Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

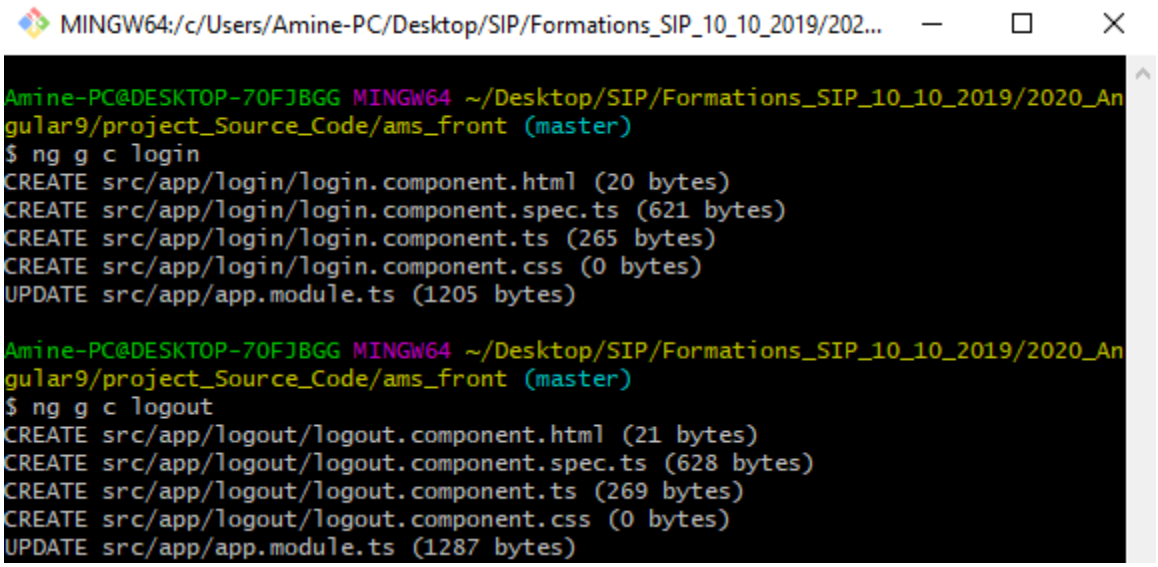
Authentication and Authorization

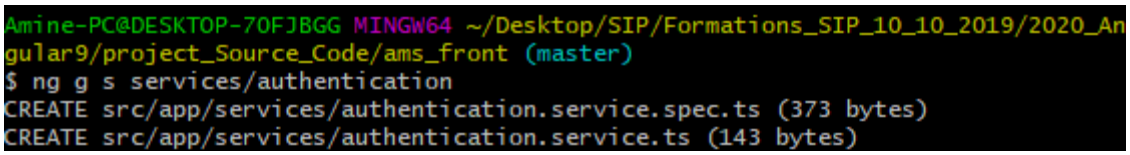
• Part 1 : Login Authentication (hard coded)

We will start this section by creating login and logout pages and we will be using hard coded user name. Also will be implementing session management so that only a user who is logged in can view the pages. Else he will be redirected to the login page. In the next section we will be implementing authentication based on the backend.

Let's start by creating 2 components: login and logout and a service called authentication.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**





Create a new authentication service where we check if the user name and password is correct then set it in session storage object. **Using sessionStorage properties we can save key/value pairs in a web browser. The session Storage object** stores data for only one session. So the data gets deleted if the browser is closed. We will be having the following methods

- **authenticate()** Authenticate the username and password
- **isUserLoggedIn()** -checks the session storage if user name exists. If it does then return true
- **logout()**- This method clears the session storage of user name

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

Our Authentication service

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class AuthenticationService {

  constructor() { }

  authenticate(username, password) {
    if (username === "amine" && password === "1234") {
      sessionStorage.setItem('username', username)
      return true;
    } else {
      return false;
    }
  }

  isUserLoggedIn() {
    let user = sessionStorage.getItem('username')
    console.log(!(user === null))
    return !(user === null)
  }

  logOut() {
    sessionStorage.removeItem('username')
  }

}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

Using the Login Component we will be taking the username and password from the user and passing it to the authentication service to check if the credentials are valid. It will have the following method- **checkLogin()**- This method checks if the user credentials are correct by calling the previously created AuthenticationService.

- **The login.ts**

```
import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute} from '@angular/router';
import { AuthenticationService } from '../services/authentication.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  username: string;
  password: string;
  invalidLogin = false;

  successMessage = "Authentication success";
  errorMessage = "Invalide username or password";
  constructor(private router: Router,
    private loginservice: AuthenticationService) { }

  ngOnInit() {
  }

  checkLogin() {
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
    if (this.loginservice.authenticate(this.username, this.password)) {
      this.router.navigate([''])
    } else
      this.invalidLogin = true
  }

}
```

- The **login.html**

```
<div class="container col-lg-6">
    <h1 class="text-center">Authentification</h1>
    <div class="card">
        <div class="card-body">
            <form class="form-group">
                <div class="alert alert-
warning" *ngIf='invalidLogin'>{{errorMessage}}</div>
                <div class="form-group">
                    <label for="email">User Name :</label>
                    <input type="text" class="form-
control" id="username" [(ngModel)]="username"
                        placeholder="Enter User Name" name="username">
                </div>
                <div class="form-group">
                    <label for="pwd">Password:</label>
                    <input type="password" class="form-
control" [(ngModel)]="password" id="password"
                        placeholder="Enter password" name="password">
                </div>
                <button (click)=checkLogin() class="btn btn-
success">Login</button>
```

```
            </form>
        </div>
    </div>
</div>
```

Add the login path to the routing module.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AddProviderComponent } from './add-provider/add-provider.component';
import { ListProviderComponent } from './list-provider/list-provider.component';
import { UpdateProviderComponent } from './update-provider/update-
provider.component';
import { LoginComponent } from './login/login.component';

const routes: Routes = [
  { path: "", pathMatch: "full", redirectTo: "app-navbar" },
  { path: "listProvider", component: ListProviderComponent },
  { path: "addProvider", component: AddProviderComponent },
  { path: "updateProvider/:id", component: UpdateProviderComponent },
  { path: 'login', component: LoginComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

In the logout component we clear the session storage username by calling the authentication service.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
import { Component, OnInit } from '@angular/core';
import { AuthenticationService } from '../services/authentication.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-logout',
  templateUrl: './logout.component.html',
  styleUrls: ['./logout.component.css']
})
export class LogoutComponent implements OnInit {

  constructor(
    private authenticationService: AuthenticationService,
    private router: Router) {

  }

  ngOnInit() {
    this.authenticationService.logOut();
    this.router.navigate(['login']);
  }

}
```

Add the logout path to the routing module-

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AddProviderComponent } from './add-provider/add-provider.component';
import { ListProviderComponent } from './list-provider/list-provider.component';
import { UpdateProviderComponent } from './update-provider/update-
provider.component';
import { LoginComponent } from './login/login.component';
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
import { LogoutComponent } from './logout/logout.component';

const routes: Routes = [
  { path: "", pathMatch: "full", redirectTo: "app-navbar" },
  { path: "listProvider", component: ListProviderComponent },
  { path: "addProvider", component: AddProviderComponent },
  { path: "updateProvider/:id", component: UpdateProviderComponent },
  { path: 'login', component: LoginComponent },
  { path: 'logout', component: LogoutComponent },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

➔Modify existing navbar component to add login , logout menu options

In the component we check if the user is logged in or not. This will be used to decide if all the menu links should be visible to the user or not.

Content of navbar component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthenticationService } from '../services/authentication.service';


@Component({
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {

  constructor(private loginService: AuthenticationService) { }

  ngOnInit() {
  }

}
```

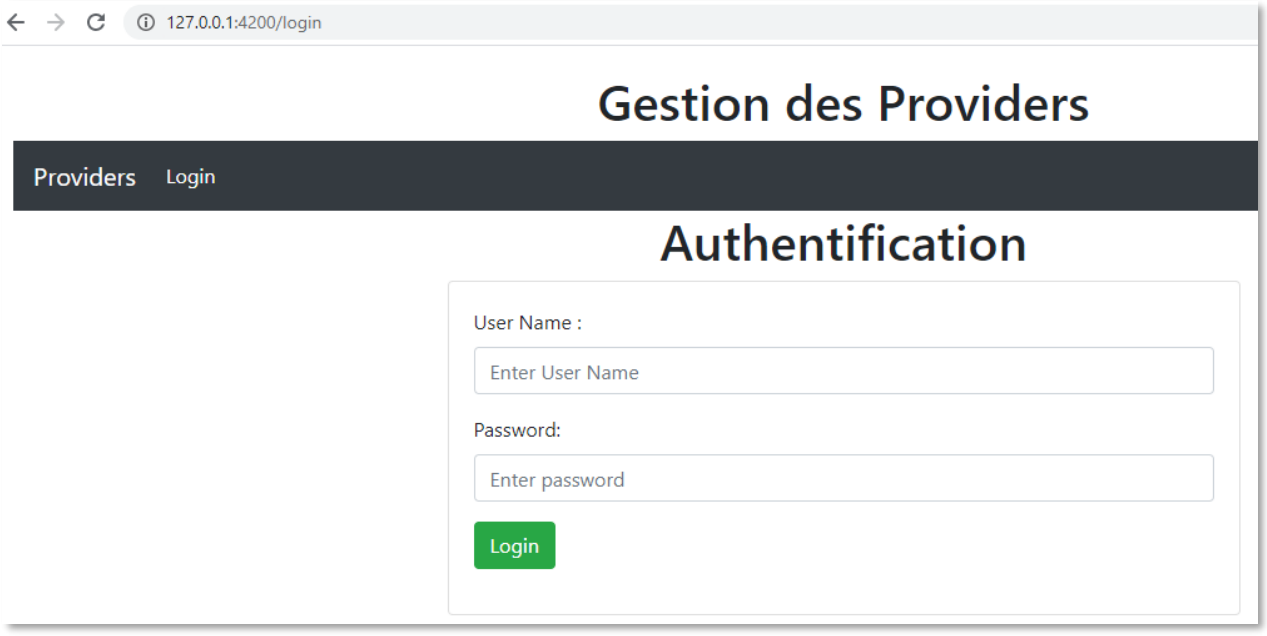
Content of navbar component.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="#">Providers</a>

    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-
link" routerLink="/listProvider" *ngIf="loginService.isUserLoggedIn()" routerLink
Active="active">Liste</a>
            <a class="nav-item nav-
link" routerLink="/addProvider" *ngIf="loginService.isUserLoggedIn()" routerLinkA
ctive="active">Ajouter</a>
            <a class="nav-item nav-
link" routerLink="/login"  *ngIf="!loginService.isUserLoggedIn()" routerLinkActiv
e="active">Login</a>
            <a class="nav-item nav-
link" routerLink="/logout" *ngIf="loginService.isUserLoggedIn()"  routerLinkActiv
e="active">LogOut</a>
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

