Date : 10-Jun-21
Spring Boot 9AM
Mr. RAGHU

Spring Security using ORM

1. User Register Process
2. User Login and PasswordEncoder
3. Custom Login Page
4. Session Management View
5. CSRF View

Stage#1. User Register Process

>> Register page must contain 3 inputs-'email, password and roles'.
[additional inputs also ok].

>> ID is auto-generated. For Roles we need to create Collection variable
[it is checkbox input]

>> For every collection one child table is created, with 2 columns here.
ie Key Column(Join Column), Element Column(Column)

---coding steps---

Name : SpringBoot2SecurityOrmEx
Dep : Web, lombok, Devtools, MySQL, Data JPA, thymeleaf,

1. properties

```
server.port=8081

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot9am
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

2. Model class

```
package in.nareshit.raghu.model;

import java.util.Set;

import javax.persistence.CollectionTable;
import javax.persistence.Column;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
```

```
import javax.persistence.Table;

import lombok.Data;

@Data
@Entity
@Table(name="user_tab")
public class User {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name="uid")
        private Integer id;

        @Column(name="uname")
        private String userName;

        @Column(name="umail")
        private String userMail;

        @Column(name="upwd")
        private String userPwd;

        @ElementCollection
        @CollectionTable(
                        name="roles_tab",
                        joinColumns = @JoinColumn(name="uid")
                        )
        @Column(name="urole")
        private Set<String> userRoles;

}
```

---

3. Repository

```
package in.nareshit.raghu.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import in.nareshit.raghu.model.User;

public interface UserRepository extends JpaRepository<User, Integer> {

        //SQL: select * from user where uname=?
        Optional<User> findByUserName(String userName);
}
```

---

4. Service Interface

```
package in.nareshit.raghu.service;

import java.util.Optional;
import in.nareshit.raghu.model.User;

public interface IUserService {

        Integer saveUser(User user);
        Optional<User> findUserbyName(String username);
}
```

---

5. ServiceImpl

```
package in.nareshit.raghu.service.impl;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import in.nareshit.raghu.model.User;
import in.nareshit.raghu.repo.UserRepository;
import in.nareshit.raghu.service.IUserService;

@Service
public class UserServiceImpl implements IUserService {

	@Autowired
	private UserRepository repo;

	public Integer saveUser(User user) {
		user = repo.save(user);
		return user.getId();
	}

	public Optional<User> findUserbyName(String username) {
		return repo.findByUserName(username);
	}

}
```

---

6. Controller

```
package in.nareshit.raghu.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import in.nareshit.raghu.model.User;
import in.nareshit.raghu.service.IUserService;

@Controller
@RequestMapping("/user")
public class UserController {

	@Autowired
	private IUserService service;

	//1. show Register page
	@GetMapping("/register")
	public String showReg() {

		return "UserRegister";
	}

	//2. on click submit button
```

```
        @PostMapping("/save")
        public String saveUser(
                        @ModelAttribute User user,
                        Model model)
        {
                Integer id = service.saveUser(user);
                model.addAttribute("message", "User '"+id+"'
created");
                return "UserRegister";
        }

}
```

7. Register Page : UserRegister.html

```
<html xmlns:th="https://www.thymeleaf.org/">
        <head>
        </head>
        <body>
                <h3>User Register Page!</h3>
                <form th:action="@{/user/save}" method="POST">
                        <pre>
                                NAME  : <input type="text"
name="userName"/>
                                EMAIL : <input type="text"
name="userMail"/>
                                PASS  : <input type="password"
name="userPwd"/>
                                ROLES :
                                  <input type="checkbox"
name="userRoles" value="ADMIN"/> ADMIN
                                  <input type="checkbox"
name="userRoles" value="EMPLOYEE"/> EMPLOYEE
                                <input type="submit"
value="REGISTER"/>
                        </pre>
                </form>
                <div th:text="${message}"></div>
        </body>
</html>
```

---UserRegister.html(with UI Design)----------

```
<html xmlns:th="https://www.thymeleaf.org/">
<head>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.mi
n.css" >
</head>
<body>
        <div class="container">
                <div class="card">
                        <div class="card-header bg-primary text-white
text-center">
                                <h3>USER REGISTER PAGE!</h3>
                        </div>
                        <div class="card-body">
                                <form th:action="@{/user/save}"
```

