# Project Name: Handling User Authentication.

## Source Code:

### Jsp:

#### Index.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Home Page</title>

</head>

<body>

    <center>

        <h2 style="text-align: center">Welcome</h2>

        <br> <br>

        <form method="get" action="/loginform">

            <button type="submit">Login</button>

        </form>

        <br> <br>

    </center>

</body>

</html>

#### deniedAccess.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Login Denied</title>

</head>

<body>

    <h2 style="text-align: center">Denied Page</h2>

    <br>

    <br>

    <p style="text-align: center">The username and/or password was

        incorrect.</p>

    <br>

    <br>

    <a style="text-align: center" href="loginform">Return to Log In</a>

    <br>

    <br>

</body>

</html>

#### Loginform.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>User Login</title>

</head>

<body>

    <h2 style="text-align: center">Login Page</h2>

    <div style="color: red;">${error}</div>

    <form style="text-align: center" action="loginform" method='post'>

        <label for="username">Name:</label><br> <input type="text"

            id="username" placeholder="Name Required" name="username" required><br>

        <br> <label for="password">Password:</label><br> <input

            type="text" id="password" placeholder="Password Required"

            name="password" required><br> <br> <input

            type="submit" value="Submit">

    </form>

</body>

</html>

#### Success.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Login Successful</title>

</head>

<body>

    <form style="text-align: center; margin-left: auto; margin-right: auto"

        action="loginform" method="get">

        <label for="username"></label> <input type="hidden" name="username">

        <br> <br> <label for="password"></label> <input

            type="hidden" name="password"> <br> <br>

    </form>

    <h2 style="text-align: center">Success Page</h2>

    <a style="text-align: center" href="loginform">Login</a>

    <h2 style="text-align: center">Welcome!</h2>

    <br>

</body>

</html>

### Controllers:

#### loginController.java:

package com.example.Authentication.controllers;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import com.example.Authentication.entities.User;

import com.example.Authentication.exceptions.UserNotFoundException;

import com.example.Authentication.services.UserService;

@Controller

public class LoginController {

    @Autowired

    UserService userService;

    Logger log = LoggerFactory.getLogger(LoginController.class);

    @GetMapping("/index")

    public String showGreeting(ModelMap map) {

        return "index";

    }

    @GetMapping("/loginform")

    public String showLogin(ModelMap map) {

        return "loginform";

    }

    @PostMapping("/loginform")

    public String submitLogin(ModelMap map, @RequestParam String username, @RequestParam String password){

        User checkUser;

        try {

            checkUser= userService.GetUserByName(username);

            if (userService.verifyPassword(username, password) == false) {

                log.info(username + " - " + password);

                map.addAttribute("error", "Error Wrong password");

                return "deniedAccess";

            }

            else

                return "success";

        } catch (UserNotFoundException unfe) {

            throw unfe;

        }

    }

}

#### userDNEController.java:

package com.example.Authentication.controllers;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import com.example.Authentication.entities.User;

import com.example.Authentication.exceptions.UserNotFoundException;

import com.example.Authentication.services.UserService;

@ControllerAdvice

public class UserDNEController {

    Logger log = LoggerFactory.getLogger(LoginController.class);

    @ExceptionHandler(value = UserNotFoundException.class)

    public String errorLogin(UserNotFoundException dne){

        log.info("error found");

        return "deniedAccess";

    }

}

### Entities:

#### User.java:

package com.example.Authentication.entities;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity // This tells Hibernate to make a table out of this class

@Table(name = "users")

public class User {

    @Id

    @GeneratedValue(strategy=GenerationType.IDENTITY)

    private Integer id;

    private String name;

    private String email;

    private String password;

    public User(String name, String email, String password) {

        this.name = name;

        this.email = email;

        this.password = password;

    }

    public User() {

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

//    //Do I need this?

//    public Integer getId() {

//        return id;

//    }

//

//    //Do I need this?

