# JWT Token

1. Stateful session
2. Stateless session

### Problem with Stateful HTTP sessions

Stateful sessions means server will remember the state because once authentication is success then sessions/session object is created and session object will be stored in the server

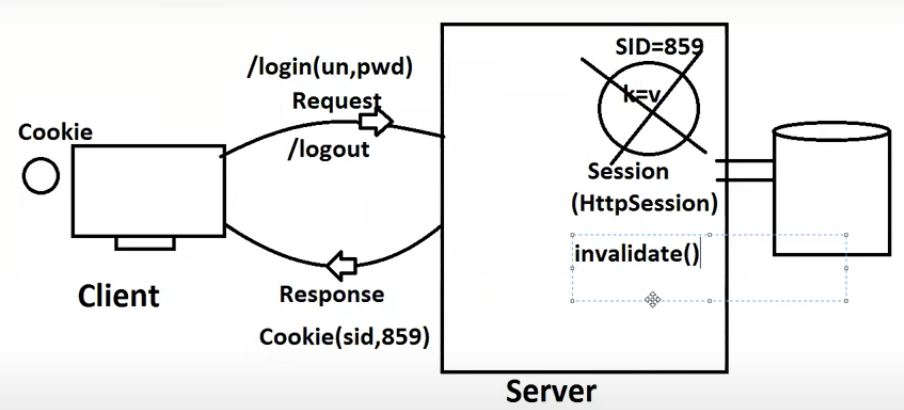
And user should pass the session id in each request

Session object is nothing but object of class implementing HTTPSession it’s a MAP

when user logs out we should invalidate the session and that object gets removed from the server ,

if 1Million people logged in then 1Million session objects will be created and stored on the server

Issue : hacker can come and tap the session id



### Stateless sessions

Means client data will not be stored in the server and server will not maintain the user details and unlike above there won’t be any session object created as server wont store any thing

Here every request is a new request

Then how authentication takes place??

Simple for every request we should pass some temporary token String in the header, this token is valid up to some 10-20 minutes

Use case:-

1. in web services mostly we will use stateless authentication as server won’t remember anything and doesn’t track anything
2. especially in micro services horizontal scaling apps we should use this token means stateless sessions

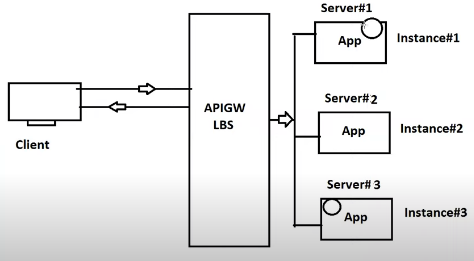
Stateless sessions are good for micro services

In horizontal scaling Stateless sessions are best fit

Stateless means server won’t maintain any state of user like server will not maintain any session objects every request is fresh request, for every request u should pass token

If you consider an app which is scaled 3 times in prod, if u maintain Stateful session, then problem is

1) 1st request may go to server1 and second time due to load on same server-1 , same user might be redirected to server -2 on next request, Session objects needs to be created in 3 servers and when user logs out we have to invalidate the session in 3 instances which is an overhead, but still if u want to avoid that we have some sticky sessions like if user request if it went to server-1 we can always ensure all his requests should go to server-1 but why to take risk so in micro services better go with stateless



Limitations of stateless sessions

Token must be validated on each and every request we should check whether token is valid or not and keep that logic in Rest template interceptor

Benefits

Server doesn’t need to maintain the user session objects and no need to invalidate the session in all replicas when user invalidates the session

## JWT

It is a token which have your encoded (not encrypted ) information it is not much secure as this data is not secure data, hence this is encoded instead of encryption

But remember if the token is tampered we can easily track as if data is changed then signature will also be changed

JWT=

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

)

See here we are encoding (encoding doesn’t secure your info it only transforms the data)

Anyone can easily decode the token and see ex:- we can decode at jwt.io,

JWT json web token is an open source API used to generate the token based on client details and secret key

