Date : 11-Jun-21
Spring Boot 9AM
Mr. RAGHU

Download :
https://www.mediafire.com/file/3hpeq0o4z5l522p/Spring+SecuritySessionBasedApps_9AM_11062021_RAGHU.zip/file

Spring Security using ORM

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

==========

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.

Q) can one class implements multiple interfaces?
A) YES.

 Here Our class UserServiceImpl implements IUserService,
UserDetailsService

Q) if method return type is interface, then what should we return?

A) Impl class object.
   Anonymous inner class
   Lambda Exp (if functional interface)
   Method Reference (if functional interface)


interface Sample {
  void show();
}
class A implements Sample { .... }

class Process  {
  void findData(){}
}

class Test {
   Sample getData(){

    // return new A();
     /*return new  Sample() {
       public void show() {}
     };*/
     return ()-> {};
     return Process::findData();
   }
}

---code------------
Set<String> roles = user.getUserRoles();

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

--equals to--(java # 8)
user.getUserRoles().stream()
    // one string --> one SimpleGrantedAuthority object
    .map(role->new SimpleGrantedAuthority(role))
    // read all SimpleGrantedAuthority objects into one Set Collection.
    .collect(Collectors.toSet())

--Core Java -------------------------
package in.nareshit.raghu;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

class MyData {

	String code;

	public MyData(String code) {
		this.code=code;
	}

	@Override
	public String toString() {
		return "MyData [code=" + code + "]";
	}

}

public class Test {

	public static void main(String[] args) {
		List<String> data = Arrays.asList("hi","hello","one");

		Set<MyData> set =
				data.stream()
				.map(a->new MyData(a))
				.collect(Collectors.toSet());

		set.forEach(System.out::println);
	}
}
```

*) UserDetails is interface User
[org.springframework.security.core.userdetails]
   is impl class
=====
(coding)==========
=
pom.xml must have added Spring Security.

> Right click Project > Spring > Add Starters > Spring Security
> Next > pom.xml > Finish

1. Fetch data from db using emailId

--UserRepository.java--

```
	//SQL: select * from user where email=?
	Optional<User> findByUserMail(String userMail);
```

2. Implement UserDetailsService

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

```
import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
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.UsernameNotFoundException
;
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,
UserDetailsService {

	.....(save method logic)

	@Override
	public UserDetails loadUserByUsername(String username)
			throws UsernameNotFoundException
	{
		//fetch user object based on emailId(username)
		Optional<User> opt = repo.findByUserMail(username);

		//if user not exist
		if(!opt.isPresent()) {
			throw new UsernameNotFoundException("Username
not exist!");
		} else {
			//read model class user
			User user = opt.get();

			//read Roles(Set<String>) Convert to Set<GA>
			/*Set<String> roles = user.getUserRoles();

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

			return new
org.springframework.security.core.userdetails
				.User(
					username,
					user.getUserPwd(),
					user.getUserRoles()
						.stream()
						.map(role->new
```

```
SimpleGrantedAuthority(role))

.collect(Collectors.toSet())
                                        );
                }
        }

}
```

---

3. Create Password Encoder object (onetime setup)

```
package in.nareshit.raghu.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
public class AppConfig {

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

4. Modify User password (encode) before save into DB

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

import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
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.UsernameNotFoundException
;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
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,
UserDetailsService {

        @Autowired
        private UserRepository repo;

        @Autowired
```

```
	private BCryptPasswordEncoder encoder;


	public Integer saveUser(User user) {
		//read pwd entered in reg page
		String pwd = user.getUserPwd();

		//encode it
		String encPwd = encoder.encode(pwd);

		//set back to same object
		user.setUserPwd(encPwd);

		user = repo.save(user);
		return user.getId();
	}

	@Override
	public UserDetails loadUserByUsername(String username)
			throws UsernameNotFoundException
	{
		//fetch user object based on emailId(username)
		Optional<User> opt = repo.findByUserMail(username);

		//if user not exist
		if(!opt.isPresent()) {
			throw new UsernameNotFoundException("Username
not exist!");
		} else {
			//read model class user
			User user = opt.get();

			//read Roles(Set<String>) Convert to Set<GA>
			/*Set<String> roles = user.getUserRoles();

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

			return new
org.springframework.security.core.userdetails
					.User(
							username,

user.getUserPwd(),

user.getUserRoles()
							.stream()
							.map(role->new
SimpleGrantedAuthority(role))

.collect(Collectors.toSet())

							);
```

```
                }
        }



}
-----------------------------------------------------------------
LazyInitializationException:
   failed to lazily initialize a collection of role:
   in.nareshit.raghu.model.User.userRoles
```