

✦ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Implementing Google Login With JWT in Django for RESTful API Authentication



Gerry Sabar · [Follow](#)

4 min read · Aug 7, 2019



356

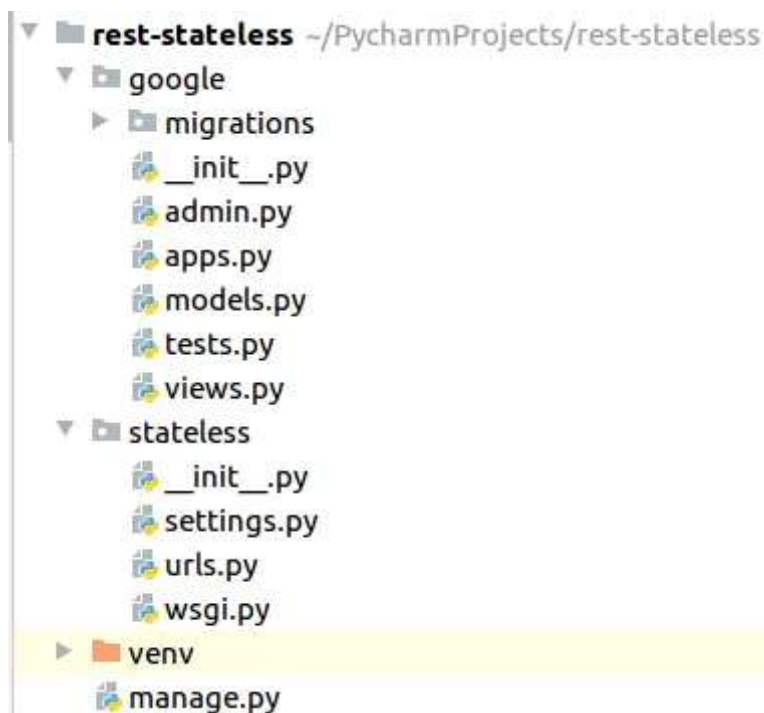


3



It's become common now implementing RESTful API since server can communicate with various devices to communicate each other. Using 3rd party login is not the exception. Nowadays it's pretty common the application support login by Google for authentication either from mobile device, PC and so on. The problem is how we authenticate access without hold any session value which is called as statelessness in RESTful API? This short article will introduce you on how to implement it easily. First of all I assume you understand and able to created a basic Django project.

I myself for this article create a Django project named rest-stateless followed by created a Django app named google, therefore my directory structure will be like this (feel free to use this approach or another approach if you like to do so to organize your working directory structure):



Now we need to install Django Rest Framework to implement RESTful API in our application

```
$ pip install djangorestframework
```

inside `INSTALLED_APPS` in `stateless/settings.py` add `'rest_framework'` so will be like this

```
INSTALLED_APPS = [  
    'google', # your newly created app  
    'rest_framework', # add this  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

at terminal inside your root project directory, let's run a migration

```
$ python manage.py migrate
```

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

`google/views.py`

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

register a path in the `urls.py`:

stateless/urls.py

```
from django.contrib import admin
from django.urls import path
from django.urls import path
from google import views

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

now we can try to access <http://localhost:8000/hello> to see the result as follow:



Great, we just implemented the basic infrastructure for our RESTful API and now we're going to implement jwt auth and using google token for authentication. Let's install django-rest-framework-simplejwt

```
$ pip install djangoRESTframework_simplejwt
```

then install requests

```
$ pip install requests
```



 Search Medium

 Write



add this code in **settings.py**

```
REST_FRAMEWORK = {  
    'DEFAULT_AUTHENTICATION_CLASSES': [  
        'rest_framework_simplejwt.authentication.JWTAuthentication',  
    ],  
}
```

