**JSON web token based authentication in Django**

[](https://medium.com/@jyotigautam108?source=post_page-----b6dcfa42a332----------------------)

[Jyoti Gautam](https://medium.com/@jyotigautam108?source=post_page-----b6dcfa42a332----------------------)

[Oct 13, 2017](https://medium.com/python-pandemonium/json-web-token-based-authentication-in-django-b6dcfa42a332?source=post_page-----b6dcfa42a332----------------------) · 3 min read

A brief description of how the JWT authentication is implemented in Django …

POINTS TO COVER-

* What is JWT and how it works?
* Using JWT authentication in Django

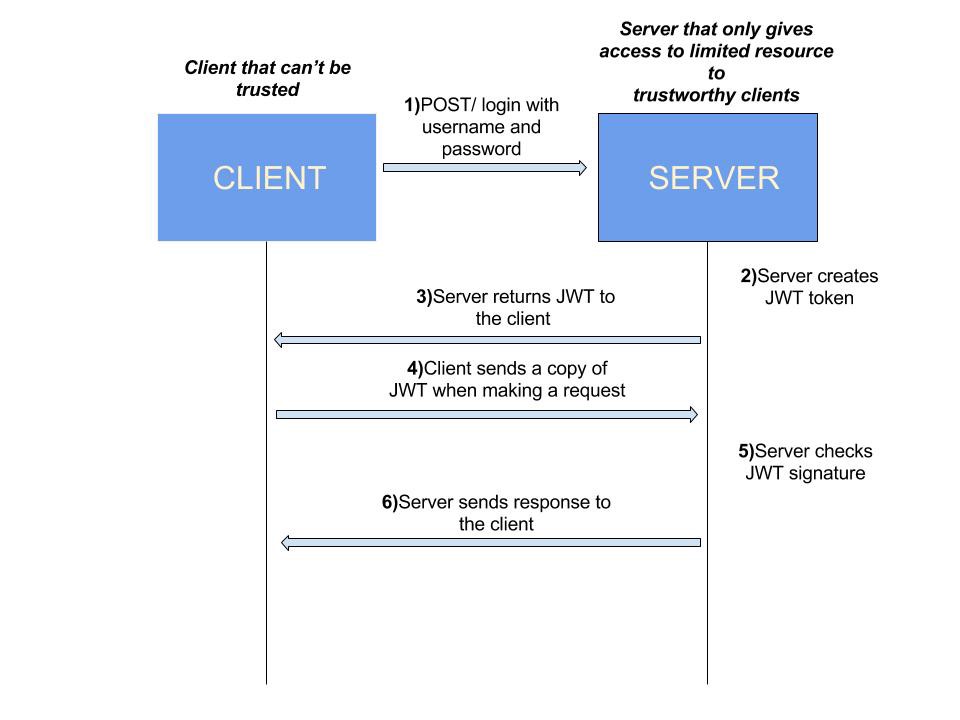
**JSON Web Token (also pronounced as jot):**

Conventionally websites used cookie-based authentication which was stateful i.e, both the client and the server had the session information stored on them but in a typical REST framework, the server does not keep client’s state, unlike cookie-based authentication.Hence, the token-based authentication which is stateless comes into the picture where the client keeps the information required for authentication in the form of a token and there’s no need for server-side storage.While there are many ways to create tokens, JWTs have become a standard for making authentication token. Before diving into authentication in Django let’s first demystify what actually JWT is and how JWT authentication works.