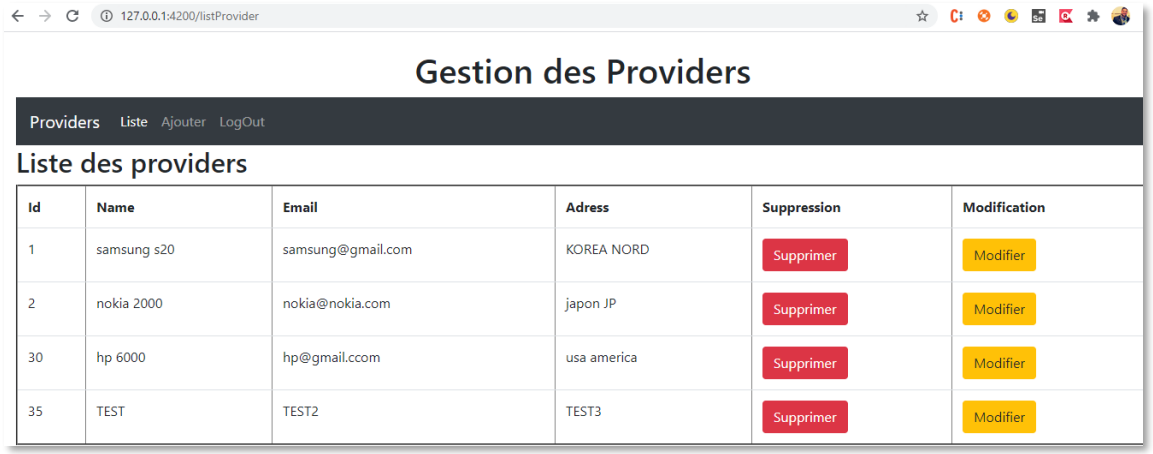
```
        </div>
    </div>
</nav>
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->



<router-outlet></router-outlet>
```

Here the content of the login page

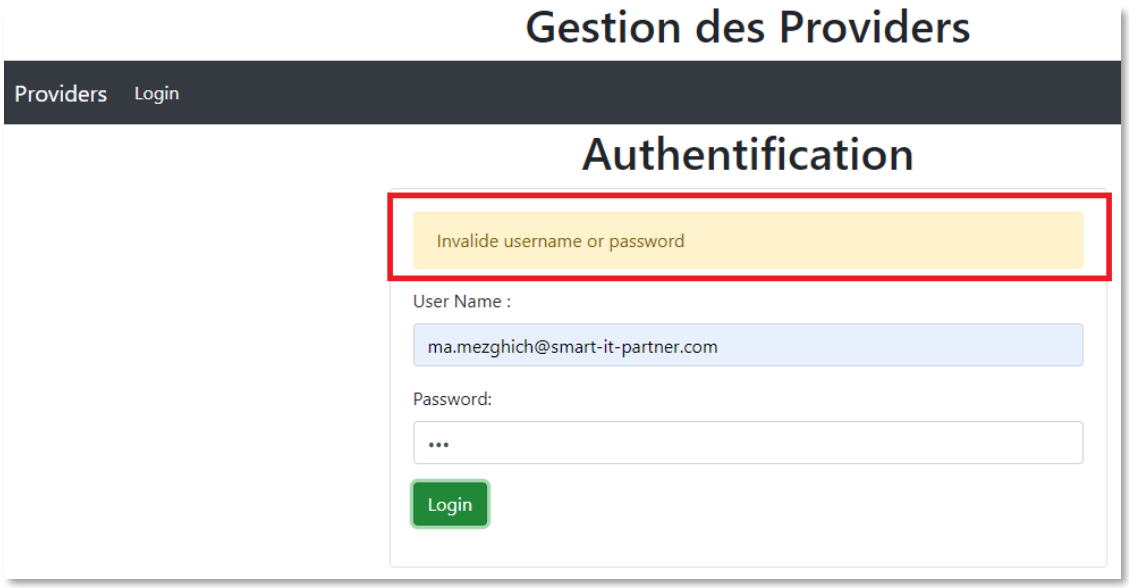


If you enter correct login and password, you will be allowed to visit data (provider list and the form to add new provider)

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



- With incorrect login or password, you will obtain warning message



Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

But what will happen if the user directly tries to access a page without login. For example if a user directly navigates to localhost:4200 He will be able to view the page. But this should not be the case as the user is not logged in. So we should first check if the user is logged in and only then allow the user to view the page. We achieve this based on the **CanActivate** interface.

- ## Create the AuthGaurd Service

We will be creating a new Service named AuthGaurdService. This service will **activate a particular route only if the user is logged in.**

```
Amine-PC@DESKTOP-7OFJBGG MINGW64 ~/Desktop/SIP/Formations_SIP_10_10_2019/2020_An
gular9/project_Source_Code/ams_front (master)
$ ng generate service services/authGaurd
CREATE src/app/services/auth-gaurd.service.spec.ts (349 bytes)
CREATE src/app/services/auth-gaurd.service.ts (138 bytes)
```

Let the AuthGaurdService implement the **CanActivate** interface. By overriding the **canActivate** method we specify that a route should be active only if the user is logged in.

```
import { Injectable } from '@angular/core';
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from
'@angular/router';
import { AuthenticationService } from './authentication.service';


@Injectable({
  providedIn: 'root'
})
export class AuthGaurdService implements CanActivate {

  constructor(private router: Router,
    private authService: AuthenticationService) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (this.authService.isUserLoggedIn())
      return true;

    this.router.navigate(['login']);
    return false;

  }

}
```

Modify the app.routing.ts to activate route only if the user is logged in using the above AuthGaurdService.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AddProviderComponent } from './add-provider/add-provider.component';
import { ListProviderComponent } from './list-provider/list-provider.component';
import { UpdateProviderComponent } from './update-provider/update-
provider.component';
```

```
import { LoginComponent } from './login/login.component';
import { LogoutComponent } from './logout/logout.component';
import { AuthGaurdService } from './services/auth-gaurd.service';

const routes: Routes = [
  { path: "", pathMatch: "full", redirectTo: "app-navbar" },
  { path: "listProvider", component: ListProviderComponent, canActivate: [AuthGau
rdService] },
  { path: "addProvider", component: AddProviderComponent, canActivate: [AuthGaurd
Service] },
  { path: "updateProvider/:id", component: UpdateProviderComponent, canActivate:
[AuthGaurdService] },
  { path: 'login', component: LoginComponent },
  { path: 'logout', component: LogoutComponent, canActivate: [AuthGaurdService] }
,
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

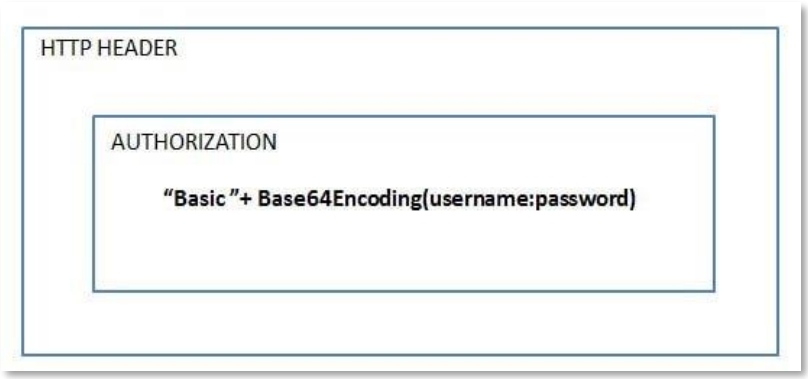
Now if the user tries to access a page without logging in, he will be directed to the login page.

That's it for the first part of the workshop! In the next section we will be based on spring boot to manage authentication.

• Part 2 : Login Authentication (based on backend)

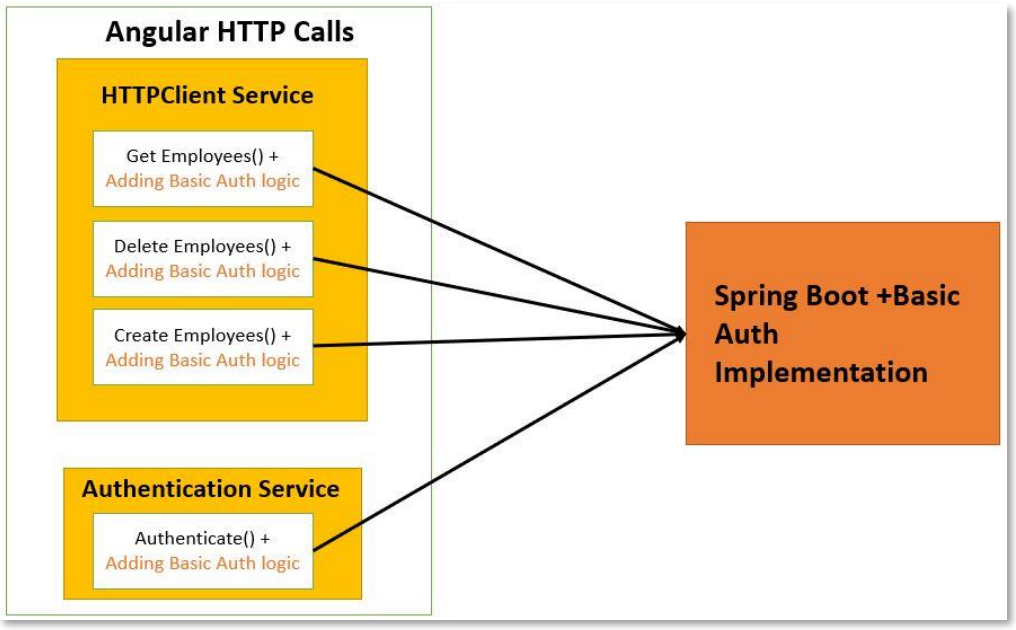
In this section we will be implementing Basic Authentication using Spring Boot. All the REST calls made from Angular to Spring Boot will be

authenticated using Basic Authentication. Basic authentication is a simple authentication schema built using the HTTP protocol. When using this protocol the HTTP requests have **Authorization header which has the word Basic followed by a space and base 64 encoded string username:password.**



In this part the angular code though functional is not optimized. There is lot of repetition of the Basic Authentication code for adding header. We will be optimizing this code using the HTTPInterceptors.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