//    public void setId(Integer id) {

//        this.id = id;

//    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    @Override

    public String toString() {

        return (id.toString() + " " + name + " " + email + " " + password);

    }

}

### Exceptions:

#### userNotFoundException.java:

package com.example.Authentication.exceptions;

public class UserNotFoundException extends RuntimeException {

    private static final long serialVersionUID = 1L;

}

### Repositories:

#### userRepository.java:

package com.example.Authentication.repositories;

import java.util.Optional;

import org.springframework.data.repository.CrudRepository;

import com.example.Authentication.entities.User;

public interface UserRepository extends CrudRepository<User, Integer> {

    public Optional<User> findByName(String name);

}

### Services:

#### userService.java:

package com.example.Authentication.services;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.example.Authentication.entities.User;

import com.example.Authentication.exceptions.UserNotFoundException;

import com.example.Authentication.repositories.UserRepository;

@Service

public class UserService {

    @Autowired

     private UserRepository userRepository;

    public Iterable<User> GetAllUsers()

    {

        return userRepository.findAll();

    }

    public User GetUserByName(String name) {

        Optional<User> foundUser = userRepository.findByName(name);

        if (!foundUser.isPresent()) {

            throw new UserNotFoundException();

        }

        return(foundUser.get());

    }

    /\*

    public User GetUserById(int id) {

        Optional<User> foundUser = userRepository.findById(id);

        //TODO: we need to decide how to handle a "Not Found" condition

        if (!foundUser.isPresent()) {

            throw new UserNotFoundException();

        }

        return(foundUser.get());

    }\*/

    public boolean verifyPassword(String username, String password) {

        boolean verified = false;

        User user = GetUserByName(username);

        if (user.getPassword().equals(password)) {

            verified = true;

        }

        return verified;

    }

    //Do we need this?

    /\*public void UpdateUser(User usertoUpdate) {

        userRepository.save(usertoUpdate);

    }\*/

}

#### authenticationApplication.java:

package com.example.Authentication;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Import;

import com.example.Authentication.controllers.LoginController;

import com.example.Authentication.controllers.UserDNEController;

import com.example.Authentication.entities.User;

import com.example.Authentication.exceptions.UserNotFoundException;

import com.example.Authentication.services.UserService;

@SpringBootApplication

@Import({

    UserNotFoundException.class,

    UserService.class,

    LoginController.class,

    User.class,

    UserDNEController.class

})

public class AuthenticationApplication {

    public static void main(String[] args) {

        SpringApplication.run(AuthenticationApplication.class, args);

    }

}

### Resources:

#### Application.properties:

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/Auther

spring.datasource.username=root

spring.datasource.password=Abc@123

logging.level.org.springframework.web: DEBUG

spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp

server.port=8090

server.error.whitelabel.enabled=false

### test:

#### authenticationApplicationTests.java

package com.example.Authentication;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class AuthenticationApplicationTests {

    @Test

    void contextLoads() {

    }

}

#### authenticationWebTests.java:

package com.example.Authentication;

import com.example.Authentication.controllers.LoginController;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.boot.web.server.LocalServerPort;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.hamcrest.Matchers.containsString;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)

@AutoConfigureMockMvc

public class AuthenticationWebTests {

    @LocalServerPort

    private int port;

    @Autowired

    private LoginController controller;

    @Autowired

    private MockMvc mockMvc;

    @Test

    public void shouldReturnDefaultMessage() throws Exception {

        this.mockMvc.perform(get("/")).andDo(print()).andExpect(status().isOk());

    }

}

#### userEntityTests.java:

package com.example.Authentication;

import com.example.Authentication.entities.User;

import com.example.Authentication.repositories.UserRepository;

import com.example.Authentication.services.UserService;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.Assertions.\*;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class UserEntityTests {

    @Test

    public void WhenSetPassword\_CheckGetPassword() {

        User testUser = new User();

        testUser.setPassword("Samplepassword");

        assertEquals(testUser.getPassword(),"Samplepassword");

    }

    @Test

    public void WhenSetPassword\_CheckPassword() {

        User testUser = new User();

        testUser.setPassword("password");

        String test = testUser.getPassword();

        assertEquals(test, "password");

