

Handling User Authentication

LoginController.java

```
package com.example.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 org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.Authentication.entities.User;
import com.example.Authentication.repositories.UserRepository;

@RestController

public class LoginController {

    @Autowired
    UserRepository userRepository;

    @GetMapping(value="/")
    public String showIndexPage(ModelMap model){
        return "<html>\n"
            + "<head>\n"
```

```

+ "    <style>\n"
+ "        .center {\n"
+ "            text-align: center;\n"
+ "        }\n"
+ "        \n"
+ "    </style>\n"
+ "</head>\n"
+ "<body style=\"background-color:lightblue;\">\n"
+ "    <div class=\"center\">\n"
+ "        <h1>User Login Page</h1>\n"
+ "        \n"
+ "        <h2 class=\"hello-title\">Welcome</h2>\n"
+ "        \n"
+ "        <a href=\"/allusers\">View all users</a>\n"
+ "        <br><br>\n"
+ "        <form method=\"get\" action=\"/login\">\n"
+ "            <br><h3>Login below:</h3>\n"
+ "            <input type=\"text\" id=\"name\"
name=\"name\" placeholder=\"Name\" required>\n"
+ "            <input type=\"text\" id=\"email\"
name=\"email\" placeholder=\"Email\" required>\n"
+ "            <input type=\"text\" id=\"password\"
name=\"password\" placeholder=\"Password\" required> \n"
+ "            <input type=\"submit\" value=\"Enter\"
/>\n"
+ "        </form>\n"
+ "    </div>\n"
+ "</body>\n"
+ "</html>";
}

```

```
@GetMapping("/login")
```

```
public String showLogin(@RequestParam("name") String name, @RequestParam("email") String email, @RequestParam("password") String password, ModelMap map) {
```

```
    User u = new User(name,email,password);
```

```
    userRepository.save(u);
```

```
    return "<html>\n"
```

```
        + "<head>\n"
```

```
        + "    <style>\n"
```

```
        + "        .center {\n"
```

```
        + "            text-align: center;\n"
```

```
        + "        }\n"
```

```
        + "    \n"
```

```
        + "    </style>\n"
```

```
        + "</head>\n"
```

```
        + "<body style=\"background-color:lightblue;\">\n"
```

```
        + "    <div class=\"center\">\n"
```

```
        + "        <h1>Logged In</h1>\n"
```

```
        + "        \n"
```

```
        + "        <h2 class=\"hello-title\">Successfully Added Your  
Information</h2>\n"
```

```
        + "    </div>\n"
```

```
        + "</body>\n"
```

```
        + "</html>";
```

```
}
```

```
@GetMapping("/allusers")
```

```
    public @ResponseBody String getAllFeedbacks() {
```

```
        // This returns a JSON or XML with the Feedbacks
```

```
        Iterable<User> allUser = userRepository.findAll();
```

```
        return "<html>\n"
```

```
            + "<head>\n"
```

```

        + "        <style>\n"
        + "                .center {\n"
        + "                        text-align: center;\n"
        + "                }\n"
        + "                \n"
        + "        </style>\n"
        + "</head>\n"
        + "<body style=\"background-color:lightblue;\">\n"
        + "        <div class=\"center\">\n"
        + "<h1>Feedback Table</h1>\n"
+ allUser.toString()
        + " </div>\n"
        + "</body>\n"
        + "</html>";
    }

    @PostMapping("/login")
    public String submitLogin(@RequestParam String username, @RequestParam String password){

        //TODO:

        return "Success";

    }
}

```

User.java

```
package com.example.Authentication.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

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

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;

    private String name;

    private String email;

    private String password;

    public User()
    {

    }

    public User(String name, String email, String password) {
        this.name = name;
        this.email = email;
        this.password = password;
    }
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    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;  
}
```

```

@Override

public String toString() {

    return "<br><h3>" + name + " [" + id + "]:" + "</h3><h4>email: " + email +
"</h4><h4>password: " + password + "</h4><br>";

}

}

