Spring Boot -Spring Security -JWT

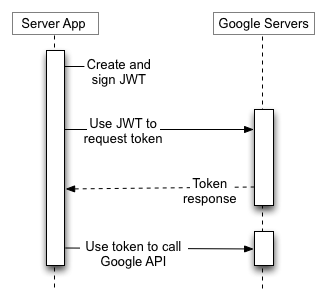
OAuth 2.0 defines a protocol, i.e. specifies how tokens are transferred, JWT defines a token format.

OAuth 2.0 and "JWT authentication" have similar appearance when it comes to the (2nd) stage where the Client presents the token to the Resource Server: the token is passed in a header.

But "JWT authentication" is not a standard and does not specify how the Client obtains the token in the first place (the 1st stage). That is where the perceived complexity of OAuth comes from: it also defines various ways in which the Client can obtain an access token from something that is called an Authorization Server.

So the real difference is that JWT is just a token format, OAuth 2.0 is a protocol (that may use a JWT as a token format).

Firstly, we have to differentiate JWT and OAuth. Basically, JWT is a token format. OAuth is an authorization protocol that can use JWT as a token. OAuth uses server-side and client-side storage. If you want to do real logout you must go with OAuth2. Authentication with JWT token can not logout actually. Because you don't have an Authentication Server that keeps track of tokens. If you want to provide an API to 3rd party clients, you must use OAuth2 also. OAuth2 is very flexible. JWT implementation is very easy and does not take long to implement. If your application needs this sort of flexibility, you should go with OAuth2. But if you don't need this use-case scenario, implementing OAuth2 is a waste of time.

