
from django.urls import path
from rest_framework.authtoken.views import obtain_auth_token # <-- Here
from myapi.core import views

urlpatterns = [
    path('hello/', views.HelloView.as_view(), name='hello'),
    path('api-token-auth/', obtain_auth_token, name='api_token_auth'), # <-- And here
]

```

So now we have a brand new API endpoint, which is `/api-token-auth/`. Let's first inspect it:

```
http http://127.0.0.1:8000/api-token-auth/
```



```

[Vitor-MacBookAir:server vitorfs$ http http://127.0.0.1:8000/api-token-auth/
HTTP/1.1 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Length: 40
Content-Type: application/json
Date: Thu, 22 Nov 2018 20:49:38 GMT
Server: WSGIServer/0.2 CPython/3.6.5
X-Frame-Options: SAMEORIGIN

{
    "detail": "Method \"GET\" not allowed."
}

Vitor-MacBookAir:server vitorfs$ ]

```

It doesn't handle GET requests. Basically it's just a view to receive a POST request with username and password.

Let's try again:

```
http post http://127.0.0.1:8000/api-token-auth/ username=vitor password=123
```



A screenshot of a terminal window titled "server — bash — 109x15". The window shows the command "http post http://127.0.0.1:8000/api-token-auth/ username=vitor password=123" being run. The response output is:
HTTP/1.1 200 OK
Allow: POST, OPTIONS
Content-Length: 52
Content-Type: application/json
Date: Thu, 22 Nov 2018 20:52:42 GMT
Server: WSGIServer/0.2 CPython/3.6.5
X-Frame-Options: SAMEORIGIN

{
 "token": "9054f7aa9305e012b3c2300408c3dfdf390fcddf"
}
Vitor-MacBookAir:server vitorfs\$

The response body is the token associated with this particular user. After this point you store this token and apply it to the future requests.

Then, again, the way you are going to make the POST request to the API depends on the language/framework you are using.

If this was an Angular client, you could store the token in the `localStorage`, if this was a Desktop CLI application you could store in a text file in the user's home directory in a dot file.

Conclusions

Hopefully this tutorial provided some insights on how the token authentication works. I will try to follow up this tutorial providing some concrete examples of Angular applications, command line applications and Web clients as well.

It is important to note that the default Token implementation has some limitations such as only one token per user, no built-in way to set an expiry date to the token.

You can grab the code used in this tutorial at github.com/sibtc/drf-token-auth-example.

Related Posts



[How to Save Extra Data to a Django REST Framework Serializer](#)



[How to Use JWT Authentication with Django REST Framework](#)



[Django Authentication Video Tutorial](#)

[django](#) [drf](#) [rest](#) [api](#) [auth](#)

Share this post



22 Comments [Simple is Better Than Complex](#)

[Login](#) ▾

[Recommend](#) 9

[Tweet](#)

[Share](#)

[Sort by Best](#) ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)

Name



[Diego Tejada](#) • 8 months ago

A very clear and detailed explanation, thank you so much!

I'm wondering how to configure token expiration time and refresh tokens to use the same way as shown here.

5 ⤵ | ⤴ • Reply • Share >



[Jan Oleszyński](#) • a month ago

Hello ! Thanks for tutorial of TokenAuthentication after after reading I understood where I was making a mistake. Greetings !

^ | ⤴ • Reply • Share >



[Vasconcelos](#) • 3 months ago

It's very nice. This example works to me

^ | ⤴ • Reply • Share >



ahmad • 3 months ago

Hi to do that 'http post http://127.0.0.1:8000/api-token-auth/ username=vitor password=123' in out django [views.py](#) code?

^ | v • Reply • Share ›



zati • 5 months ago

why when i add 'DEFAULT_AUTHENTICATION_CLASSES': [
'rest_framework.authentication.TokenAuthentication',
,
. log in on django rest framework become problem?

^ | v • Reply • Share ›



Hendi Santika • 5 months ago

A very clear and detailed explanation, thank you so much!

I was having an issue.

I already run :

```
python3 manage.py createsuperuser --username naruto --email naruto@example.com
```

then hit the request :

```
http http://127.0.0.1:8000/hello/ 'Authorization: Token  
874cd3a3a8c29d0ff5fb595fb01aeec5e2f84b9'
```

But, it always said that : 403 / forbidden

HTTP/1.1 403 Forbidden

Allow: GET, HEAD, OPTIONS

Content-Length: 58

Content-Type: application/json

Date: Wed, 20 Feb 2019 22:37:10 GMT

Server: WSGIServer/0.2 CPython/3.7.2

Vary: Accept, Cookie

X-Frame-Options: SAMEORIGIN

