247)Module Introduction

Here we will see authentication in angular app. Large part of authentication happens in backend. here we will not stress on backend. We will see front end part only here. We are already using firebase and it is easy to enable authentication in firbase.

So we want to allow only logged in user to access recipes on server and create new recipes on front end.

248)How authentication works in single page application

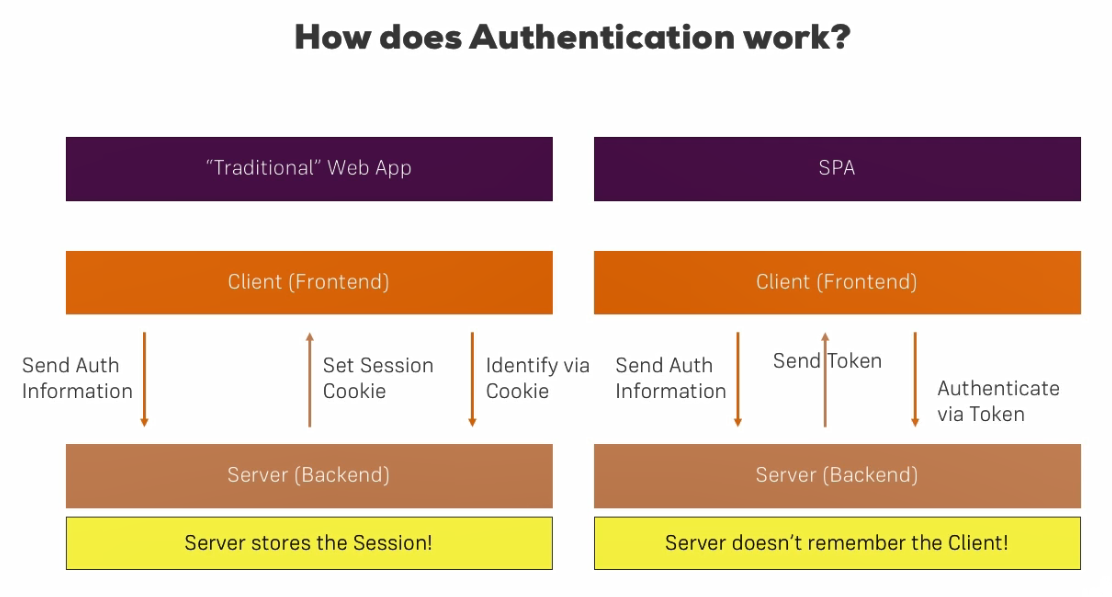
Here we will see difference between traditional applications and SPA. In traditional app we have also have front end and backend but we have strong connection between front end and backend because,this is super important- views that user sees(HTML code) is dynamically generated by server. So if you are using server side framework chances are that you are using some kind of templating engine to render views. In the end the client, and that’s important will always receive the finished HTML code.

In SPA it is different, we receive a single page and the client, angular is responsible for changing HTML code dynamically.

Back to traditional app, here if we want to authenticate then client sends some auth information(username & password), server will validate it in database. if it is valid then server will create a session and store that session on server. As we have close connection between server and client, with server rendering view anyways, the server is able to remember the client because it is communicating with client all the time. Therefore this session is then stored on the server, for example in database and the client will get back a session cookie which mainly contains the id of the session which can then be sent with every request where you want to access some sensible information or resource which might be protected so that server can now check if that cookie is valid i.e session id is correct. if it has as session for this id stored on server. Again because that’s really important , this is working because we have close connection between server and front end.