Reference jwt.io

Dependency information is

implementation **'io.jsonwebtoken**:jjwt-api:0.11.2'

if u are using >java 8 add jaxb dependency manually because it was removed from java 11 onwards I think

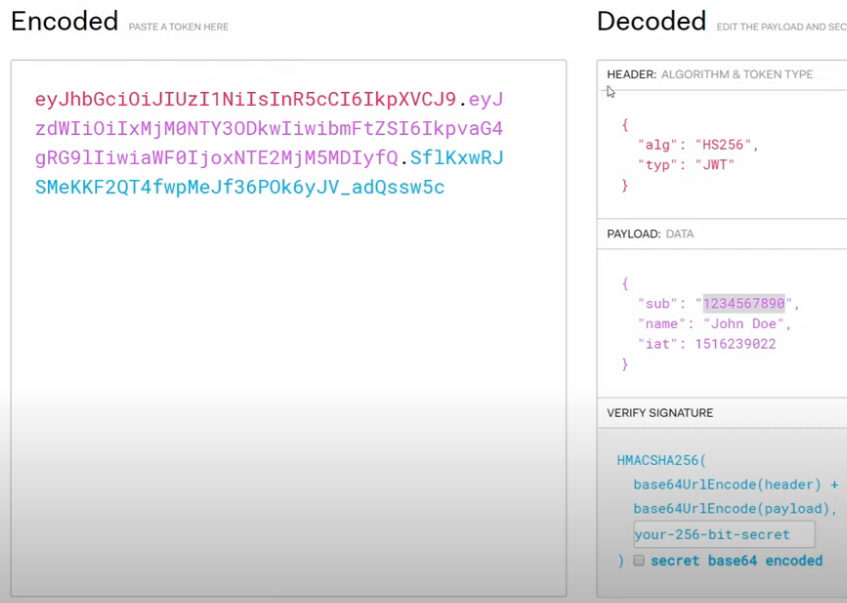
// https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api

implementation 'javax.xml.bind:jaxb-api:2.3.0'

**What is a JWT????????**

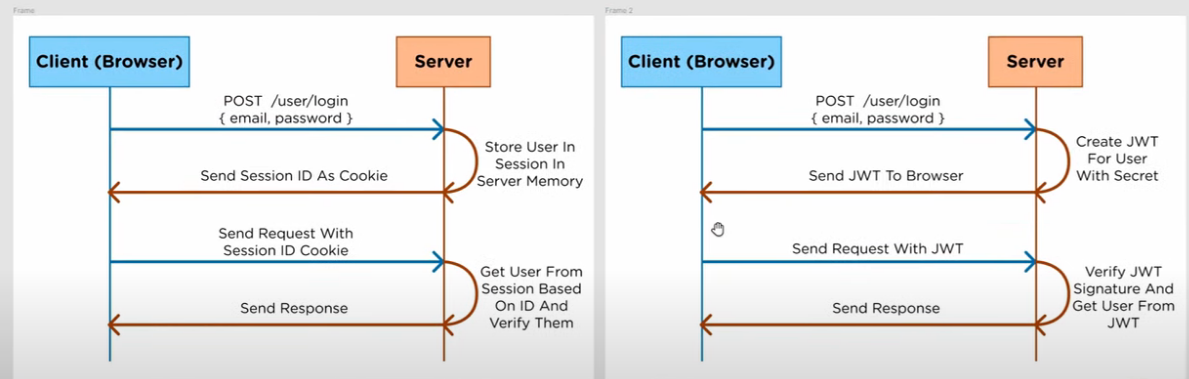
If no JWT as part of being Stateful server should maintain users data in form of session objects, so if 10,000 users logged in 10k session objects will be created in server memory, in order to avoid that instead of server maintaining that data, user when he comes to server he has to bring his info in the form of JWT token with him whenever time he come,

He will carry his info in the form of JWT token



The above diagram clearly shows encoded means it’s just data transformation only, it’s not encrypted with any symmetric or asymmetric key, anybody can encode and decode without using any key