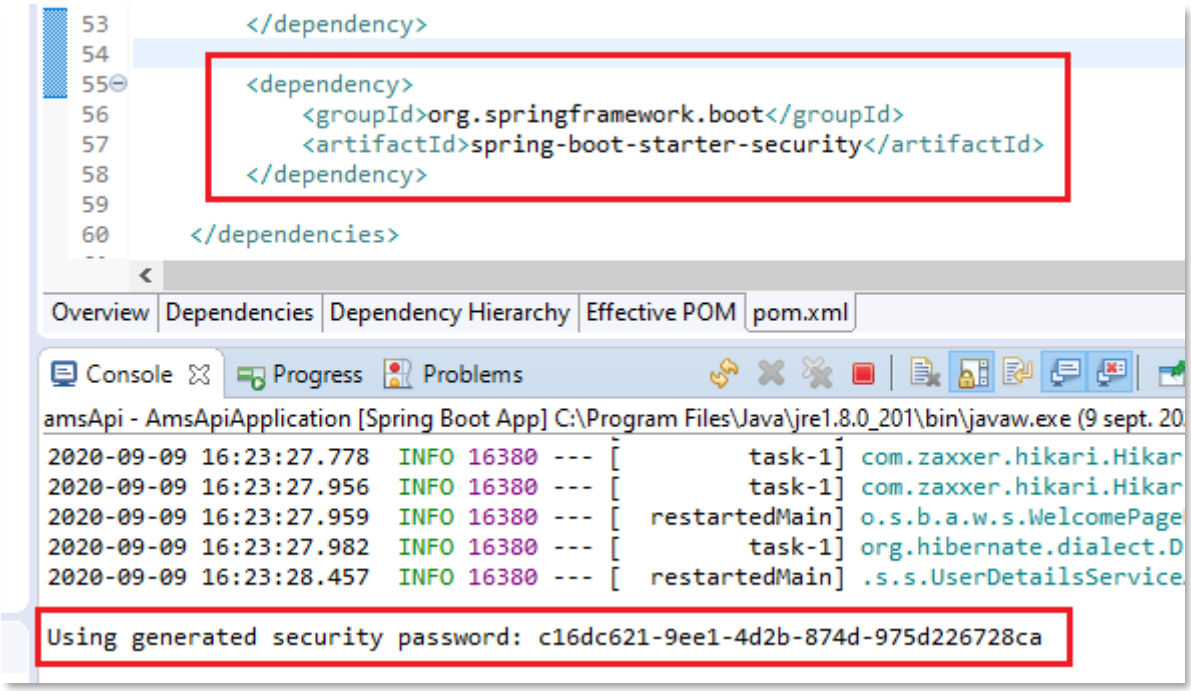
In the previous workshop we had implemented Spring Boot REST API's for performing CRUD operations for both Provider and Articles entities. In this workshop we will be adding the basic authentication to this application.

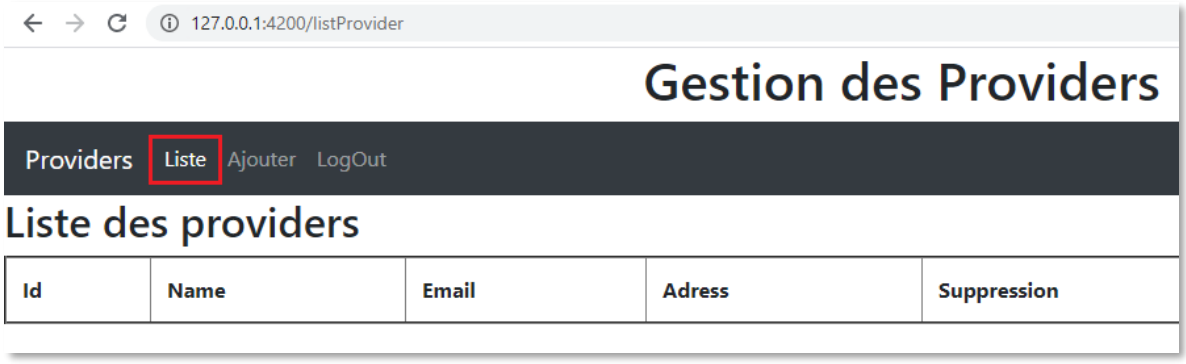
• Backend Side

Let's add this dependency in the pom.xml file and **then restart the project**

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

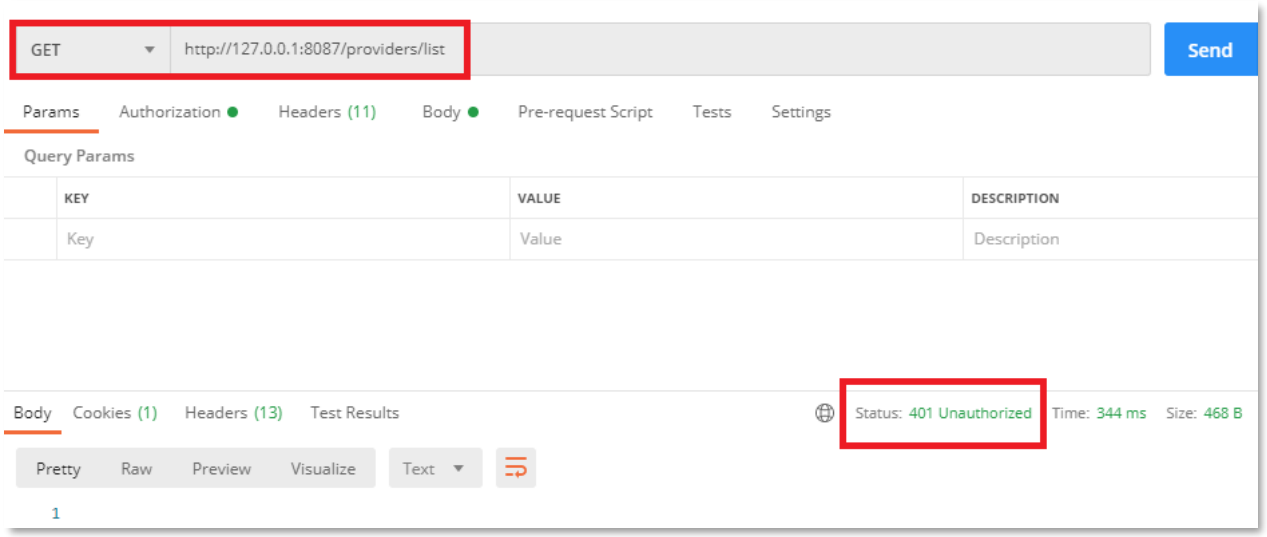
Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**





Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

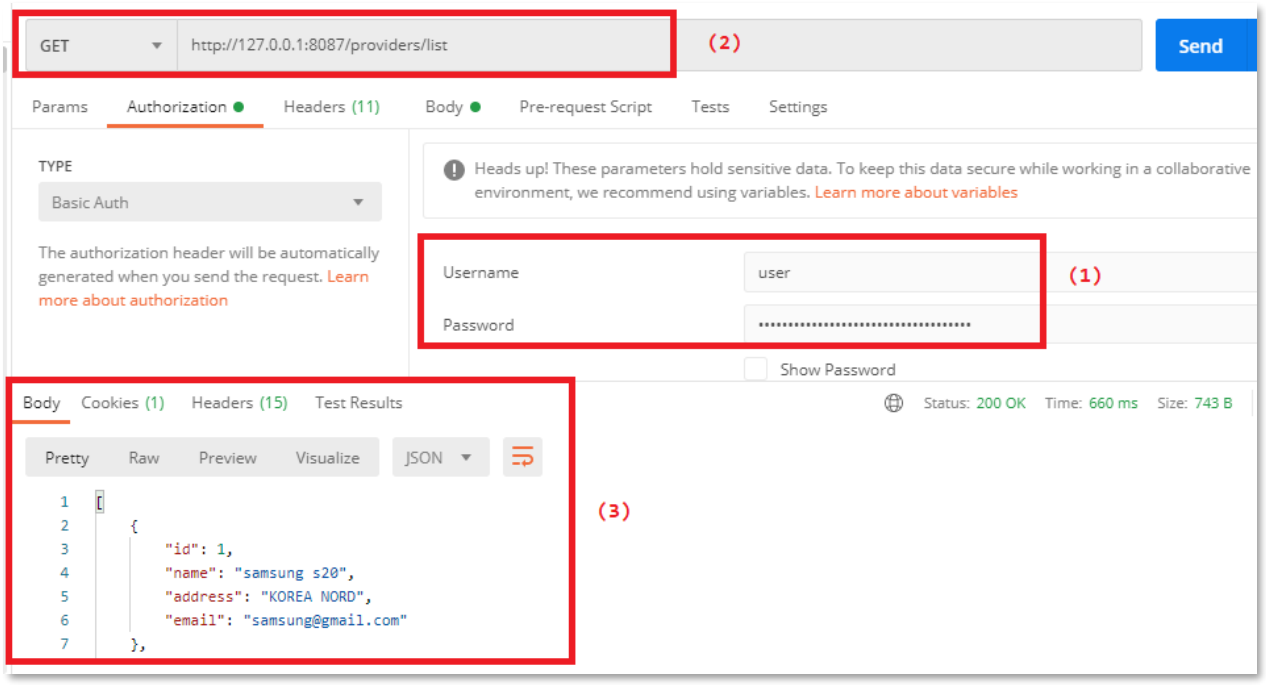
Once Spring Security is on the classpath, then Spring Boot automatically secures all HTTP endpoints with "basic" authentication.



When you start the spring boot project, the default password is randomly generated and printed in the console log:

```
Default username: user
Using generated security password: c16dc621-9ee1-4d2b-874d-975d226728ca
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

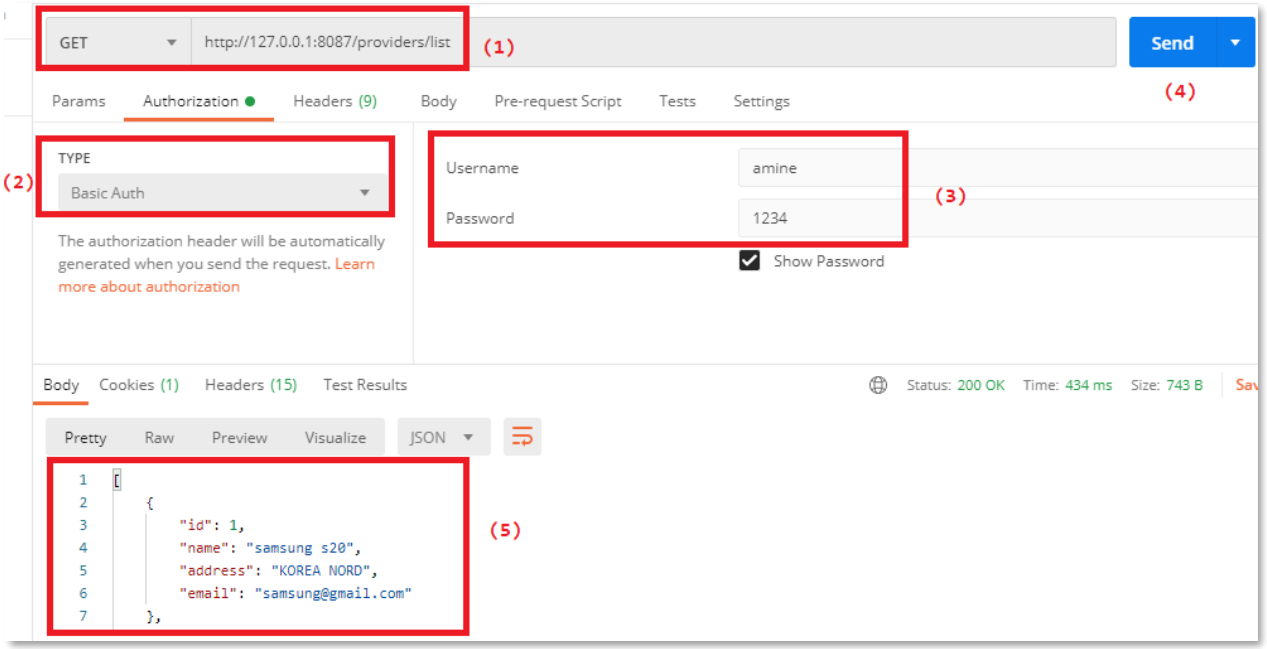


Configure standard user and password

We can override the default user and password using the below properties in the *application.properties* file:

```
spring.security.user.name=amine
spring.security.user.password=1234
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



Configure WebSecurityConfigurerAdapter

To enable authentication and authorization support in spring boot rest APIs, we can configure a utility class *WebSecurityConfigurerAdapter*. It helps in requiring the user to be authenticated prior to accessing any configured URL (or all URLs) within our application.

```
package com.sip.ams.configuration;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
```

```
import
org.springframework.security.config.annotation.authentication.builders.
AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurit
y;
import
org.springframework.security.config.annotation.web.configuration.Enable
WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSec
urityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
	@Override
	protected void configure(HttpSecurity http) throws Exception {

		http.csrf().
		  disable()
		      .authorizeRequests()
		      .antMatchers(HttpMethod.OPTIONS, "/**")
		      .permitAll()
		      .anyRequest()
		      .authenticated()
		      .and()
		      .httpBasic();
	}

}
```

- ✓ **Other method** (create in memory user using the configuration class):

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
package com.sip.ams.configuration;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.authentication.builders.Authentication
ManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigure
rAdapter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http.csrf().disable().

				authorizeRequests().antMatchers(HttpMethod.OPTIONS,
"/**").permitAll().anyRequest().authenticated()
				.and().httpBasic();
	}

	@Autowired
	public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {

	auth.inMemoryAuthentication().withUser("mohamed").password("{noop}1234").roles
("USER");
	}
}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

You can also simply prefix {noop} to your passwords in order for the DelegatingPasswordEncoder use the NoOpPasswordEncoder to validate these passwords

• Define AuthenticationBean and BasicAuthController

Let's create an *AuthenticationBean*, which is used to return a success message to the client:

```
package com.sip.ams.entities;

public class AuthenticationBean {
        private String message;

   public AuthenticationBean(String message) {
      this.message = message;
   }

   public String getMessage() {
      return message;
   }

   public void setMessage(String message) {
      this.message = message;
   }

   @Override
   public String toString() {
      return String.format("Bienvenue dans backend [message=%s]",
message);
```

```
    }
}
```

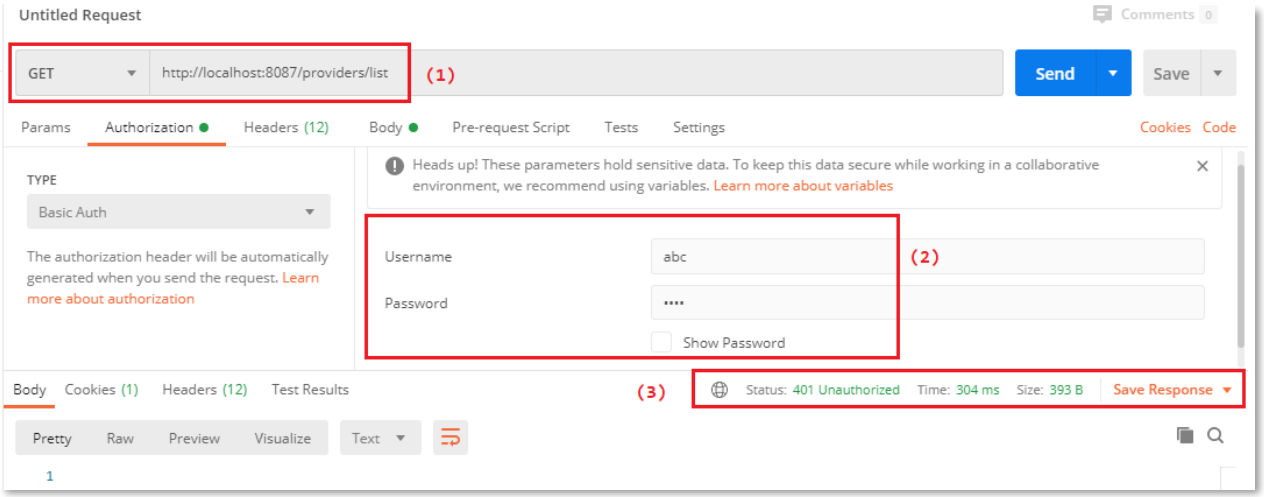
Let's create a *BasicAuthRestController* class with **/basicauth** REST API for returning the authentication success message.

```
package com.sip.ams.controllers;


import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.sip.ams.entities.AuthenticationBean;


@CrossOrigin(origins = "http://localhost:4200")
@RestController
public class BasicAuthRestController {

        @GetMapping(path = "/basicauth")
          public AuthenticationBean basicauth() {
             return new AuthenticationBean("You are authenticated");
          }

}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

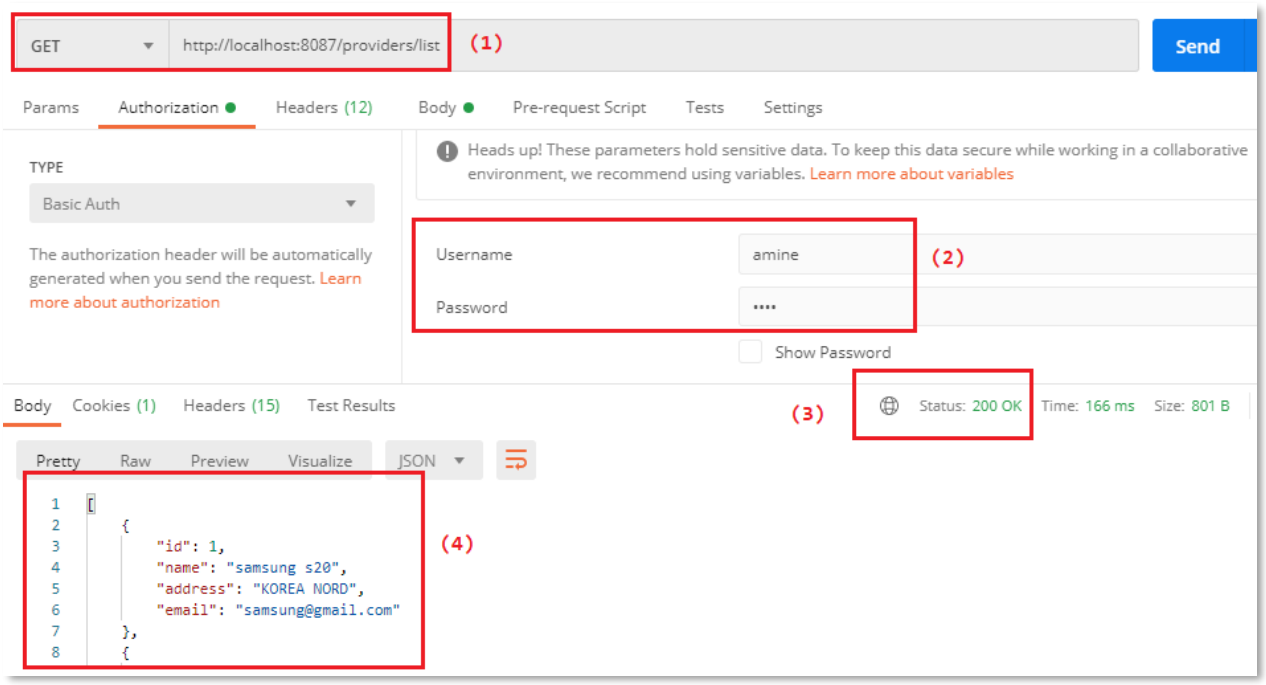
- **Testing above Security Implementation using Postman Rest Client**

- For incorrect credentials



- With correct credentials

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



Frontend Side

- Let's first update our authentication service

Provider.ts service

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders  } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
export class ProviderService {


  urlProviders = 'http://127.0.0.1:86/providers';

  provider: any;
  username = sessionStorage.getItem('username');
  password = sessionStorage.getItem('password');
  constructor(private Http: HttpClient) { }

  listProviders() {

    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usernam
e + ':' + this.password) });
    return this.Http.get(this.urlProviders + '/list', { headers });
  }


  createProvider(myform) {

    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usernam
e + ':' + this.password) });

    this.provider = {
      'name': myform.value.providerName,
      'email': myform.value.providerEmail,
      'address': myform.value.providerAdress
    }
    return this.Http.post(this.urlProviders + '/add', this.provider,{ headers });
  }

  updateProvider(myObj) {
    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usernam
e + ':' + this.password) });

    return this.Http.put(this.urlProviders + '/' + myObj['id'], myObj, { headers
});
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
  }


  deleteProvider(myObj) {
    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usernam
e + ':' + this.password) });

    return this.Http.delete(this.urlProviders + '/' + myObj['id'],{ headers })
  }

  getProvider(id) {
    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usernam
e + ':' + this.password) });

    return this.Http.get(this.urlProviders + '/' + id, { headers })
  }
}
```

Authentication.ts service

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { map } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class AuthenticationService {

  constructor(private httpClient: HttpClient) { }

  authenticate(username, password) {
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(username + '
:' + password) });
    return this.httpClient.get('http://localhost:86/basicauth', { headers }).pipe
(
      map(
        userData => {
          sessionStorage.setItem('username', username);
          sessionStorage.setItem('password', password);
          console.log(username + " " + password);
          return userData;
        }
      )

    );
    /*
    if (username === "amine" && password === "1234") {


      sessionStorage.setItem('username', username)
      return true;
    } else {
      return false;
    }*/

  }

  isUserLoggedIn() {
    let user = sessionStorage.getItem('username')
    console.log(!(user === null))
    return !(user === null)
  }

  logOut() {
    sessionStorage.removeItem('username')
  }

}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

Login.ts component

```
import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { AuthenticationService } from '../services/authentication.service';


@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  username: string;
  password: string;
  invalidLogin = false;

  successMessage = "Authentication success";
  errorMessage = "Invalide username or password";
  constructor(private router: Router,
    private loginservice: AuthenticationService) { }

  ngOnInit() {
  }

  checkLogin() {

    (this.loginservice.authenticate(this.username, this.password).subscribe(
      data => {
        this.router.navigate([''])
        this.invalidLogin = false
```

