Python 101

REST and Working with REST in Django, Authentication with JWT, WebSocket Protocol and Building A QR Authentication Backend with Django

Contents

1. Web Services
2. REST
3. Django REST Framework
4. JWT Authentication
5. Problems due to statelessness of HTTP and Websocket Protocol as its solution
6. Using WS Protocol in Django using Django Channels
7. Building a QR Authentication Backend

Web Services

There are various definitions of web services but the one that we are most concerned with is -

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. The applications can be of various types eg. written in Java, Python etc. each app can talk irrespective of what language its written in through these services. These services can work on whatever protocol we require but mostly work on HTTP.

REST

REST Stands for REpresentational State Transfer. It is an approach for developing Web Services. It maps the operations in a web service to HTTP Methods eg. GET, POST, DELETE, PUT etc. preferably using JSON. To take an example of Databasing Operations CRUD we can map it as follows -

CREATE - PUT

READ - GET

UPDATE - POST

DELETE - DELETE

Django REST Framework

We can implement REST APIs in Raw Django.

See Examples/django_raw_rest/

But its better that we do it using some framework which can streamline our workflow in a more sophisticated sequence. We use Django REST Framework for this purpose.

See Examples/django_rest_framework_example/

JWT Authentication

JWT is abbreviation for JSON Web Token, JWT is a safe way to maintain a stateless session between two nodes.

See the following picture to understand purpose of JWT.

https://cdn-images-1.medium.com/max/800/1*SSXUQJ1dWjiUrDoKaaiGLA.png Consider JWT a simple string that is base64 encoded for exchanging data and is compose of **Header+Payload+Signature.**

In Python we can use PyJWT to create JWT tokens. We also have a django app named django-rest-framework-jwt that can peform this task for us. See Examples/jwt_example.py

JWT Authentication contd..

JWT Token can be saved on the client end to authenticate themselves later by sending that token again.

In a typical scenario they are saved in the browser’s localstorage much like the session tokens etc.

Problems with Statelessness of HTTP and WebSocket as its solution

HTTP closes its connection with the clients by default after each request is processed. So it creates a huge overhead if many requests are to be issued one after one to a certain client.

Over this there is also a problem of being unable to send client multiple responses whenever server wants.

And if the user is not logged in we also can’t keep track (IP is not reliable) about the origin of requests.

All of these problems can be resolved if the socket connection between server

Problems with Statelessness of HTTP and WebSocket as its solution contd..

All of these problems can be resolved if the socket connection between server is not closed implicitly and we give programmatic access to the closing of this Data Stream between client and server sockets.

In order to achieve this there exists another protocol over HTTP that is known as WebSocket protocol.

Because its just **Upgraded** version of HTTP protocol so a server can listen for both WS:// and HTTP:// at the same port.

By default Django doesn't support WS:// but using a **django-channels** we can use it in django as well.

Building A QR Authentication Backend with Django