Let's modify **google/views.py** become like this:

```

1  from django.contrib.auth.base_user import BaseUserManager
2  from django.contrib.auth.hashers import make_password
3  from rest_framework.utils import json
4  from rest_framework.views import APIView
5  from rest_framework.response import Response
6  import requests
7  from rest_framework_simplejwt.tokens import RefreshToken
8  from django.contrib.auth.models import User
9
10 class HelloView(APIView):
11     def get(self, request):
12         content = {'message': 'Hello, World!'}
13         return Response(content)
14
15
16 class GoogleView(APIView):
17     def post(self, request):
18         payload = {'access_token': request.data.get("token")} # validate the token
19         r = requests.get('https://www.googleapis.com/oauth2/v2/userinfo', params=payload)
20         data = json.loads(r.text)
21
22         if 'error' in data:
23             content = {'message': 'wrong google token / this google token is already expired.'}
24             return Response(content)
25
26         # create user if not exist
27         try:
28             user = User.objects.get(email=data['email'])
29         except User.DoesNotExist:
30             user = User()
31             user.username = data['email']
32             # provider random default password
33             user.password = make_password(BaseUserManager().make_random_password())
34             user.email = data['email']
35             user.save()
36
37         token = RefreshToken.for_user(user) # generate token without username & password
38         response = {}
39         response['username'] = user.username
40         response['access_token'] = str(token.access_token)
41         response['refresh_token'] = str(token)
42         return Response(response)

```

Finally in `stateless/urls.py` we modify as follow:

```
1  from django.contrib import admin
2  from django.urls import path
3  from google import views
4
5  urlpatterns = [
6      path('admin/', admin.site.urls),
7      path('hello/', views.HelloView.as_view(), name='hello'),
8      path('google/', views.GoogleView.as_view(), name='google'), # add path for google authentication
9  ]
```

urls.py hosted with ❤ by GitHub

[view raw](#)

Cool, we have created a method to authenticate but how to test it? let's go to <https://developers.google.com/oauthplayground/> and we're going to create a valid google token to be tested in our application. At input your own scopes let's add this:

<https://www.googleapis.com/auth/userinfo.email>

So maybe in the future you can grab email when made request. For further detail about scope you can visit this [page](#)

← → ↻ 🔒 https://developers.google.com/oauthplayground/

Google Developers

OAuth 2.0 Playground ×

▼ Step 1 Select & authorize APIs

Select the **scope** for the APIs you would like to access or input your own OAuth scopes below. Then click the "Authorize APIs" button.

- ▶ Abusive Experience Report API v1
- ▶ Access Approval API v1beta1
- ▶ Access Context Manager API v1
- ▶ Ad Exchange Buyer API II v2beta1
- ▶ Ad Exchange Buyer API v1.4
- ▶ Ad Experience Report API v1
- ▶ AdSense Host API v4.1
- ▶ AdSense Management API v1.4
- ▶ Admin Reports API reports_v1
- ▶ Analytics Reporting API v4
- ▶ Android Management API v1
- ▶ App Engine Admin API v1
- ▶ Apps Script API v1
- ▶ BigQuery API v2
- ▶ BigQuery Data Transfer API v1
- ▶ Binary Authorization API v1beta1
- ▶ Blogger API v3
- ▶ Books API v1
- ▶ Calendar API v3