**How it works ?-**

Let’s take a real-life scenario where a customer wants to enter a mall. Just before the entrance he has to keep his luggage at the counter outside the mall where he can get a space for his luggage after he receives a token which contains the information of the slot where his luggage(i.e, his resource) is kept. Next time he comes he has to show this token to get the item stored in the slot mentioned in the token.  
Similarly, token-based authentication is used to access REST APIs and here’s the diagram showing how JWT is used to implement the same:-

https://miro.medium.com/max/60/1*l-FS80RhxUgjZOKGgOXnTQ.jpeg?q=20



**Why JWT ?**

* JWTs work across different programming languages like .net, python, nodejs, java, php.
* Used to securely communicate JSON objects over unsecured connections like HTTP.
* JWTs are self-contained entities i.e, they can carry all the basic information about themselves, the user they are associated to as payload and the signature.
* Perfect for use inside an HTTP header when authenticating an API.

**JWT’s Structure-**

JSON Web Token comprises 3 strings separated by “.” as follows where each part is encoded with [base64url](https://en.wikipedia.org/wiki/Base64) encoding :

“eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjp7ImlkIjoiNTlhZDFmZTI0MDVkNzk0YTFkYWQ2YmFkIiwiZGlzcGxheV9uYW1lIjoiQWRtaW4iLCJyb2xlX3R5cGUiOiJhZG1pbiJ9LCJpZCI6IlwiNTliYmJjODc0MDVkNzk0NjYwNGEzZjUyXCIiLCJlbWFpbCI6Imp5b3RpZ2F1dGFtMTA4QGdtYWlsLmNvbSJ9.oGA-goFi7ee6DdKn0Z4sctomaY6Ki0mfuJfxT4OK9WA”

https://miro.medium.com/max/60/1*TSckCFJw_yc-buFIUHFBUg.jpeg?q=20



**The HEADER portion** :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

A header is a JSON object which declares that the encoded object is a JWT token which is MACed using the HMAC SHA-256 algorithm in the following manner:

* { "**typ**":"JWT", "**alg**":"HS256" }

Here,typ and alg are the strings and

-“typ” specifies the type of token used ,

-“alg” denotes the type of algorithm used to sign the token.

**The PAYLOAD portion or JWT Claim-set** :

eyJyb2xlIjp7ImlkIjoiNTlhZDFmZTI0MDVkNzk0YTFkYWQ2YmFkIiwiZGlzcGxheV9uYW1lIjoiQWRtaW4iLCJyb2xlX3R5cGUiOiJhZG1pbiJ9LCJpZCI6IlwiNTliYmJjODc0MDVkNzk0NjYwNGEzZjUyXCIiLCJlbWFpbCI6Imp5b3RpZ2F1dGFtMTA4QGdtYWlsLmNvbSJ9

A payload is a JSON object that consists of user-defined attributes(called public claims). There are some pre-defined attributes in the standard also known as reserved claims.

**JWT Signature :**

oGA-goFi7ee6DdKn0Z4sctomaY6Ki0mfuJfxT4OK9WA

JWT signature is the hash of following components:

* header
* payload
* “SECRET” ( *A signature held by server* )

i.e, signature is an encoded header and payload signed with a “SECRET”

This is how we get it:-

*var* **encoded\_string** = *base64URLEncode*(header)+”.”+*base64URLEncode*(payload)

Signature = HMACSHA256(**encoded\_string,”SECRET”**)

**JWT** in short is :-encoded(**header**)+encoded(**payload**)+**signature**(that is already encoded).

***Creating JSON Web token in python :-***

First we have to install Python pyjwt library and then using pyjwt:

>>> import jwt>>> encoded\_token = jwt.encode({‘user\_id’: “abc”}, ‘SECRET’, algorithm=’HS256')>>> encoded\_token  
‘eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoiYWJjIn0.OW6BZboviYgO6Yy\_UTj5jloba7WlPwZnKHPYDUyY3MU’

Decoding the above created token on server:

>>> jwt.decode(encoded\_token, ‘SECRET’, algorithms=[‘HS256’])  
{’user\_id’: ’abc’}

**Using JWT authentication in Django:**

Implementing JWT authentication in Django basically requires following three steps:

1. A Login view that takes user’s username and password and creates a token with user information corresponding to the passed credentials as payload and returns it to the browser.