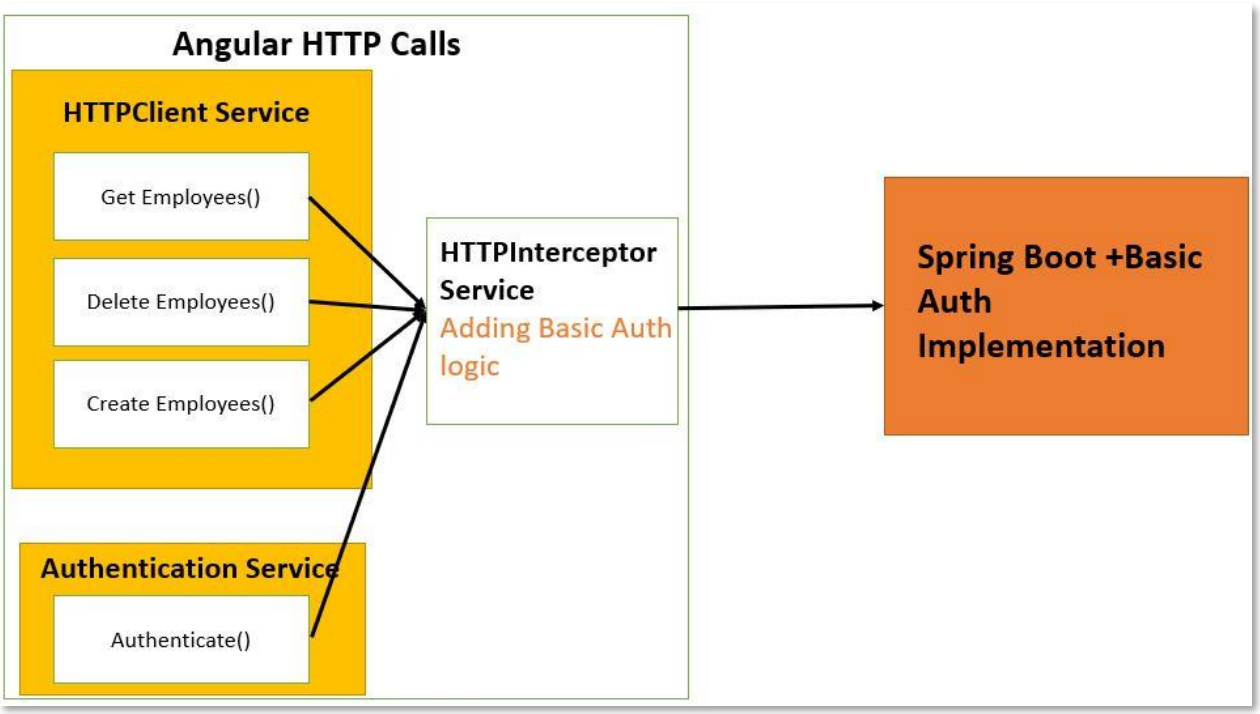
Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
      },
      error => {
        this.invalidLogin = true

      }
    )
    );
   /* if (this.loginservice.authenticate(this.username, this.password)) {
      this.router.navigate([''])
    } else
      this.invalidLogin = true*/

  }

}
```

Add and Configure HttpInterceptor

We had seen we had to duplicate the code for adding Basic Auth Headers to the HTTPRequest before making HTTP calls. In this section we will be implement a HTTPInterceptor which will intercept all outgoing HTTP requests.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



Implement changes for Basic Authentication on the Angular side

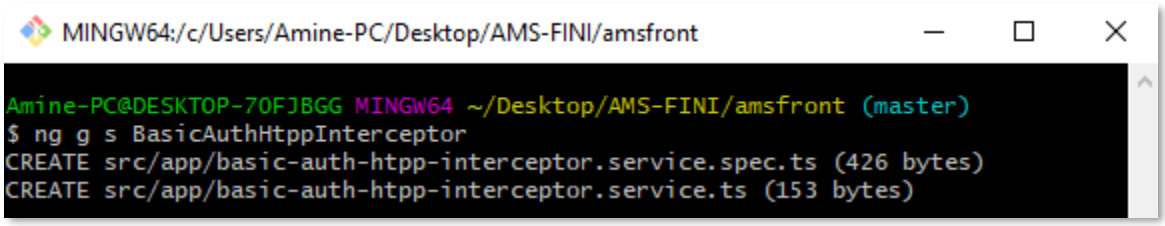
In the **authentication.service.ts** if the authentication for the user entered username and password is successful, we will be saving the basicAuth string which we are adding the Authorization Header for basic Authenication in the session.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
 authenticate(username, password) {

    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(username + '
:' + password) });
    return this.httpClient.get('http://localhost:86/basicauth', { headers }).pipe
(
      map(
        userData => {
          sessionStorage.setItem('username', username);
          sessionStorage.setItem('password', password);
          console.log(username + " " + password);
          let authString = 'Basic ' + btoa(username + ':' + password);
          sessionStorage.setItem('basicauth', authString);
          return userData;
        }
      )

    );
    /*
    if (username === "amine" && password === "1234") {


      sessionStorage.setItem('username', username)
      return true;
    } else {
      return false;
    }*/

  }
```

Next we will be creating a new **HttpInterceptor service** called **BasicAuthInterceptor Service**. This service will check if the session has valid **username and basicAuth String**, then it will update the headers of all outgoing HTTP requests. We implement the interceptor by extending the HttpInterceptor.

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



The content of the service:

```
import { Injectable } from '@angular/core';
import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/common/http';
import { AuthenticationService } from './services/authentication.service';

@Injectable({
  providedIn: 'root'
})
export class BasicAuthHtppInterceptorService implements HttpInterceptor {

  constructor() { }

  intercept(req: HttpRequest<any>, next: HttpHandler) {

    if (sessionStorage.getItem('username') && sessionStorage.getItem('basicauth')
) {
      req = req.clone({
        setHeaders: {
          Authorization: sessionStorage.getItem('basicauth')
        }
      })
    }

    return next.handle(req);

  }
}
```

Now we will register the created HTTPInterceptor using the app.module.ts by updating it in the provider section.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NavbarComponent } from './navbar/navbar.component';
import { AddProviderComponent } from './add-provider/add-provider.component';
import { ListProviderComponent } from './list-provider/list-provider.component';
import { UpdateProviderComponent } from './update-provider/update-
provider.component';
import { LoginComponent } from './login/login.component';
import { LogoutComponent } from './logout/logout.component';

import { BasicAuthHtppInterceptorService } from './basic-auth-htpp-
interceptor.service';
import { HTTP_INTERCEPTORS } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    NavbarComponent,
    AddProviderComponent,
    ListProviderComponent,
    UpdateProviderComponent,
    LoginComponent,
    LogoutComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
    HttpClientModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: BasicAuthHtppInterceptorService,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Finally we will remove the hardcoded basic auth logic from the Http client service. So the **provider.ts** service is as follows:

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders  } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class ProviderService {


  urlProviders = 'http://127.0.0.1:86/providers';

  provider: any;
  username = sessionStorage.getItem('username');
  password = sessionStorage.getItem('password');
  constructor(private Http: HttpClient) { }

  listProviders() {
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
    //const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usern
ame + ':' + this.password) });
    //return this.Http.get(this.urlProviders + '/list', { headers });
    return this.Http.get(this.urlProviders + '/list');
  }


  createProvider(myform) {

    //const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usern
ame + ':' + this.password) });

    this.provider = {
      'name': myform.value.providerName,
      'email': myform.value.providerEmail,
      'address': myform.value.providerAdress
    }
    //return this.Http.post(this.urlProviders + '/add', this.provider, { headers
});
    return this.Http.post(this.urlProviders + '/add', this.provider);
  }

  updateProvider(myObj) {
   // const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usern
ame + ':' + this.password) });

    //return this.Http.put(this.urlProviders + '/' + myObj['id'], myObj, { header
s });
    return this.Http.put(this.urlProviders + '/' + myObj['id'], myObj);
  }


  deleteProvider(myObj) {
    //const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usern
ame + ':' + this.password) });

    //return this.Http.delete(this.urlProviders + '/' + myObj['id'], { headers })
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
    return this.Http.delete(this.urlProviders + '/' + myObj['id'])
  }

  getProvider(id) {
   // const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(this.usern
ame + ':' + this.password) });