```
method="POST">

                                        <div class="row">
                                                <div class="col-3">

<label>NAME</label>
                                                </div>
                                                <div class="col-6">
                                                        <input
type="text" name="userName" class="form-control" required/>
                                                </div>
                                        </div>

                                        <div class="row">
                                                <div class="col-3">

<label>EMAIL</label>
                                                </div>
                                                <div class="col-6">
                                                        <input
type="text" name="userMail" class="form-control" required/>
                                                </div>
                                        </div>

                                        <div class="row">
                                                <div class="col-3">

<label>PASS</label>
                                                </div>
                                                <div class="col-6">
                                                        <input
type="password" name="userPwd" class="form-control" required/>
                                                </div>
                                        </div>

                                        <div class="row">
                                                <div class="col-3">

<label>ROLES</label>
                                                </div>
                                                <div class="col-6">
                                                         <input
type="checkbox" name="userRoles" value="ADMIN"/> ADMIN
                                                         <input
type="checkbox" name="userRoles" value="EMPLOYEE"/> EMPLOYEE
                                                </div>
                                        </div>

                                    <input type="submit"
value="REGISTER" class="btn btn-success"/>
                                </form>
                        </div>
                        <div th:if="${message!=null}" class="card-
footer bg-info text-white">
                                <div th:text="${message}"></div>

                        </div>
```

```
                </div>
        </div>

        </body>
</html>

=========================================================================
=
                Stage #2. User Login and PasswordEncoder

> WebSecurityConfigurerAdapter call internally 'UserDetailsService'(I)
   #loadUserbyUsername(username) method by taking 'username'
   from login page to check user exist or not DB.


Q) Why UserDetailsService?
A) Here, it is used to load Database data into Spring Security
    User class object.

Q) Can we define multiple classes with same name?
A) YES. Packages must be different.

Q) Who will validate user exist or not?
A) WebSecurityConfigurerAdapter
   If valid create HttpSession, store userdata
   and redirect to defaultSuccessUrl.
   If invalid, redirect to Login page.

Q) What is the diff b/w Authority and GrantedAuthority?
A)
  Authority : Roles in project
  GrantedAuthority : Allocated Roles to user.

ex: BankApp, Roles/Authorities -- ADMIN, MANAGER, CLERK, CAHIER

    Employee#SAM -- Alloacated Role(GrantedAuthority) : MANAGER

Q) What is the diff b/w GrantedAuthority and DB Role?
A) In Database data is stored as String(role)
   But Spring Security converts String into 'GrantedAuthority'
   Where it is interface, so impl class is used:
SimpleGrantedAuthority

//DB Format
   String r1="ADMIN";

//Security Format
   GrantedAuthority ga1 = new SimpleGrantedAuthority(r1);

*) Multiple Roles are converted into Multiple GrantedAuthority objects
  and given as Set<GrantedAuthority> to Spring Security User object.

-----------UserDetailsService Impl code-------------------------------
------------------------------
package in.nareshit.raghu.service.impl;

import java.util.HashSet;
```

```
import java.util.Optional;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundExceptio
n;
import org.springframework.stereotype.Service;

import in.nareshit.raghu.model.User;
import in.nareshit.raghu.repo.UserRepository;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

	@Autowired
	private UserRepository repo;

	public UserDetails loadUserByUsername(String username)
			throws UsernameNotFoundException {
		//goto db and get Model class object
		Optional<User> opt = repo.findByUserMail(username);
		if(opt.isEmpty()) {
			throw new UsernameNotFoundException("User not
exist !!");
		} else {
			// read object if exist
			User user = opt.get();

			//user roles from DB
			Set<String> roles = user.getUserRoles();

			//converting role into Spring GrantedAuthority
			Set<GrantedAuthority> authorities = new
HashSet<>();
			for(String r:roles) {
				authorities.add(new
SimpleGrantedAuthority(r));
			}

			//converting into Spring Security user object
			return new
org.springframework.security.core.userdetails
					.User(
						username,
						user.getUserPwd(),
						authorities);
		}

	}
```

```
}
```