Login Form with Spring Security 5.x

1. Setup Spring Boot Project with Spring Security

Ensure that you have the necessary dependencies for Spring Security and Thymeleaf (for rendering the login form).

```
<dependencies>
    <!-- Spring Boot Web Dependency -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Spring Security Dependency -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>

    <!-- Thymeleaf Template Engine -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>

    <!-- Spring Boot Starter Test for JUnit -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

---

2. Spring Security Configuration (`SecurityConfig.java`)

Configure Spring Security to handle login, logout actions, and permissions for different users.

```
package com.example.security.config;

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

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.buil
import org.springframework.security.config.annotation.web.builders.HttpSe
import org.springframework.security.config.annotation.web.configuration.E
import org.springframework.security.config.annotation.web.configuration.W
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
                .antMatchers("/", "/login", "/error").permitAll()  // All
                .antMatchers("/admin/**").hasRole("ADMIN")  // Admin sect
                .antMatchers("/user/**").hasRole("USER")   // User sectio
                .anyRequest().authenticated()  // Other requests require
            .and()
            .formLogin()
                .loginPage("/login")  // Custom login page
                .loginProcessingUrl("/login")  // URL to submit the logir
                .defaultSuccessUrl("/home", true)  // Redirect to home or
                .failureUrl("/login?error=true")  // Redirect to login pa
                .permitAll()
            .and()
            .logout()
                .logoutUrl("/logout")  // Logout URL
                .logoutSuccessUrl("/login?logout=true")  // Redirect to l
                .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Ex
        auth.inMemoryAuthentication()
            .withUser("admin").password(passwordEncoder().encode("adminpa
            .and()
            .withUser("user").password(passwordEncoder().encode("userpass
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
```

```
        return new BCryptPasswordEncoder();  // Use BCrypt for hashing pa
    }
}
```

---

3. Controller for Handling Login and Access Pages (`HomeController.java`)

The controller will handle the login page and the redirect after successful login, along with the error message in case of failure.

```
package com.example.security.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class HomeController {

    @GetMapping("/login")
    public String login(@RequestParam(value = "error", required = false)
                        @RequestParam(value = "logout", required = false)
        if (error != null) {
            System.out.println("Login failed due to invalid credentials."
        }
        if (logout != null) {
            System.out.println("Successfully logged out.");
        }
        return "login";  // Return the login page
    }

    @GetMapping("/home")
    public String home() {
        return "home";  // Return the home page after successful login
    }

    @GetMapping("/admin")
    public String admin() {
        return "admin";  // Return the admin page (for users with the "AD
    }

    @GetMapping("/user")
    public String user() {
        return "user";  // Return the user page (for users with the "USER
```

```
    }
}
```

---

4. Login Page Template (`login.html`)

The login page template will be rendered using **Thymeleaf**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="/login" method="POST">
        <div>
            <label for="username">Username:</label>
            <input type="text" name="username" id="username" required>
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" name="password" id="password" required
        </div>
        <button type="submit">Login</button>
        <div>
            <p th:if="${param.error}">Invalid username or password. Pleas
            <p th:if="${param.logout}">You have been logged out successfu
        </div>
    </form>
</body>
</html>
```

---

5. Home Page Template (`home.html`)

The home page for the logged-in users.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>Home</title>
</head>
<body>
    <h2>Welcome to the Home Page</h2>
    <div>
        <a href="/logout">Logout</a>
    </div>
</body>
</html>
```

---

6. Admin Page Template (`admin.html`)

The page for users with the **ADMIN** role.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>Admin Page</title>
</head>
<body>
    <h2>Welcome, Admin!</h2>
    <div>
        <a href="/logout">Logout</a>
    </div>
</body>
</html>
```

---

7. User Page Template (`user.html`)

The page for users with the **USER** role.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>User Page</title>
```

```
</head>
<body>
    <h2>Welcome, User!</h2>
    <div>
        <a href="/logout">Logout</a>
    </div>
</body>
</html>
```

---

8. Running the Application

- Run the application using `mvn spring-boot:run` or `./mvnw spring-boot:run`.
- Visit `http://localhost:8080/login` to access the login page.
- Use the credentials:
    - `admin` / `adminpass` for **Admin** role
    - `user` / `userpass` for **User** role
- Upon successful login, you will be redirected to the appropriate page based on your role.