```

UserNotFoundException.java

```

package com.example.Authentication.exceptions;

public class UserNotFoundException extends RuntimeException {
    private static final long serialVersionUID = 1L;
}

```

UserRepositories.java

```

package com.example.Authentication.repositories;

import org.springframework.data.repository.CrudRepository;

import com.example.Authentication.entities.User;

public interface UserRepository extends CrudRepository<User, Integer> {

    public User findByName(String name);

}

```

UserService.java

```
package com.example.Authentication.services;

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) {
```



```

        User foundUser = userRepository.findByName(name);
        return foundUser;
    }

    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 void UpdateUser(User usertoUpdate) {
        userRepository.save(usertoUpdate);
    }

}

```

AuthnticationApplication.java

```

package com.example.Authentication;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

```

```
@SpringBootApplication
public class AuthenticationApplication {

    public static void main(String[] args) {
        SpringApplication.run(AuthenticationApplication.class, args);
    }

}
```

Application.Properties

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/phase2
spring.datasource.username=root
spring.datasource.password=root
```

```
logging.level.org.springframework.web: DEBUG
```

```
server.port=8090
```

```
server.error.whitelabel.enabled=false
```

Error.jsp

```
<html>
<body>
<h2>Page not found</h2>

</body>
```

```
</html>
```

Greeting.jsp

```
<html>
```

```
<body>
```

```
<h2>Spring Application</h2>
```

```
</body>
```

```
</html>
```

AuthenticationApplicationTests.java

```
package com.example.Authentication;
```

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

```
import static org.junit.jupiter.api.Assertions.assertNotEquals;
```

```
import static org.junit.jupiter.api.Assertions.assertNotNull;
```

```
import org.junit.jupiter.api.Test;
```

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

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

```
import com.example.Authentication.entities.User;
```

```
import com.example.Authentication.services.UserService;
```

```
@SpringBootTest
```

```
class AuthenticationApplicationTests {
```

@Autowired

private UserService userService;

@Test

```
void contextLoads() {  
}
```

@Test

```
void testServiceCall() {  
    Iterable<User> users = userService.GetAllUsers();  
    Integer count = 0;  
  
    for(User u: users)  
        count++;  
  
    assertEquals(count, 0);  
}
```

@Test

```
void countUsers() {  
    Iterable<User> users = userService.GetAllUsers();  
    Integer count = 0;  
  
    for(User u: users)  
        count++;  
  
    assertEquals(count, 4);  
}
```

@Test

```

        void checkAllUsers() {

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

            for(User u: users)

                assertNotNull(u);

        }
    }
}

```

AutheticationTests.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 static org.junit.jupiter.api.Assertions.assertNotNull;

@DataJpaTest

public class AuthenticationTests {

    @Autowired

```

```
private TestEntityManager entityManager;
```

```
@Autowired
```

```
private UserRepository userRepository;
```

```
@Test
```

```
public void returnUserFromName() {
```

```
    User testUser = new User();
```

```
    testUser.setName("newTest");
```

```
    testUser.setEmail("test@email.com");
```

```
    testUser.setPassword("testpw");
```

```
    entityManager.persist(testUser);
```

```
    entityManager.flush();
```

```
    User found = userRepository.findByName(testUser.getName());
```

```
    assertEquals(found.getName(), testUser.getName());
```

```
}
```

```
@Test
```

```
public void passwordNotNull() {
```

```
    Iterable<User> users = userRepository.findAll();
```

```
    for(User u: users)
```

```
        assertNotNull(u.getPassword());
```

```
}
```

```
@Test
```

```
public void nameNotNull() {
```

```

        Iterable<User> users = userRepository.findAll();

        for(User u: users)

            assertNotNull(u.getName());

    }

    @Test

    public void emailNotNull() {

        Iterable<User> users = userRepository.findAll();

        for(User u: users)

            assertNotNull(u.getEmail());

    }

}