Authorize APIs

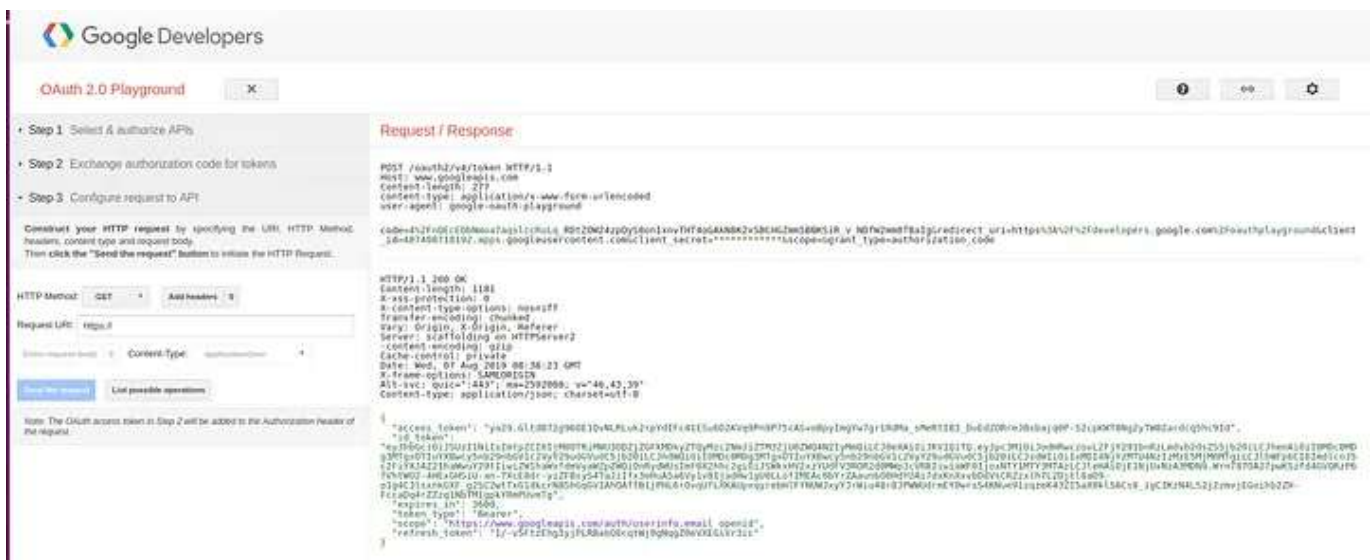
Request / Response

No request.

▼ Step 2 Exchange authorization code for tokens

▼ Step 3 Configure request to API ☐ Wrap Lines

click Authorize APIs and Sign in using your own Google account followed by clicking Exchange authorization code for tokens. We got something like this:



now we can copy access token and test our application using curl as follow:

```
curl -X POST \
  http://localhost:8888/google/ \
  -H 'Content-Type: application/json' \
  -d '{
    "token": "__PASTE YOUR ACCESS TOKEN HERE__"
  }'
```

Cool! We have implemented the token. Now depends on your needs you can keep using this token for authentication or create a new user then grant a new jwt access token.

For testing purpose we can try to implement authentication to localhost:8000/hello/ , let's modify a bit by adding

```
#permission_classes = (IsAuthenticated,)
```

below “class HelloView(APIView):” so the code will be like this

```
class HelloView(APIView):
    permission_classes = (IsAuthenticated,)

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

also don't forget to import the library on top

```
from rest_framework.permissions import IsAuthenticated
```

last in settings.py we need to tell Django that we're using jwt authentication

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ],
}
```

Now if you access localhost:8000/hello/ you'll get error message

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

Therefore we need to make request using curl by also adding access token we create from /google endpoint, so the curl will be like this:

```
curl -X GET \
  http://localhost:8888/hello/ \
```

```
-H 'Authorization: Bearer PUT_ACCESS_TOKEN_HERE'
```

If you executed this command you can see message “Hello World!” again. Congratulation we have accomplished authentication from google to our RESTful API in Django.

Django

Authentication

Google

Oauth

Rest Api



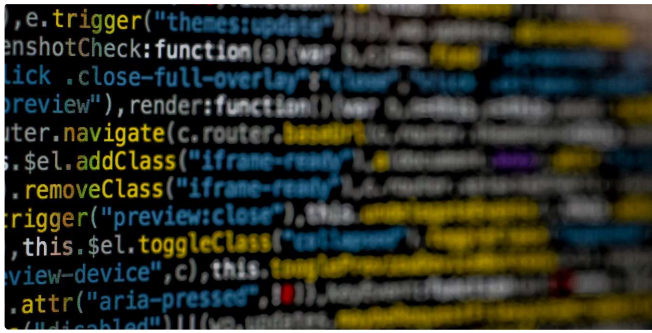
Written by Gerry Sabar

Follow

114 Followers

Full stack web developer. www.linkedin.com/in/gerrysabar

More from Gerry Sabar



 Gerry Sabar

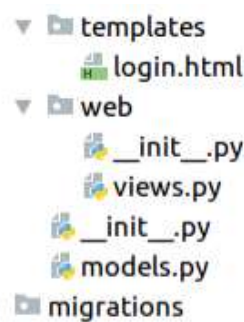
Implementing UUID for Model User in Django & PostgreSQL

The default id structure for Django model is using auto-increment integer. For example...

2 min read · Aug 6, 2019



40



 Gerry Sabar

Organize Flask Code Professionally

The common tutorial for running Flask microframework is as follow:

6 min read · Jul 18, 2019



2



 Gerry Sabar

Deploying Django App With Docker & PostgreSQL To Digital Ocean

From my previous article here, now we can try to deploy it to Digital Ocean VPS Droplet....

