**Phase 3**

**Lesson 3**

**Handling User Authentication**

**DESCRIPTION**

**Project objective:**

Set up a standalone project to do unit testing of the user authentication class which is used in the main web application. The objective is to create a JUnit class that will test all aspects of the authentication class.

**Background of the problem statement:**

As a part of developing an ecommerce web application, a test-suite is being created to do unit testing of all backend components in the web application. This project will test the user authentication class. This project will be a standalone Java application, since Junit does not directly test servlets or web pages. We are only testing the classes that have the business logic.

**Package name:** **com.Authentication**

**Class name:** **AuthenticationApplication**

package com.Authentication;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Import;

import com.Authentication.controllers.AuthenticationController;

import com.Authentication.entities.User;

import com.Authentication.exceptions.UserNotFoundException;

import com.Authentication.services.AuthenticationService;

@SpringBootApplication

@Import({

AuthenticationController.class,

UserNotFoundException.class,

AuthenticationService.class,

User.class

})

public class AuthenticationApplication {

public static void main(String[] args) {

SpringApplication.run(AuthenticationApplication.class, args);

}

}

**Package name:** **com.Authentication.entities**

**Class name: user**

package com.Authentication.entities;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

@Entity

@Table(name = "user")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@NotNull

private Integer id;

@Column(name = "name")

@NotNull

private String name;

@Column(name = "email")

@NotNull

private String email;

@Column(name = "password")

@NotNull

private String password;

public User() {

super();

}

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

super();

this.name = name;

this.email = email;

this.password = password;

}

public Integer getId() {

return id;

}

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;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "User [id=" + id + ", name=" + name + ", email=" + email + ", password=" + password + "]";

}

}

**Package name:** **com.Authentication.repository**

**Interface name: Authenticationrepository**

package com.Authentication.repositories;

import java.util.Optional;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.Authentication.entities.User;

@Repository

public interface AuthenticationRepository extends CrudRepository<User, Integer> {

public Optional<User> findUserByName(String name);

}

**Package name:** **com.Authentication.exceptions**

**Class name: UserNotFoundException**

**package** com.Authentication.exceptions;

**public** **class** UserNotFoundException **extends** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

}

**Package name:** **com.Authentication.services**

**Class name: AuthenticationServices**

package com.Authentication.services;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.Authentication.entities.User;

import com.Authentication.exceptions.UserNotFoundException;

import com.Authentication.repositories.AuthenticationRepository;

@Service

public class AuthenticationService {

@Autowired

AuthenticationRepository authRepository;

public User GetUserByName(String name) {

Optional<User> found = authRepository.findUserByName(name);

if(found.isPresent()) return found.get();

else throw new UserNotFoundException();

}

public Boolean isValidPassword(String cmp, String actual) {

return ((cmp.equals(actual)) ? true : false);

}

}

**Package name:** **com.Authentication.controller**

**Class name: AuthenticationController**

package com.Authentication.controllers;

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.RequestParam;

import com.Authentication.entities.User;

import com.Authentication.services.AuthenticationService;

@Controller

public class AuthenticationController {

Logger logger = LoggerFactory.getLogger(AuthenticationController.class);

@Autowired

AuthenticationService authenticationService;

@GetMapping("/")

public String showGreeting() {

return "greeting";

}

@GetMapping("/Auth")

public String showLogin() {

return "authenticate";

}

@PostMapping("/Auth")

public String authenticateUser(@RequestParam("username") String username, @RequestParam("password") String pswd) {

User user = authenticationService.GetUserByName(username);

logger.info(user.getName() + " attempted to login with " + user.getPassword());

String path = (authenticationService.isValidPassword(pswd, user.getPassword())) ? "success" : "failure";

logger.info("The path return: " + path);

return path;

}

@GetMapping("/home")

public String returnToHome(ModelMap model) {

model.addAttribute("test", model);

return "greeting";