```

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());
    }
```

```
    @Test
    public void checkLoginPage() throws Exception {
```



```

        this.mockMvc.perform(get("/login")).andDo(print()).andExpect(status().is4xxClientError());
    }

    @Test
    public void checkUsersPage() throws Exception {
        this.mockMvc.perform(get("/allusers")).andDo(print()).andExpect(status().isOk());
    }
}

```

EntityTests.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;
import static org.junit.jupiter.api.Assertions.assertNotNull;

public class EntityTests {

    @Test
    public void getAndSetPassword() {
        User testUser = new User();

        testUser.setPassword("mypassword");
        assertEquals(testUser.getPassword(), "mypassword");
    }
}

```

@Test

```
public void getAndSetName() {  
    User testUser = new User();  
  
    testUser.setName("joe");  
    assertEquals(testUser.getName(), "joe");  
}
```

@Test

```
public void getAndSetEmail() {  
    User testUser = new User();  
  
    testUser.setEmail("joe@email.com");  
    assertEquals(testUser.getEmail(), "joe@email.com");  
}
```

@Test

```
public void checkToString() {  
    User testUser = new User();  
  
    assertNotNull(testUser.toString());  
}
```

@Test

```
public void checkConstructor() {  
    User testUser = new User("joe", "joe@email.com", "123");  
  
    User checkUser = new User();  
    checkUser.setName("joe");  
    checkUser.setEmail("joe@email.com");  
}
```

```

        checkUser.setPassword("123");

        assertEquals(testUser.getName(), checkUser.getName());
        assertEquals(testUser.getEmail(), checkUser.getEmail());
        assertEquals(testUser.getPassword(), checkUser.getPassword());
    }

    @Test
    public void testDefaultConstructor() {
        User testUser = new User();

        assertNotNull(testUser);
    }
}

```

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>Test</description>

<properties>
    <java.version>1.8</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>

