Date : 16-Jun-21
Spring Boot 9AM
Mr. RAGHU

Spring Security REST + JWT + Angular

Application:
https://github.com/javabyraghu/SpringBoot2RestJwtSecurityEx
BackEnd Application (Concept names)

> Spring REST [Spring Web]
> Data JPA
> MySQL
> Spring Security [Stateless]
> JWT (JSON Web Token)

https://github.com/javabyraghu/angular-springboot-jwt
FrontEnd Application [Angular](Concepts)
> Components
> Models
> Services
> Interceptors
> Routing
> HttpClient
> Guards [Security Gurads]
> SessionStorage

POSTMAN REQUEST:
1. Creating user as admin/moderator

POST http://localhost:9900/api/auth/register SEND

```
{
 "username" : "sample1",
 "email"    : "sample1@gmail.com",
 "password" : "sample1",
 "role"     : [ "user", "admin"]
}
```

```
{
 "username" : "sample2",
 "email"    : "sample2@gmail.com",
 "password" : "sample2",
 "role"     : [ "user", "mod"]
}
```

---

2) Login checking:

POST http://localhost:9900/api/auth/login SEND

```
{
  "username" : "sample2",
  "password" : "sample2",
}
```

If login is valid JWT Token is generated , copy same and

```
  and send header param for next request onwards.

  Authorization = Bearer xxxxxxx.yyyyyyyyyy.zzzzz
-------------------------------------------------------
3. Access Other services
GET http://localhost:9900/api/home/all SEND


GET http://localhost:9900/api/home/user SEND
   Authorization
      type : Bearer Token

      Enter Token : xxxxxxx.yyyyyyyyyy.zzzzz

GET http://localhost:9900/api/home/admin SEND
   Authorization
      type : Bearer Token

      Enter Token : xxxxxxx.yyyyyyyyyy.zzzzz


GET http://localhost:9900/api/home/mod SEND
   Authorization
      type : Bearer Token

      Enter Token : xxxxxxx.yyyyyyyyyy.zzzzz

=========================================================================
=======
1. Model class
     User  *----<> * Role

  SAM -- ADMIN, USER
  AJAY -- MOD, USER
  SYED -- USER
  VINAY -- ADMIN

class:
Role [in.nareshit.raghu.model]

roles_table [ROLE_ADMIN,ROLE_USER,ROLE_MODERATOR]

enum ERole {
        ROLE_USER,
        ROLE_MODERATOR,
        ROLE_ADMIN
}

*) Inside Role Model class
       @Enumerated(EnumType.STRING)
        private ERole name;

=> We can store enum value as String converted into Database table.
-------------------------------------------------------------------
----
We we do user register from frontend application , they will JSON
looks like
```

```
{
 "username" : "sample1",
 "email"    : "sample1@gmail.com",
 "password" : "sample1",
 "role"     : [ "user", "admin"]
}
```

this data must be converted into one class Object, class is:
SignupRequest

Angular -->Register --> JSON --> Object(SignupRequest)--> REST--> Converted
---> User & Roles --> Store in DB
--> Success--> Object(MessageResponse)-->JSON--> Angular

Angular -->Login --> JSON --> Object(LoginRequest)--> REST--> Converted
---> UsernamePasswordAuthenticationToken --> login validation
--> Success --> object(JwtResponse) -->JSON --> Angular

---

2. Repository
RoleRepository :
To create/modify/remove Roles data

```
//select * from roletab where rname=?
Optional<Role> findByName(ERole name);
```

UserRepository

```
//select * from usertab where uname=?
Optional<User> findByUsername(String username);

//select count(uname)>0 from usertab where uname=?
boolean existsByUsername(String username);

//select count(email)>0 from usertab where email=?
boolean existsByEmail(String email);
```

3. Service
UserDetailsServiceImpl (: UserDetailsService)
This is called when we try to login, check given user exist in Db or not.
if exist then convert Model class User Object to Spring Security (User object)
[or define your own class : UserDetails]

4. RestController

a. HomeRestController : is proving common services like /all, /user, /admin, /mod

```
.antMatchers("/home/all").permitAll()
.antMatchers("/home/mod").hasAuthority(ERole.ROLE_MODERATOR.name())
```

```
 .antMatchers("/home/admin").hasAuthority(ERole.ROLE_ADMIN.name())
 .anyRequest().authenticated()
```

b. AuthenticationRestController

```
   # authenticateUser() --> /auth/login
   # createUser() --> /auth/register

  .antMatchers("/auth**").permitAll()
```

---

5. Util

6. Java Config files

7. Filter

8. Security

9. properties