    //return this.Http.get(this.urlProviders + '/' + id, { headers })
    return this.Http.get(this.urlProviders + '/' + id)
  }
}
```

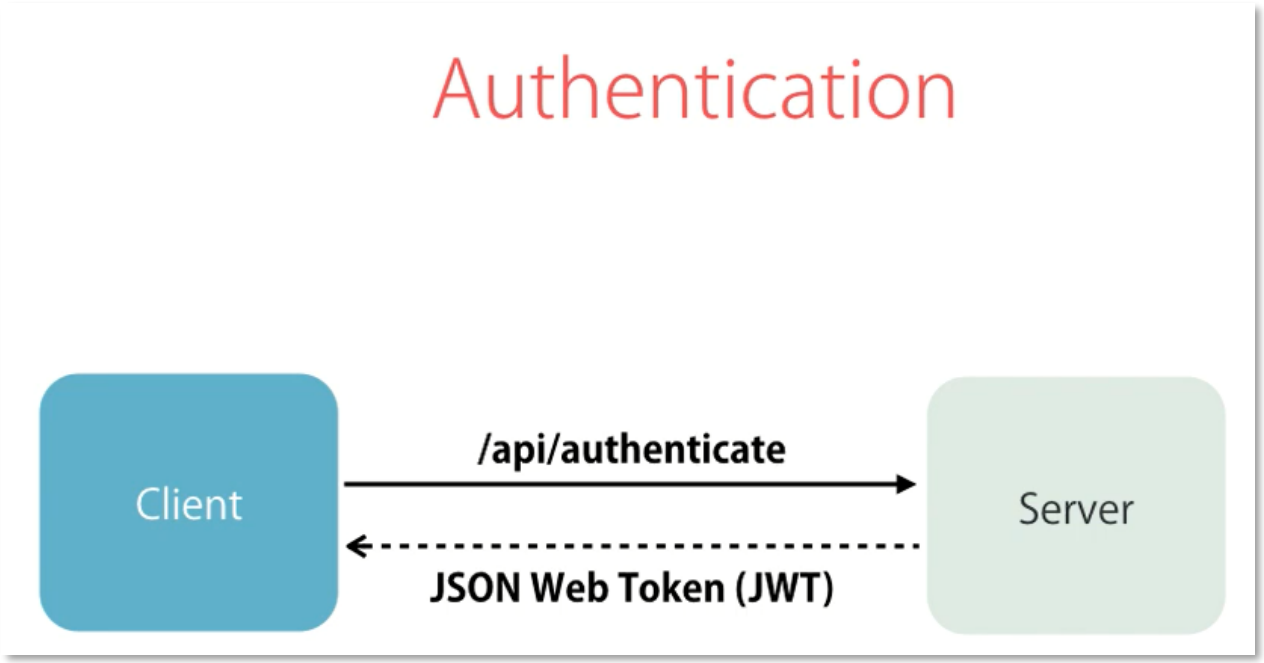
Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- Part 3 : JWT(backend)

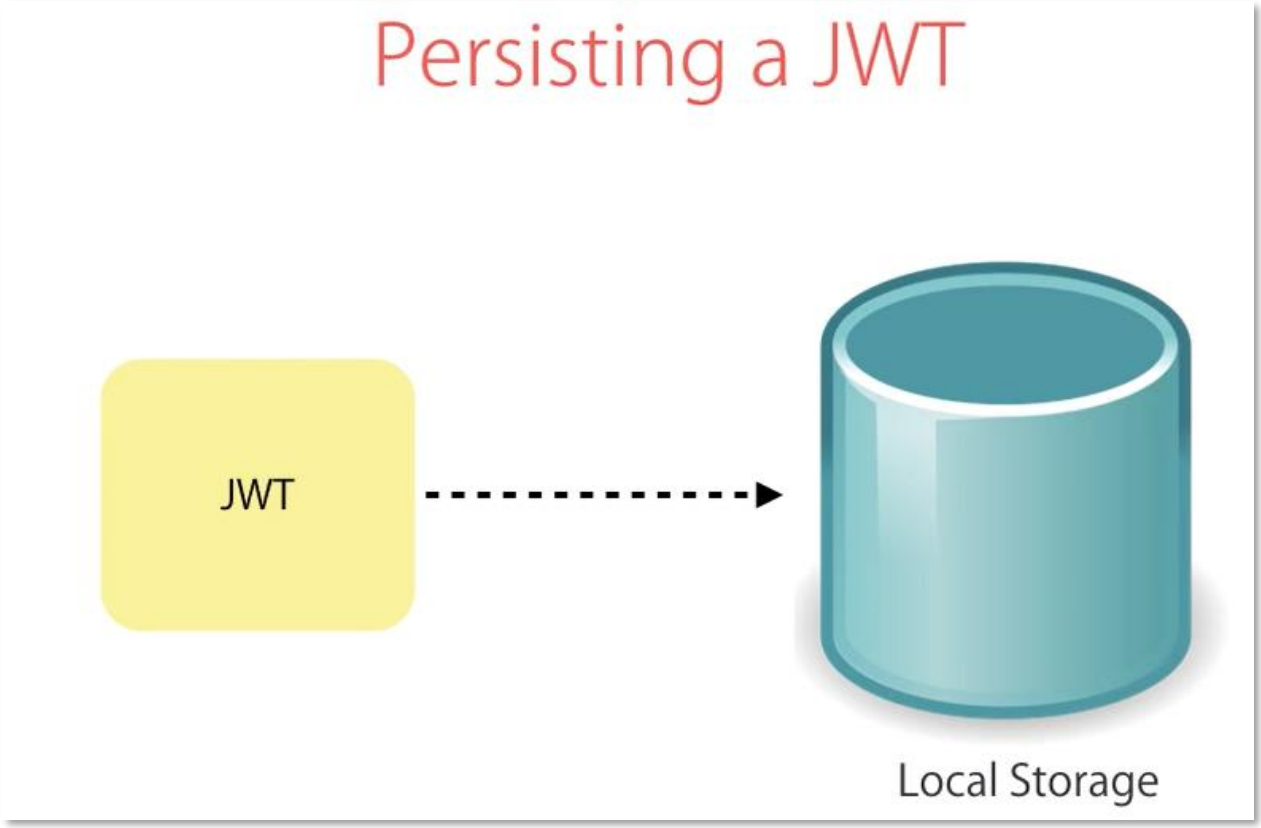
In this section...

- JSON Web Tokens (JWT)
- Stateless authentication
- Protect routes
- Redirect the users to a “login” or “access denied” page
- Show/hide elements
- Get the current user

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



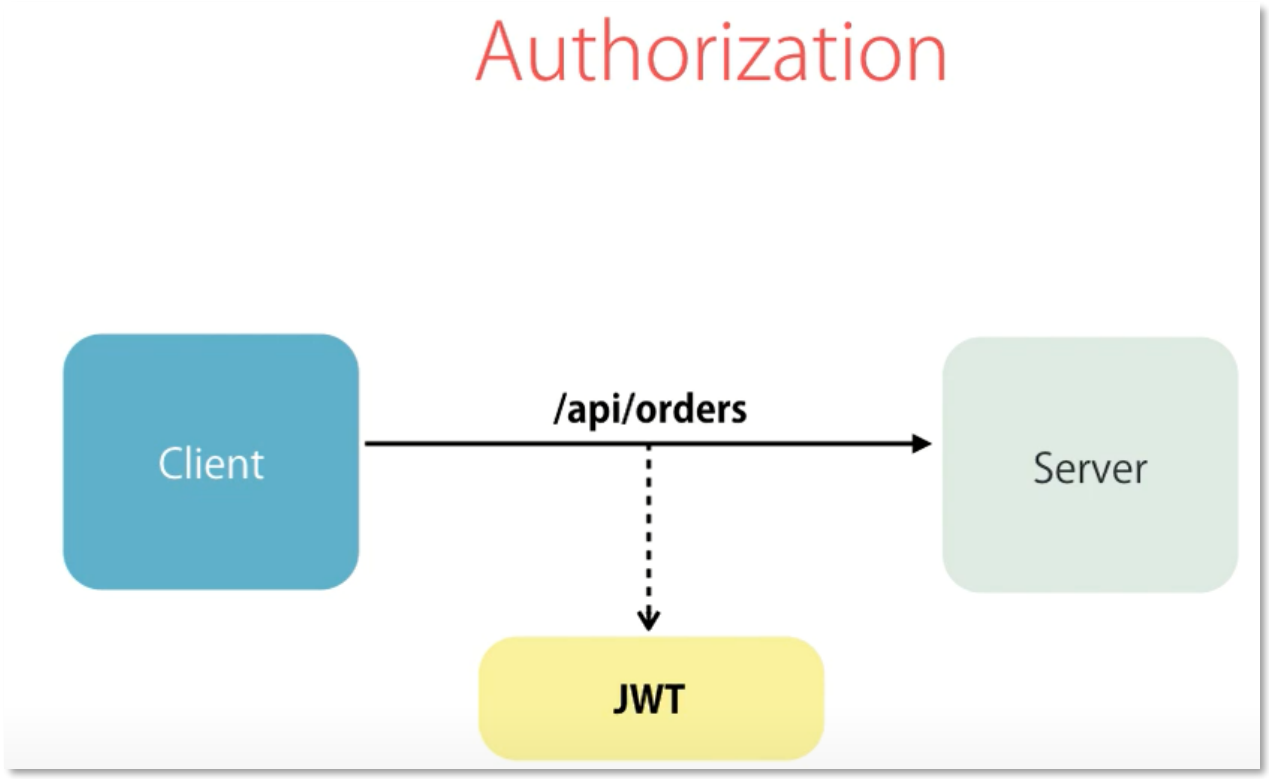
Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



JWT on the Client

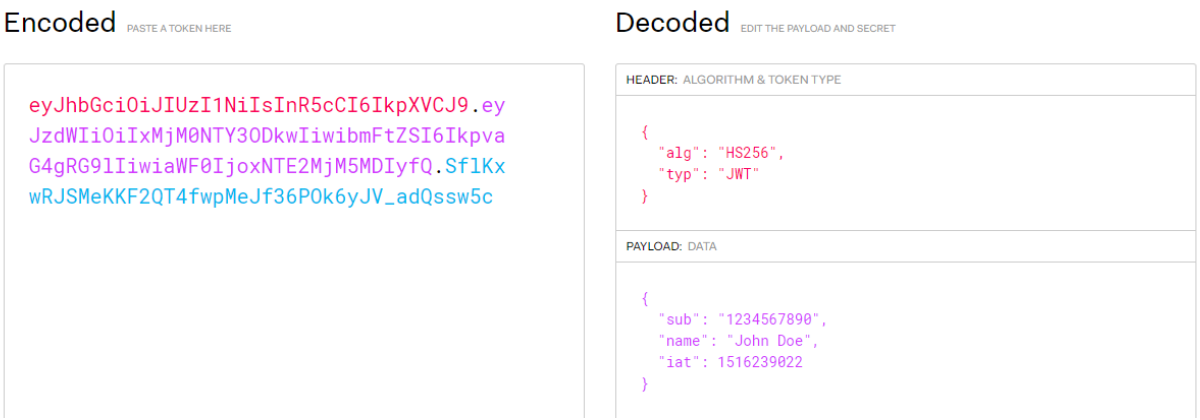
- Display current user's name
- Show/hide parts of a page
- Prevent access to certain routes

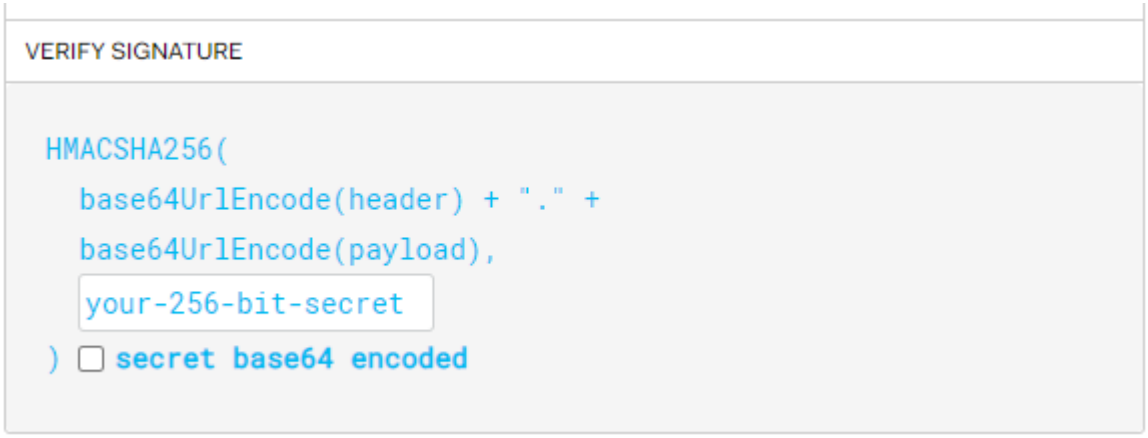
Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



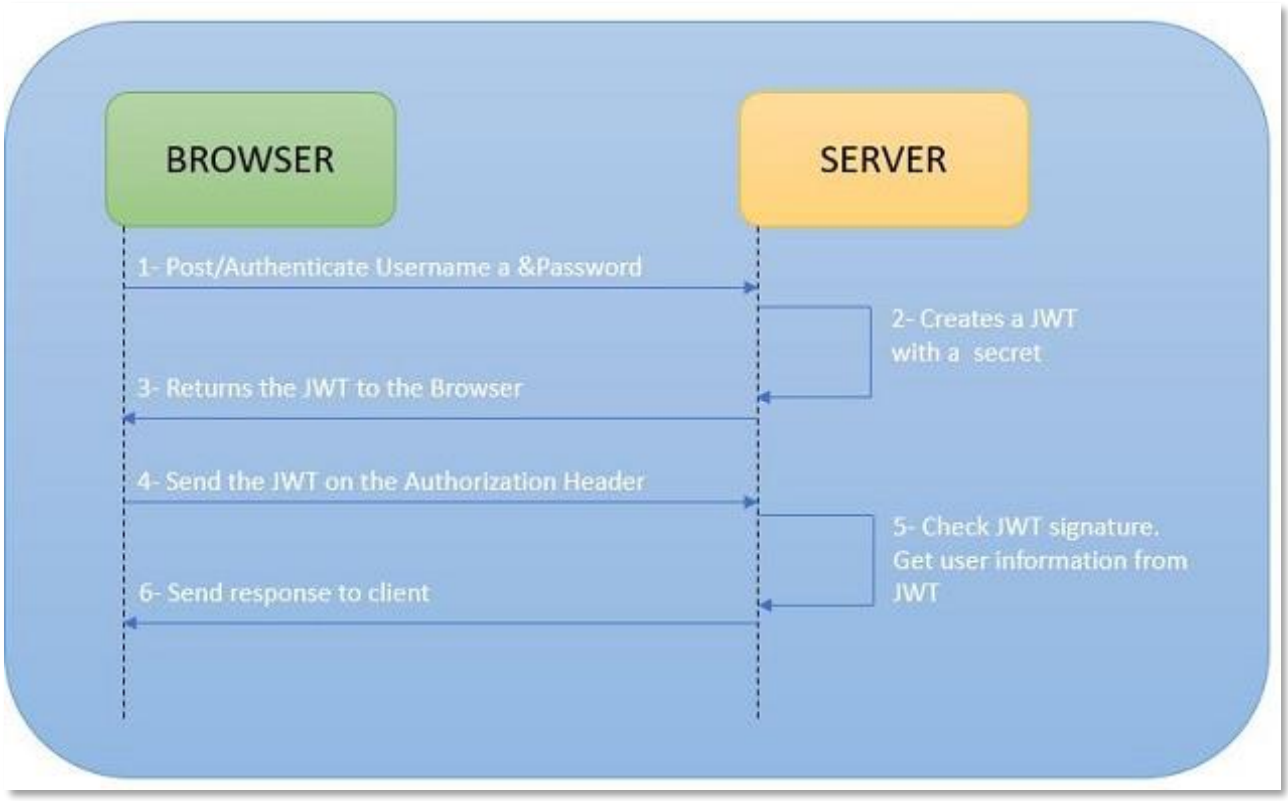
https://jwt.io/

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

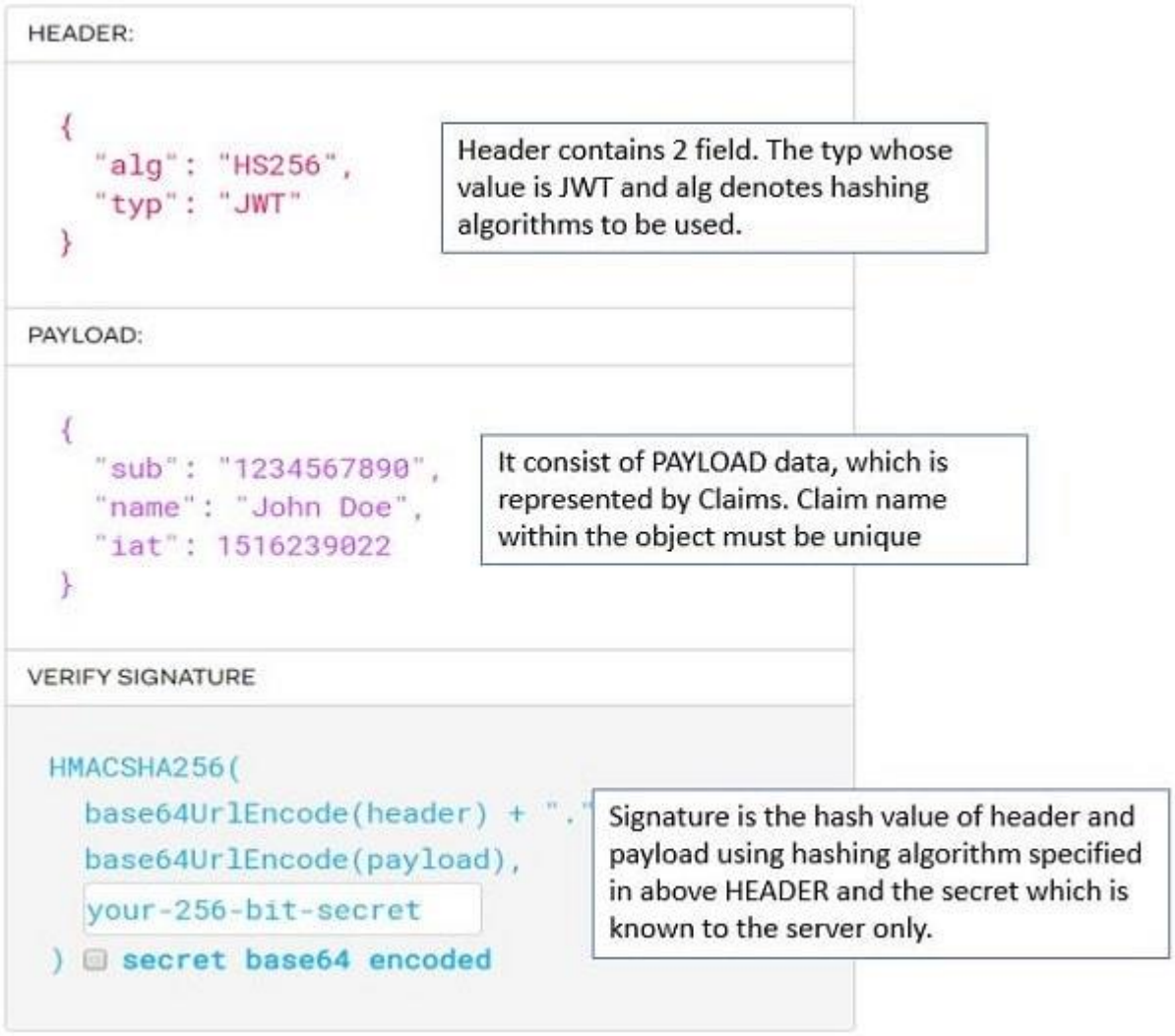




Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



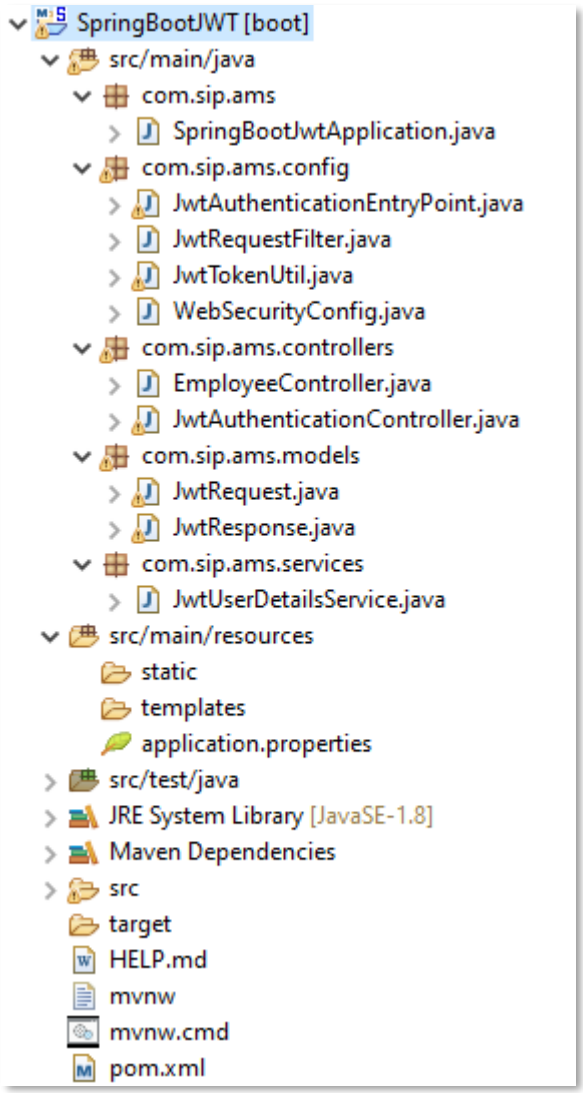
Create JWT Token Online

Will generate JWT Token by using JWT Online Token Generator.

http://jwtbuilder.jamiekurtz.com/

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- The project structure at the end



Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- ***Create Simple Spring boot with /greeting rest end point***

I will use at the beginning only the web dependency

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.3.4.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.sip</groupId>
	<artifactId>SpringBootJWT</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>SpringBootJWT</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
			<exclusions>
				<exclusion>
					<groupId>org.junit.vintage</groupId>
					<artifactId>junit-vintage-engine</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
```

```
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Then let's add the controller

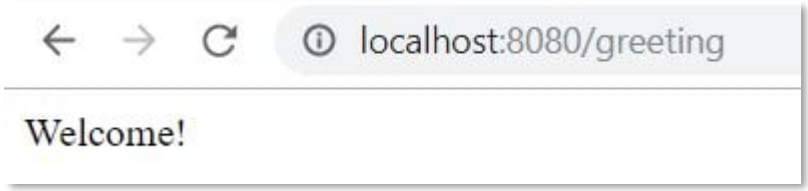
```
package com.sip.ams.controllers;