For SPA this changes, here we do not have that strong connection b/w frontend and backend. we might reach out to backend from time to time but that is not guaranteed, a lot of request that user makes are handled by client entirely.HTMl code can be changed by angular. So that’s why we dnt have strong connection and beside that we might have very generic backend exposing a restful api to which we can connect from several web apps and mobile apps. So in case server will not remember who the client is. So to authenticate we still send auth information to server, and server will check it in database. But if it is valid it won’t create a session because it does’nt regularly communicate with the client , it does’nt really know the client. So server does not know the client and hence it cannot store a session. Still we want to give client something which allows the client to kind of inform the server , on future request that user is logged in. so therefore solution is that server will send back a token, json web token (JWT) typically which endcodes some information about authenticated user, ofcourse no sensible information and is hashed with certain algorithm and certain secret only known to server. The client now has this token, now if we want to access some resource on server that is protected, we would simply attach that token to the request and since the token was generated on server and server knows the secret and algorithm, the server is able to validate this token. So we authenticate via token on future request, because we attached this token and the server is able to check if token is valid. That is how we do authentication on single page application. So w ehave to use this pparoach.

Firebase makes it very easy. But you implement this with another backend, this is what you need to do in frontend- store the token and attach the token to all future request.



249)More about JWT

The following page should be helpful: <https://jwt.io/> - specifically, the introduction: <https://jwt.io/introduction/>

250)Creating a signup page and route

Here we create a signup component. Then we add a link in navigation bar. When we click on this link then this signup page is opened.

251)Setting Up Firebase SDK

Now we have to enable authentication at backend. so go to your firebase project.then to authentication. Then setup a signin method. Select email/password. Then save it. Now you should be able to send create new users and sign users in request to the backend.

To send authentication request from front end we will use Firbase SDK. The reason for this is there is no great rest API from firebase side for this.

If you are using your backened then you will use http service to send authentication related data to your backend. it will then send you a token. Then you will store that token ,for example in local storage, manually. To get firebase run this-

**npm install –save firebase**

then we create a new service auth in auth folder. This service will have methods that we will use to create new users and sign new users in. as we are going to use firebase sdk, so we need to configure it. We should configure it at the point of time when our app starts. So we configure it in ngOnInit hook of app.component. then we import firebase package.

App.component.ts-

ngOnInit() {

firebase.initializeApp({

apiKey: "AIzaSyARhoPV\_qKYVzlUEsdtcCU7Zky3hvoaweo",

authDomain: "udemy-ng-http2.firebaseapp.com"

});

}

Object that we passed to this function can be obtained from, firebase. Go to authentication then websetup. There you will find this object.

252)signing users up

Here in auth service we use firebase to create a new user on bacakend. Then we call this method on service when we hit sibmit button in Signup Component. We can see new users created in firebase by going to authentication ->users.

So now we are creating users on our backend.

253)Signin Users in

Here we call signInUser method in auth service to signin user. We call this method when we submit form in siginComponent. Firebase sends us a response and this response holds our token, it also holds other things. But importantly it holds the token by which we are able to identify ourselves.

Now we should store this token but firebase does that automatically for us. It stores our token in local storage of browser.

254)Requiring a token(on backend)

Now I want to use the token that I get back from backend. I want to make sure that saving and fetching data is possible only if we send token to backend. so on backend I want to check whether user is authenticated or not. If user is not authenticated then I want to deny access.

Right now we allow read and write access to anyone. We need to change it. Go to firebase, database,rules. Here we set both read and write to true, which means allow access to all. We change access rights.

**{**

**"rules": {**

**".read": "auth != null",**

**".write": "auth != null"**

**}**

**}**

With this we are making sure that only authenticate users are able to access our content. If now we try to access to routes we get authentication error.

255)Sending the Token

Here we want to send the token which firebase send us backs and save it in our local storage. Then we need to send this token with pur http request to get pass firebase authentication.

We add a new method getToken in authService to get the token.

getToken() {

return firebase.auth().currentUser.getToken()

}

This is asynchronous operation, because it gets the token in localStorage and checks this token with firebase that whether this token has timeout or not. So this returns a promise. When promise resolves then we get back the token.

Now we want to send this token we send http request to our backend, so we go to data-storage.service, here in getReceipe we send the get request. But before sending it we want to send token. So we can call getToken method of authService and get token. But problem is since fetching token is asynchronous operation, at a time we are sending request we may not have token. So how should we get token?

So we do this in auth.service. we create a property called token, we set this property with token returned from backend when we sign in. in our getToken method we return this token property , also we set up a promise to get the new token if our token has expired.

Then we send our token with query parameters in our url.