## JWT vs. session ids /cookie



|  |  |
| --- | --- |
| Session ID | JWT |
| This is like some unique id which is created by server and that is stored in browser  1,000 users = 1,000 session objects in server and each user should pass this id  1 Million users= 1Million session objects created in server to store users info | Here 1 Million users= 0 objects stored in server, user info will not be stored in server, this is a stateless model, user details are in the token itself  JWT= Because users details are encrypted or encoded or signed with secret key  So JWT contains the user info only (hence these details will not be stored in server session objects again) |
| User details will be stored in server and retrieved with help of session id | Nothing is stored in server, JWT token itself has all the user info stored in it |
|  | When JWT token is presented to server, server will verify the signature of token and extract the required info |
|  |  |
|  |  |
|  |  |

Signatures---> HS256 vs. RS256

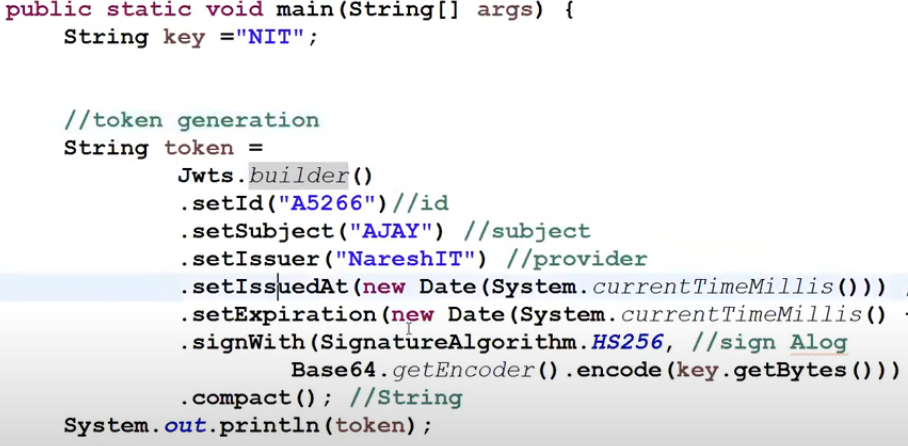
HS256 and RS256 are algorithms to sign JWT, here we are not hashing the content if I hash we can’t de-hash that data

HS256- Hash based message authentication code HMAC

Signatures are not encryptions, Signatures doesn’t make data un-readable,

Why to sign/ use of signing the token? It says content of the JWT hasn’t changed or tampered

Code to generate Token



There are many signature algorithms to generate the token like HS256 or HM245…

Base64 is the class given by java 8

We should set token expiration time , subject is the main thing

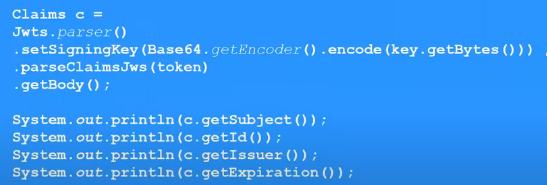
And output will be like

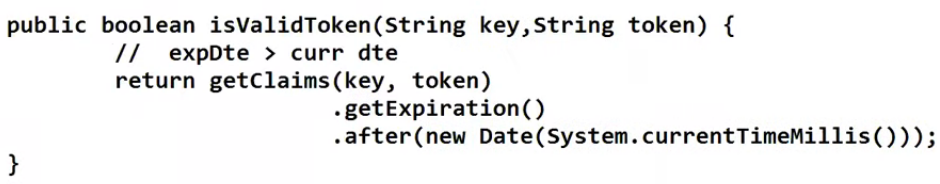
XXXXXXXX.YYYYYYYYYYYYYYYYYYYYYYYYY.ZZZZZZZZZZZZZZ

Header.payload.signature

Extracting token

We can decode JWT at jwt.io





Practice:

1. modify the token and see
2. use an expired token and see what happens