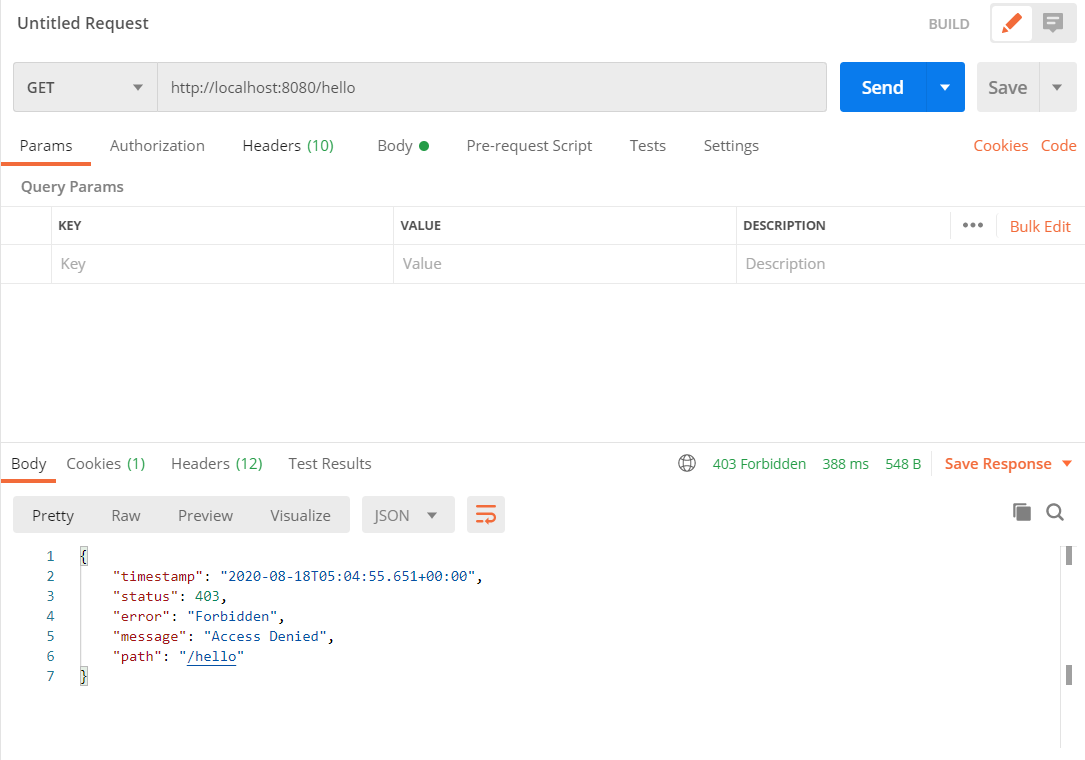


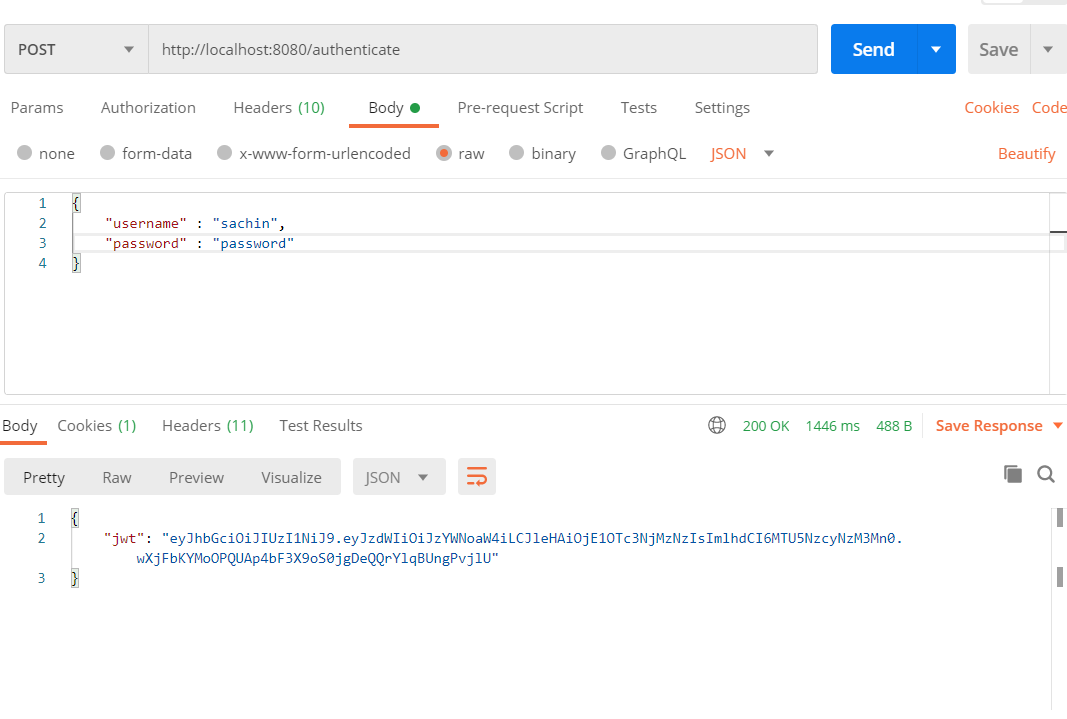
Step 1 :

A /authenticate API endpoints

-- Accpect USER ID and PASSWORD

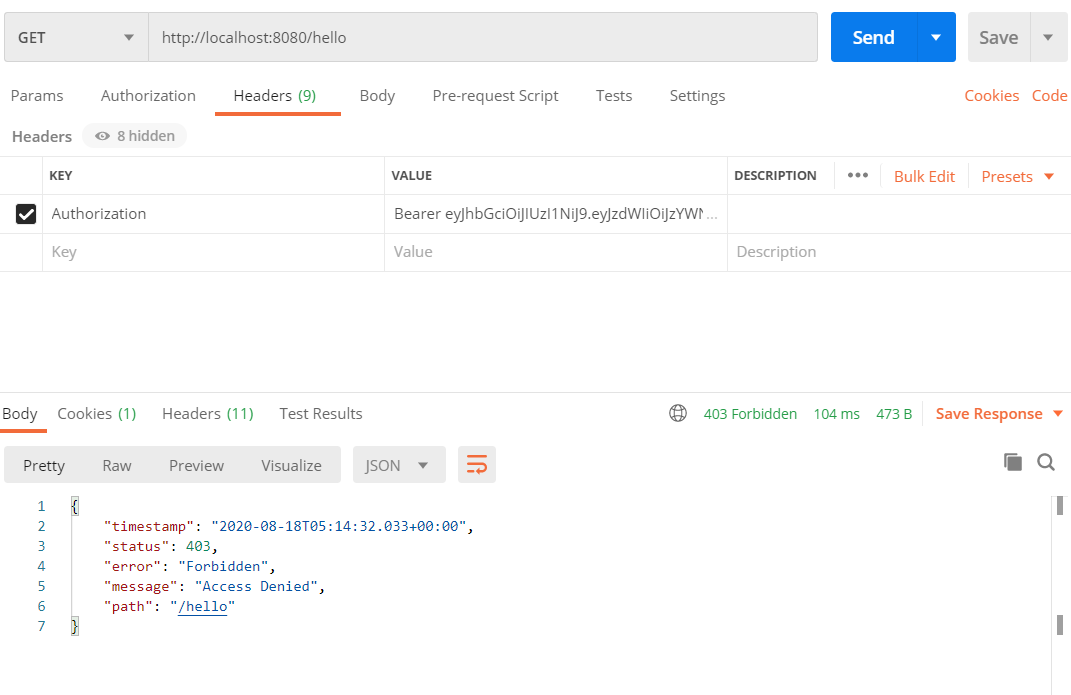
-- Return JWT as RESPONSE





JWT support is still not been implemeneted

Only /authenticate endpoints is configured whcih retruns JWT value till now so even with JWT it not going to work



Step 2 :

Intercept All incoming Requests

- Extract JWT from the Header

-Validate and set execution context