import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
public class EmployeeController {
	@RequestMapping({ "/greeting" })
	public String welcomePage() {
		return "Welcome!";
	}
```

```
}
```

That's all.

Test* /greeting *GET Api without JWT

Compile and the run this project by using endpoint localhost:8080/greeting.



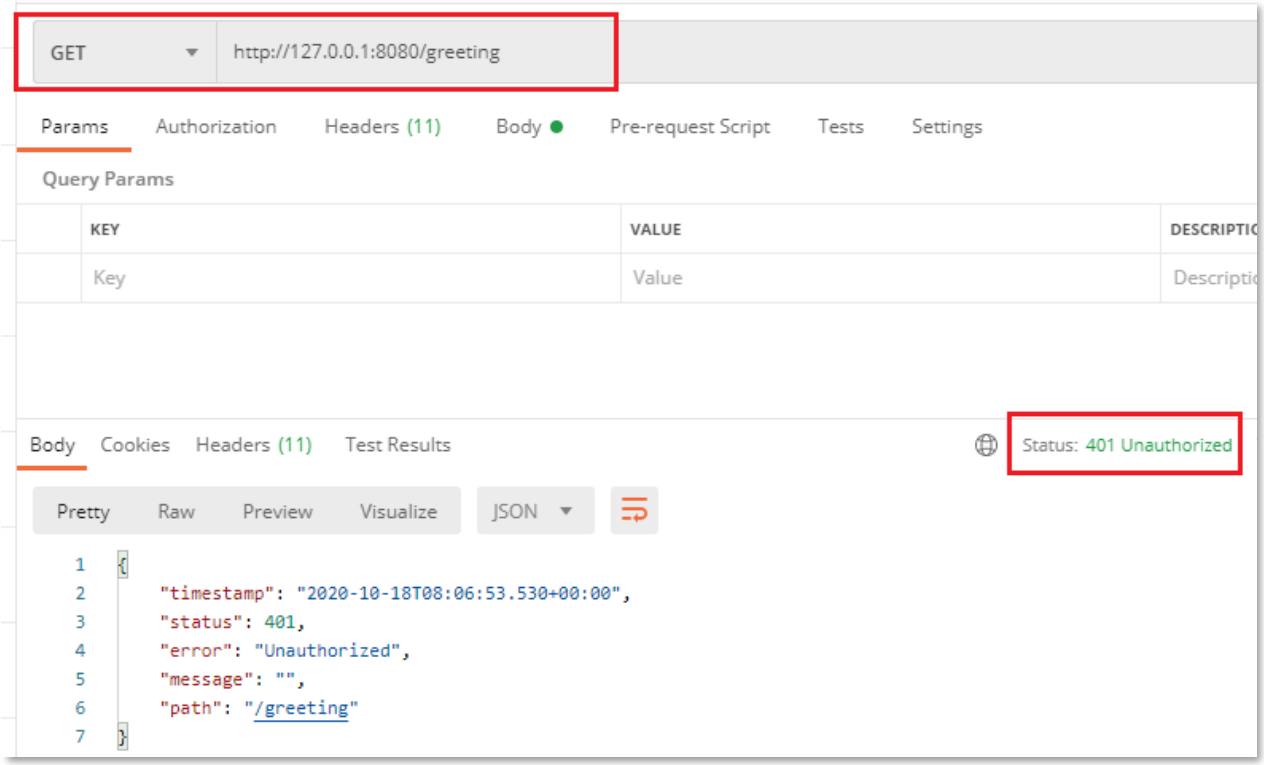
➔Now we will add spring security and JWT into our project.

pom.xml:

Add Spring Security and JWT dependencies as given below.

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
	<groupId>io.jsonwebtoken</groupId>
	<artifactId>jjwt</artifactId>
	<version>0.9.1</version>
</dependency>
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**



- **The service**

```
package com.sip.ams.services;
```

```
import java.util.ArrayList;


import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;


@Service

public class JwtUserDetailsService implements UserDetailsService {


	@Override

	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

		if ("med".equals(username)) {

			return new User("med",
"$2a$10$slYQmyNdGzTn7ZLBXBChFOC9f6kFjAqPhccnP6DxlWXx2lPk1C3G6",new
ArrayList<>());

		} else {

			throw new UsernameNotFoundException("User not found with
username: " + username);

		}

	}
```

```
}
```

- **The JWT Utility class**

```
package com.sip.ams.config;

import java.io.Serializable;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;


import org.springframework.beans.factory.annotation.Value;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Component;


import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;


@Component

public class JwtTokenUtil implements Serializable {
```

```
	public static final long JWT_TOKEN_VALIDITY = 5*60*60;



	private String secret="ams2020";

	public String getUsernameFromToken(String token) {
		return getClaimFromToken(token, Claims::getSubject);
	}

	public Date getIssuedAtDateFromToken(String token) {
		return getClaimFromToken(token, Claims::getIssuedAt);
	}

	public Date getExpirationDateFromToken(String token) {
		return getClaimFromToken(token, Claims::getExpiration);
	}

	public <T> T getClaimFromToken(String token, Function<Claims, T>
claimsResolver) {
		final Claims claims = getAllClaimsFromToken(token);
		return claimsResolver.apply(claims);
```

```
    }

    private Claims getAllClaimsFromToken(String token) {
        return
Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();
    }

    private Boolean isTokenExpired(String token) {
        final Date expiration = getExpirationDateFromToken(token);
        return expiration.before(new Date());
    }

    private Boolean ignoreTokenExpiration(String token) {
        // here you specify tokens, for that the expiration is ignored
        return false;
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return doGenerateToken(claims, userDetails.getUsername());
    }
```

```
	private String doGenerateToken(Map<String, Object> claims, String subject) {


		return
Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
.setExpiration(new Date(System.currentTimeMillis() +
JWT_TOKEN_VALIDITY*1000)).signWith(SignatureAlgorithm.HS512,
secret).compact();
	}


	public Boolean canTokenBeRefreshed(String token) {
		return (!isTokenExpired(token) || ignoreTokenExpiration(token));
	}


	public Boolean validateToken(String token, UserDetails userDetails) {
		final String username = getUsernameFromToken(token);
		return (username.equals(userDetails.getUsername()) &&
!isTokenExpired(token));
	}
}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- # The JWT Authentication Controller

```
package com.sip.ams.controllers;


import java.util.Objects;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;


import com.sip.ams.config.JwtTokenUtil;
```

```
import com.sip.ams.models.JwtRequest;
import com.sip.ams.models.JwtResponse;


@RestController
@CrossOrigin(origins = "*")
public class JwtAuthenticationController {


	@Autowired
	private AuthenticationManager authenticationManager;


	@Autowired
	private JwtTokenUtil jwtTokenUtil;


	@Autowired
	private UserDetailsService jwtInMemoryUserDetailsService;


	@RequestMapping(value = {"/auth"}, method = RequestMethod.POST)


	public ResponseEntity<?> generateAuthenticationToken(@RequestBody
JwtRequest authenticationRequest)
				throws Exception {
```

```
		authenticate(authenticationRequest.getUsername(),
authenticationRequest.getPassword());


		final UserDetails userDetails = jwtInMemoryUserDetailsService

	.loadUserByUsername(authenticationRequest.getUsername());


		final String token = jwtTokenUtil.generateToken(userDetails);


		return ResponseEntity.ok(new JwtResponse(token));
	}


	private void authenticate(String username, String password) throws Exception {
		try {
			authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(username, password));
		} catch (DisabledException e) {
			throw new Exception("USER_DISABLED", e);
		} catch (BadCredentialsException e) {
			throw new Exception("INVALID_CREDENTIALS", e);
		}
	}
}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- **The Request Object**

```
package com.sip.ams.models;

import java.io.Serializable;

public class JwtRequest implements Serializable {

	private String username;
	private String password;

	//default constructor for JSON Parsing
	public JwtRequest()
	{
	}

	public JwtRequest(String username, String password) {
		this.setUsername(username);
		this.setPassword(password);
	}

	public String getUsername() {
		return this.username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public String getPassword() {
		return this.password;
	}

	public void setPassword(String password) {
		this.password = password;
	}
}
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

- **The Response Object**

```
package com.sip.ams.models;

import java.io.Serializable;

public class JwtResponse implements Serializable {

	private final String jwttoken;

	public JwtResponse(String jwttoken) {
		this.jwttoken = jwttoken;
	}

	public String getToken() {
		return this.jwttoken;
	}
}
```

- **The JWT Request Filter**

```
package com.sip.ams.config;


import java.io.IOException;


import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;


import com.sip.ams.services.JwtUserDetailsService;


import io.jsonwebtoken.ExpiredJwtException;


@Component
public class JwtRequestFilter extends OncePerRequestFilter {


	@Autowired
	private JwtUserDetailsService jwtUserDetailsService;


	@Autowired
	private JwtTokenUtil jwtTokenUtil;
```

```
	@Override
	protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)
			throws ServletException, IOException {


		final String requestTokenHeader = request.getHeader("Authorization");


		String username = null;
		String jwtToken = null;
		// JWT Token is in the form "Bearer token". Remove Bearer word and get only the Token
		if (requestTokenHeader != null && requestTokenHeader.startsWith("Bearer ")) {
			jwtToken = requestTokenHeader.substring(7);
			try {
				username = jwtTokenUtil.getUsernameFromToken(jwtToken);
			} catch (IllegalArgumentException e) {
				System.out.println("Unable to get JWT Token");
			} catch (ExpiredJwtException e) {
				System.out.println("JWT Token has expired");
			}
		} else {
```

```
                    logger.warn("JWT Token does not begin with Bearer String....");
                }


                //Once we get the token validate it.
                if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {


                    UserDetails userDetails =
this.jwtUserDetailsService.loadUserByUsername(username);


                    // if token is valid configure Spring Security to manually set
authentication
                    if (jwtTokenUtil.validateToken(jwtToken, userDetails)) {


                        UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken = new
UsernamePasswordAuthenticationToken(
                                userDetails, null, userDetails.getAuthorities());
                        usernamePasswordAuthenticationToken
                                .setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
                        // After setting the Authentication in the context, we specify
                        // that the current user is authenticated. So it passes the
Spring Security Configurations successfully.
```

```
        SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthe
ntication Token);
                        }
                }
                chain.doFilter(request, response);
        }

}
```

- **The JWTAuthenticationEntryPoint**

⇨ This class rejects unauthenticated request and send code 401

```
package com.sip.ams.config;

import java.io.IOException;
import java.io.Serializable;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
```

```
import org.springframework.stereotype.Component;


@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint, Serializable {




	@Override
	public void commence(HttpServletRequest request, HttpServletResponse response,
				AuthenticationException authException) throws IOException {


		response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");
	}
}
```

- **The WebSecurityConfig**

```
package com.sip.ams.config;


import org.springframework.beans.factory.annotation.Autowired;
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import
org.springframework.security.config.annotation.authentication.builders.Authentication
ManagerBuilder;

import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMe
thodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfig
urerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFil
ter;


@Configuration
```

Trainning: Springboot & Angular 9 **Trainner:** Dr. Mohamed Amine MEZGHICH **Period of trainning :**30 Hours **Email :** ma.mezghich@smart-it-partner.com **Phone :** +216 51 36 36 34	**Workshop n° 5: Angular 9** **Goals:** Authentication and Authorization STEP 1 : HardCoded Authentication STEP 2 : Basic Authentication STEP 3 : JSON WEB TOKEN (JWT) **August Session, 2020**

```
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {


	@Autowired
	private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;


	@Autowired
	private UserDetailsService jwtUserDetailsService;


	@Autowired
	private JwtRequestFilter jwtRequestFilter;


	@Autowired
	public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
		// configure AuthenticationManager so that it knows from where to load
		// user for matching credentials
		// Use BCryptPasswordEncoder

	auth.userDetailsService(jwtUserDetailsService).passwordEncoder(passwordEnco
der());
	}
```

```
@Bean
public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
}

@Bean
@Override
public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
}

@Override
protected void configure(HttpSecurity httpSecurity) throws Exception {
        // We don't need CSRF for this example
        httpSecurity.csrf().disable()
                        // dont authenticate this particular request
                        .authorizeRequests().antMatchers("/auth").permitAll()
                        .antMatchers(HttpMethod.OPTIONS, "/**")
                        .permitAll().
                        // all other requests need to be authenticated
                        anyRequest().authenticated().and().
```

```
                // make sure we use stateless session; session won't be used
to

                // store user's state.



        exceptionHandling().authenticationEntryPoint(jwtAuthenticationEntryPoint).and
().sessionManagement()

                .sessionCreationPolicy(SessionCreationPolicy.STATELESS);


                // Add a filter to validate the tokens with every request

                httpSecurity.addFilterBefore(jwtRequestFilter,
UsernamePasswordAuthenticationFilter.class);

        }

}
```

The end.