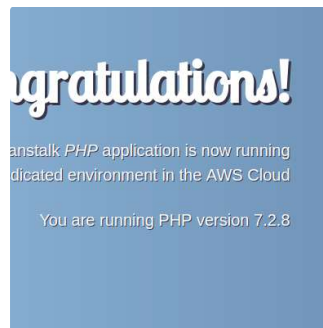
3 min read · Jul 25, 2019



15



1



 Gerry Sabar

Deploying PHP application to AWS Beanstalk for Beginner

AWS Beanstalk is getting familiar for web-development nowadays. Unfortunately...

5 min read · Oct 17, 2018



4



1



What's Next?

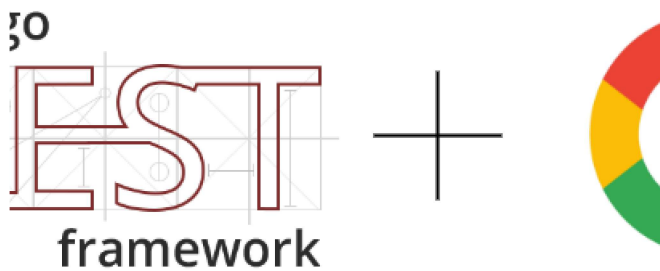
- [AWS Elastic Beanstalk overview](#)
- [Deploying AWS Elastic Beanstalk Applications in PHP](#)
- [Using Amazon RDS with PHP](#)
- [Customizing the Software on EC2 Instances](#)
- [Customizing Environment Resources](#)

AWS SDK for PHP

- [AWS SDK for PHP home](#)
- [PHP developer center](#)
- [AWS SDK for PHP on GitHub](#)

See all from Gerry Sabar

Recommended from Medium



elijah samson in AWS Tip

Integrating Google Login into your Django API without using any...

Sure, third-party packages like drf-social-oauth2 can be handy for handling OAuth. Bu...

5 min read · Jul 12



9



Gokul nath

JWT Authentication in Django API: A QuickStart Guide

When you develop a Restful API using awesome Django Rest Framework, often you...

2 min read · Sep 19



2



Lists



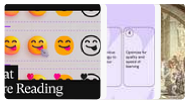
Coding & Development

11 stories · 200 saves



Leadership upgrades

7 stories · 29 saves



Stories to Help You Grow as a Designer


11 stories · 309 saves



Tech & Tools

15 stories · 56 saves



 Ritik khandelwal

JWT Authentication in the Frontend: Enhancing Security in...

In this blog, we will learn how we can use the JWT token in our front end to authenticate th...

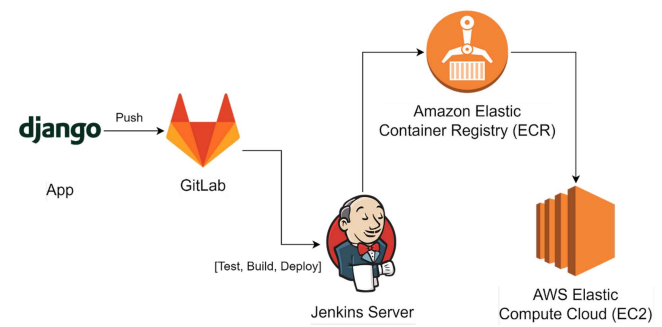
6 min read · Jul 31



 Manish Sharma

Django REST Framework : JWT, Custom User Role

Recently I had the opportunity to create a Consulting Application using DRF (Django...

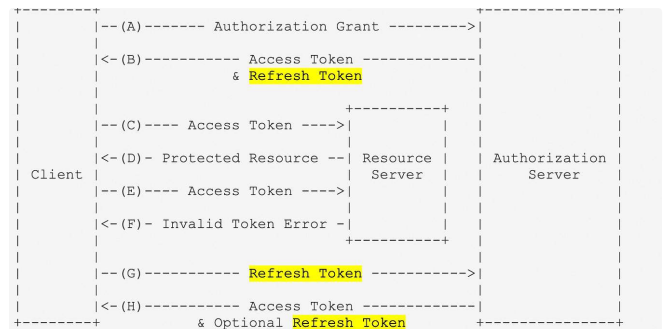


 Chinedu Olebu

Building a Complete CI/CD Pipeline for a Django Project with...

Introduction

8 min read · Jun 30



 Wei Xu in Stackademic

JWT Refresh Token

What Is A Token?

4 min read · Jul 4

 17



 +



3 min read · May 19

 27



 +



See more recommendations