    }

    @Test

    public void WhenSetName\_CheckGetName() {

        User testUser = new User();

        testUser.setName("Samplename");

        assertEquals(testUser.getName(),"Samplename");

    }

    @Test

    public void WhenSetName\_CheckName() {

        User testUser = new User();

        testUser.setName("name");

        assertEquals(testUser.getName(),"name");

    }

    @Test

    public void WhenSetEmail\_CheckEmail() {

        User testUser = new User();

        testUser.setEmail("sample@test.com");

        String test = testUser.getEmail();

        assertEquals(test, "sample@test.com");

    }

    @Test

    public void WhenSetEmail\_CheckGetEmail() {

        User testUser = new User();

        testUser.setEmail("sample@test.com");

        String test = testUser.getEmail();

        assertEquals(test, "sample@test.com");

    }

}

#### userRepoTests.java:

package com.example.Authentication;

import com.example.Authentication.entities.User;

import com.example.Authentication.repositories.UserRepository;

import com.example.Authentication.services.UserService;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;

import org.springframework.boot.test.context.SpringBootTest;

import org.junit.jupiter.api.Assertions.\*;

import static org.junit.jupiter.api.Assertions.assertEquals;

import java.util.Optional;

@DataJpaTest

public class UserRepoTests {

    @Autowired

    private TestEntityManager entityManager;

    @Autowired

    private UserRepository userRepository;

    @Test

    public void whenFindByName\_thenReturnUser() {

        // given

        User dummyUser = new User();

        dummyUser.setName("Dummy");

        dummyUser.setEmail("test@test.com");

        dummyUser.setPassword("password");

        entityManager.persist(dummyUser);

        entityManager.flush();

        // when

        Optional<User> found = userRepository.findByName(dummyUser.getName());

        User founded = found.get();

        // then

        assertEquals(founded.getName(), dummyUser.getName());

    }

//

//    @Test

//    public void whenGetPassword\_thenReturnBoolean() {

//

//      User dummyUser = new User();

//        dummyUser.setName("Dummy");

//        dummyUser.setEmail("test@test.com");

//        dummyUser.setPassword("password");

//        entityManager.persist(dummyUser);

//        entityManager.flush();

//

//        String test = dummyUser.getPassword();

//

//      assertEquals(test, "password");

//    }

}

#### userServiceTest.java:

package com.example.Authentication;

import static org.junit.jupiter.api.Assertions.assertEquals;

import java.util.List;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;

import com.example.Authentication.entities.User;

import com.example.Authentication.services.UserService;

@DataJpaTest

public class UserServiceTest {

    @Autowired

    private TestEntityManager eM;

    @Autowired

    private UserService us;

    @BeforeEach

    public void bulid() {

        eM.persist(new User("Dummy", "test@test.com", "password"));

        eM.persist(new User("Dummy2", "test2@test.com", "password2"));

        eM.flush();

    }

    @Test

    public void testGetAllUsers() {

        Iterable<User> users = us.GetAllUsers();

        int count = 0;

        for (User user : users) {

            count++;

        }

        assertEquals(count, 2);

    }

    public void testGetUserByName() {

        String name = "Dummy";

        User u = us.GetUserByName(name);

        assertEquals(u.getName(), name);

    }

    @Test

    public void testVerifyPassword() {

        String username = "Dummy";

        String password = "password";

        boolean b = us.verifyPassword(username, password);

        assertEquals(b, true);

    }

}

#### Pom.xml:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.4.3</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.example</groupId>

    <artifactId>Authentication</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>Authentication</name>

    <description>Demo project for Spring Boot</description>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>mysql</groupId>

            <artifactId>mysql-connector-java</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>javax.servlet</groupId>

            <artifactId>jstl</artifactId>

            <version>1.2</version>

        </dependency>

        <dependency>

             <groupId>org.apache.tomcat.embed</groupId>

             <artifactId>tomcat-embed-jasper</artifactId>

             <scope>provided</scope>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

                <configuration>

                    <excludes>

                        <exclude>

                            <groupId>org.projectlombok</groupId>

                            <artifactId>lombok</artifactId>

                        </exclude>

                    </excludes>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>