2. Next time browser makes any API request it has to send the token in Auth Headers to authenticate itself.

Authorization Token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjp7ImlkIjoiNTlhZDFmZTI0MDVkNzk0YTFkYWQ2YmFkIiwiZGlzcGxheV9uYW1lIjoiQWRtaW4iLCJyb2xlX3R5cGUiOiJhZG1pbiJ9LCJpZCI6IlwiNTliYmJjODc0MDVkNzk0NjYwNGEzZjUyXCIiLCJlbWFpbCI6Imp5b3RpZ2F1dGFtMTA4QGdtYWlsLmNvbSJ9.oGA-goFi7ee6DdKn0Z4sctomaY6Ki0mfuJfxT4OK9WA

3.The server will validate the token and allow the browser to access API protected with authentication class based on validation results.

|  |
| --- |
| from rest\_framework import status, exceptions |
|  |

|  |
| --- |
| from django.http import HttpResponse |
|  |

|  |
| --- |
| from rest\_framework.authentication import get\_authorization\_header, BaseAuthentication |
|  |

|  |
| --- |
| from users.models import User |
|  |

|  |
| --- |
| import jwt |
|  |

|  |
| --- |
| import json |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class TokenAuthentication(BaseAuthentication): |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| model = None |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def get\_model(self): |
|  |

|  |
| --- |
| return User |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def authenticate(self, request): |
|  |

|  |
| --- |
| auth = get\_authorization\_header(request).split() |
|  |

|  |
| --- |
| if not auth or auth[0].lower() != b'token': |
|  |

|  |
| --- |
| return None |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if len(auth) == 1: |
|  |

|  |
| --- |
| msg = 'Invalid token header. No credentials provided.' |
|  |

|  |
| --- |
| raise exceptions.AuthenticationFailed(msg) |
|  |

|  |
| --- |
| elif len(auth) > 2: |
|  |

|  |
| --- |
| msg = 'Invalid token header' |
|  |

|  |
| --- |
| raise exceptions.AuthenticationFailed(msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| try: |
|  |

|  |
| --- |
| token = auth[1] |
|  |

|  |
| --- |
| if token=="null": |
|  |

|  |
| --- |
| msg = 'Null token not allowed' |
|  |

|  |
| --- |
| raise exceptions.AuthenticationFailed(msg) |
|  |

|  |
| --- |
| except UnicodeError: |
|  |

|  |
| --- |
| msg = 'Invalid token header. Token string should not contain invalid characters.' |
|  |

|  |
| --- |
| raise exceptions.AuthenticationFailed(msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return self.authenticate\_credentials(token) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def authenticate\_credentials(self, token): |
|  |

|  |
| --- |
| model = self.get\_model() |
|  |

|  |
| --- |
| payload = jwt.decode(token, "SECRET\_KEY") |
|  |

|  |
| --- |
| email = payload['email'] |
|  |

|  |
| --- |
| userid = payload['id'] |
|  |

|  |
| --- |
| msg = {'Error': "Token mismatch",'status' :"401"} |
|  |

|  |
| --- |
| try: |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| user = User.objects.get( |
|  |

|  |
| --- |
| email=email, |
|  |

|  |
| --- |
| id=userid, |
|  |

|  |
| --- |
| is\_active=True |
|  |

|  |
| --- |
| ) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if not user.token['token'] == token: |
|  |

|  |
| --- |
| raise exceptions.AuthenticationFailed(msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| except jwt.ExpiredSignature or jwt.DecodeError or jwt.InvalidTokenError: |
|  |

|  |
| --- |
| return HttpResponse({'Error': "Token is invalid"}, status="403") |
|  |

|  |
| --- |
| except User.DoesNotExist: |
|  |

|  |
| --- |
| return HttpResponse({'Error': "Internal server error"}, status="500") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return (user, token) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def authenticate\_header(self, request): |
|  |

return 'Token'