}

}

**JspCode:**

**Greeting.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>Landing Page</title>

</head>

<h2>!Welcome Page!</h2>

<body>

<h2>Home page</h2> <br/>

Click below to login <br/>

<a href=*"Auth"*>Login</a>

</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>User Page</title>

</head>

<body>

<h1>Login Successful </h1><br/>

<a href=*"/home"*>Home</a>

</body>

</html>

**Failure.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 Failed </title>

</head>

<body>

<h1>Login Failed

</h1><br/>

<a href=*"/home"*>Login again</a>

</body>

</html>

**Authenticate.jsp**

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

pageEncoding=*"ISO-8859-1"*%>

<%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Authentication Page</title>

</head>

<h2>Login Page</h2>

<body>

welcome to the authentication page

<form:form action=*"Auth"* method=*"post"* commandName=*"login"*>

<label for=*"username"*> Username:</label>

<input name=*"username"* id=*"username"* type=*"text"* placeholder=*"Username"* required/>

<label for=*"password"*>Password:</label>

<input name=*"password"* id=*"password"* type=*"password"* placholder=*"Password"* required/>

<input type=*"submit"* name=*"Submit"*/>

</form:form>

</body>

</html>

**Test case code:**

**Package name:** **com.Authentication**

**Class name:** **AuthenticationApplicationTests**

package com.Authentication;

import org.junit.Test;

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

@SpringBootTest

public class AuthenticationApplicationTests {

@Test

public void contextLoads() {

}

}

**Package name:** **com.Authentication**

**Class name:** **AuthenticationCodeTests**

package com.Authentication;

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

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

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

import java.util.Optional;

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.Authentication.entities.User;

import com.Authentication.exceptions.UserNotFoundException;

import com.Authentication.repositories.AuthenticationRepository;

import com.Authentication.services.AuthenticationService;

@DataJpaTest

public class AuthenticationCodeTests {

@Autowired

private TestEntityManager entityManager;

@Autowired

private AuthenticationService authenticationService;

@Autowired

private AuthenticationRepository authenticationRepo;

private User testUser;

@BeforeEach

private void Setup() {

testUser = new User("nischith", "nischith13@gmail.com", "1234");

System.out.println(testUser.toString());

entityManager.persist(testUser);

entityManager.flush();

}

@Test

public void shouldGetUserByName() {

User test = authenticationService.GetUserByName("nischith");

assertEquals(testUser.getName(), test.getName());

}

@Test

public void shouldFindUserByName() throws UserNotFoundException {

Optional<User> temp = authenticationRepo.findUserByName("nischith");

User tempUser = (temp.isPresent()) ? temp.get() : new User();

assertEquals(testUser.getName(), tempUser.getName());

tempUser = new User();

assertFalse(testUser.getName().equals(tempUser.getName()));

}

@Test

public void shouldValidateUser() {

// incorrect username

User input = new User("happy", "BigEars", null);

Optional<User> temp = authenticationRepo.findUserByName(input.getName());

User tempUser = (temp.isPresent()) ? temp.get() : new User();

assertFalse(testUser.getName().equals(input.getName()));

// incorrect password but correct username

input.setName("nischith");

temp = authenticationRepo.findUserByName(input.getName());

tempUser = (temp.isPresent()) ? temp.get() : new User();

assertFalse(authenticationService.isValidPassword(tempUser.getPassword(), input.getPassword()));

//correct username and password

input.setPassword("1234");

assertTrue(authenticationService.isValidPassword(tempUser.getPassword(), input.getPassword()));

}

}

**Package name:** **com.Authentication**

**Class name:** **AuthenticationWebTesting**

package com.Authentication;

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.status;

import org.junit.jupiter.api.Test;

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

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

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

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

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

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

@AutoConfigureMockMvc

public class AuthenticationWebTesting {

@Autowired

private MockMvc mockMvc;

@LocalServerPort

private int port;

@Test

public void shouldGetDefaultMessageFromGreetings() throws Exception{

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

}

@Test

public void shouldGetDefaultMessageFromAuthenticate() throws Exception {

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

}

}