```
{  
"detail": "Authentication credentials were not provided."  
}
```

I already create username & password.

Why is that happening?

Thanks

^ | v • Reply • Share ›



Hendi Santika → Hendi Santika • 5 months ago

My repo is here : <https://github.com/hendisan...>

^ | v • Reply • Share ›



aman kh • 5 months ago

imagine user(in android phone)enter her username and password. you response to her a token?with your tutorial(mabe all) need two step for [login.how](#) i resolve in??

^ | v • Reply • Share >



Thomas Lee Sebastian • 6 months ago

Thank you for the lucid explanation. Could follow through the steps easily

^ | v • Reply • Share >



Ado Erkocevic • 7 months ago

How secure is this solution to have as a auth feature for an ios app (I am considering to implement it to my ios+webapp system)? Is data sent as plain text and can https be used to encrypt the authentication?

^ | v • Reply • Share >

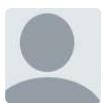


ali reza • 7 months ago

TNXXXXXXXXXXXXXX.

tnx a lot.

^ | v • Reply • Share >



Douglas Petrin Bertini • 7 months ago

Nice one Vitor, Thanks!

^ | v • Reply • Share >



venkatesh • 7 months ago

can you please tell me how to use jwt token in apiview

^ | v • Reply • Share >



Dimple Mathew • 8 months ago

Hi, whenever I am running the command http http://127.0.0.1/hello/hello 'Aut...' I am getting this:

http http://127.0.0.1:8000/hello/ 'Authorization: Token

c9a5b630b1887efdd3ca2db82aae9cec2c44257e'

usage: http [--json] [--form] [--pretty {all,colors,format,none}]

[--style STYLE] [--print WHAT] [--headers] [--body] [--verbose]

[--all] [--history-print WHAT] [--stream] [--output FILE]

[--download] [--continue]

[--session SESSION_NAME_OR_PATH | --session-read-only SESSION_NAME_OR_PATH]

[--auth USER[:PASS]] [--auth-type {basic,digest}]

[--proxy PROTOCOL:PROXY_URL] [--follow]

[--max-redirects MAX_REDIRECTS] [--timeout SECONDS]

[--check-status] [--verify VERIFY]

[--ssl {ssl2.3,ssl3,tls1,tls1.1,tls1.2}] [--cert CERT]

[--cert-key CERT_KEY] [--ignore-stdin] [--help] [--version]

[--traceback] [--default-scheme DEFAULT_SCHEME] [--debug]

[METHOD] URL [REQUEST_ITEM [REQUEST_ITEM ...]]

http: error: argument REQUEST_ITEM: "Token" is not a valid value

^ | v • Reply • Share >

Vitor Freitas Mod **Dimple Mathew** • 8 months ago



Vitor Freitas Mod 7 months ago

Make sure you have quotes around the Authorization: Token

```
c9a5b630b1887efdd3ca2db82aae9cec2c44257e
```

You can try with double quotes:

```
http http://127.0.0.1:8000/hello/ "Authorization: Token c9a5b630b1887efdd3ca2db82aae9cec2c44257e"
```

4 ^ | v • Reply • Share >



dimple mathew → Vitor Freitas 8 months ago

hey thank you double quotes worked.

1 ^ | v • Reply • Share >



Cuahutli M. Ulloa 8 months ago

Amazing.... is a great tutorial vitor, thank you so much. Only have a one question.

This tutorial is about auth-token, is the same to JWT Token implementation?

^ | v • Reply • Share >



Vitor Freitas Mod → Cuahutli M. Ulloa 8 months ago

The concept is about the same, but I will follow up this tutorial with an example using JWT with Django REST Framework. JWT provide some important security features

5 ^ | v • Reply • Share >



Cuahutli M. Ulloa → Vitor Freitas 8 months ago

Yeah, ok vitor, i'll waiting for this

^ | v • Reply • Share >



Paul Rogers → Cuahutli M. Ulloa 8 months ago

You can use this as a reference to implement JWT: <https://thinkster.io/tutor...>

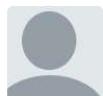
1 ^ | v • Reply • Share >



Cuahutli M. Ulloa → Paul Rogers 8 months ago

Thats great, Thank you so much Paul, i'll see the reference.

1 ^ | v • Reply • Share >



ravindar sharma 2 months ago

Hi i want to implement normal user and admin user functionality whereas front end is written in angular and backend is django...any help is appreciated...

^ | v • Reply • Share >

ALSO ON SIMPLE IS BETTER THAN COMPLEX

How to Use JWT Authentication with Django REST Framework

15 comments • 7 months ago

 Philip J Cox — After further digesting and Avatar going over this blog post, it really tells us nothing. There is so much missing to actually

<https://simpleisbetterthancomplex.com/tutorial/2018/11/22/how-to-implement-token-authentication-using-django-rest-framework.html>

How to Use Date Picker with Django

21 comments • 7 months ago

 Vitor Freitas — The live example side by side Avatar is just using `$("#datetimepicker1").datetimepicker();` where

Launching our Community Forum

4 comments • 8 months ago

 Trung Bát — Great guide, thanks for sharing.
Avatar

Django Tips #22 Designing Better Models

24 comments • a year ago

 Evgenii Fedulov — Cool! Great example about
Avatarrelated name, it has to be added to official
django tutorials.

Subscribe to our Mailing List

Receive updates from the Blog!

Your email address

SUBSCRIBE

Popular Posts



[How to Extend Django User Model](#)



[How to Setup a SSL Certificate on Nginx for a Django Application](#)



[How to Deploy a Django Application to Digital Ocean](#)