Date : 18-Jun-21
Spring Boot 9AM
Mr. RAGHU

Security : JWT + Spring Boot + Angular

Q) How can we execute some logic in Spring Boot while starting Application?
A) CommandLineRunner.

*) On APplication starup insert Roles into Database table.
  If given role is not exist in DB.

```
--Manual SQL queries--
INSERT INTO ROLES VALUES(1,'ROLE_ADMIN');
INSERT INTO ROLES VALUES(2,'ROLE_USER');
INSERT INTO ROLES VALUES(3,'ROLE_MODERATOR');
```

--Automated by code--------------
*) SetupRunner class is executed when we start application.
  It will insert ROLES DATA INTO TABLES, if not present.

```
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.ERole;
import in.nareshit.raghu.model.Role;
import in.nareshit.raghu.repo.RoleRepository;

@Component
@Order(Ordered.HIGHEST_PRECEDENCE)
public class SetupRunner implements CommandLineRunner {

        @Autowired
        private RoleRepository repository;

        public void run(String... args) throws Exception {

if(!repository.findByName(ERole.ROLE_USER).isPresent()) {
                        Role r = new Role();
                        r.setName(ERole.ROLE_USER);
                        repository.save(r);
                }

if(!repository.findByName(ERole.ROLE_ADMIN).isPresent()) {
                        Role r = new Role();
                        r.setName(ERole.ROLE_ADMIN);
                        repository.save(r);
                }
if(!repository.findByName(ERole.ROLE_MODERATOR).isPresent()) {
                        Role r = new Role();
                        r.setName(ERole.ROLE_MODERATOR);
                        repository.save(r);
                }
        }
}
==============================================================
```

5. Util

JWT UTIL#Token Generation(login case)

```
Jwts.builder()
.setSubject(userDetails.getUsername())
.setIssuedAt(new Date())
.setExpiration(new Date(System.currentTimeMillis() + jwtExpTimeInMs))
.signWith(SignatureAlgorithm.HS512, jwtSecret)
.compact();
```

JWT UTIL#TOKEN VAIDATION(after login case)

```
 Jwts.parser()
        .setSigningKey(jwtSecret)
        .parseClaimsJws(token)
        .getBody()
        .getSubject();
```

*)RolUtil:

User Roles are String types like "admin", "mod","user".

While register Person can have multiple roles or zero roles.
If they give multiple roles ex: "user","admin" then they are
converted into Role(ROLE_USER), Role(ROLE_ADMIN).

If empty/null given by user then default given as Role(ROLE_USER).

```
           User       ------       Role
        10     A               55  ROLE_ADMIN
        11     B

                  userroletab
                     10  55
                     11  55
--------------------------------------------------
```

6. Java Config files

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

}
```

*) Using Password Encoder for register/login process
for enduser password.

7. Filter:
=> For every request JWT token is validated.(not for permitAll methods)
=> it is also type of login checking
=> First read toke and validate
=> if valid read subject and load UserObject
=> Create Authentication, set to SecurityContext
=> Continue to FC.

8. Security
Use UserDetailsService
Use PasswordEncoder

DO NOT USE FORM
DO NOT SPECIFY LOGOUT (Token Expiary/delete token)

PROVIDE AS NO HTTP SESSION

```
              .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
```

REGISTER FILTER (name, type)

```
 http.addFilterBefore(securityFilter(), UsernamePasswordAuthenticationFilter.class);
```

DISABLE CSRF as we are using TOKEN (JWT)

```
 .csrf().disable()
```

Create AuthenticationManager (not created in stateless)

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

Create object to SecurityFilter

```
@Bean
public SecurityFilter securityFilter() {
       return new SecurityFilter();
}
```

LOGIN FAILED??
AuthenticationEntryPoint

----

*) Cross Origin :- It is a process of restricting Clients to make request for backend procesing
in simple which clients can make request to out backend application.

a. Using annotation
b.*** Global Configuration