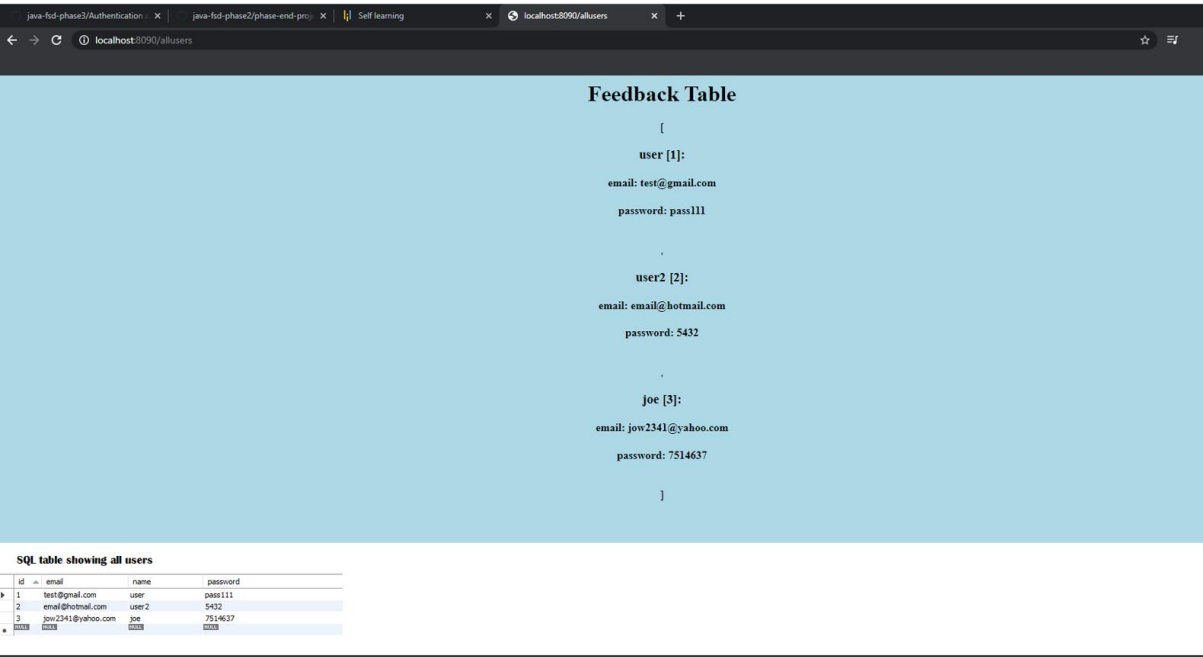
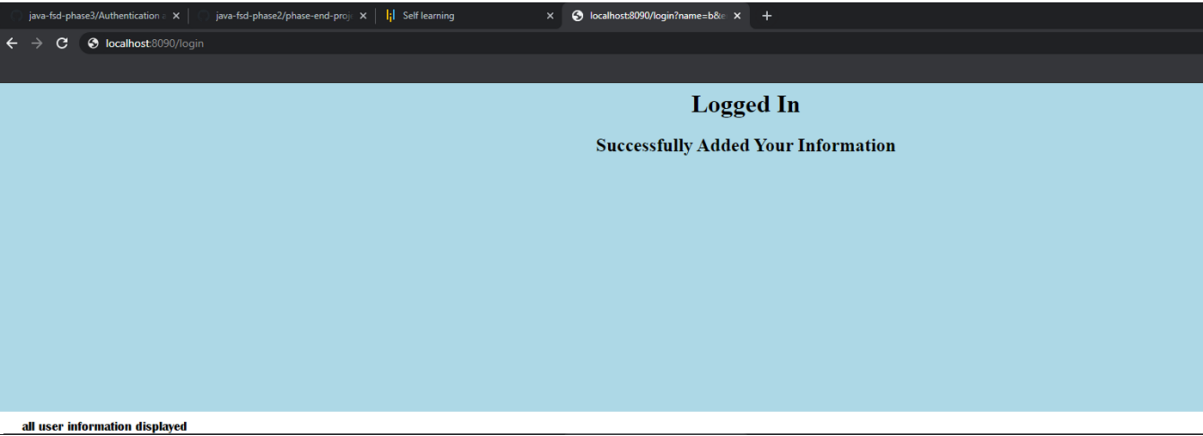
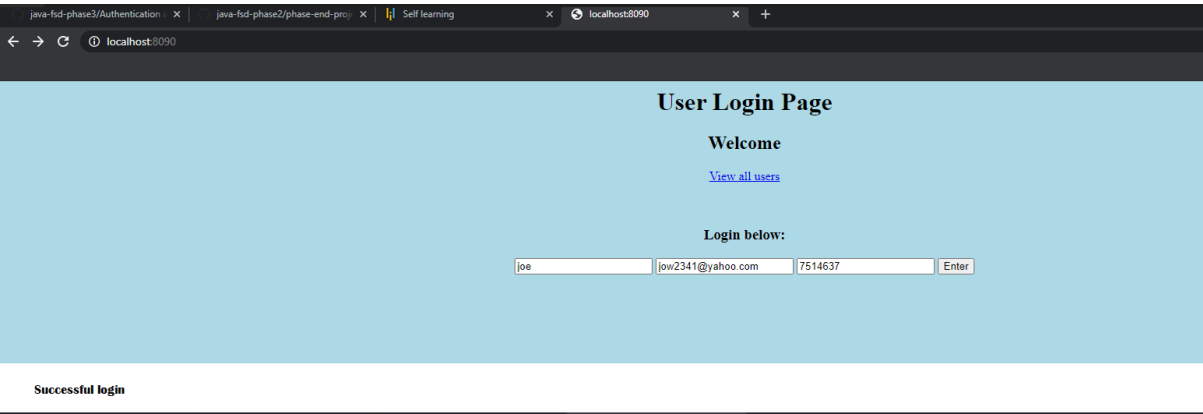
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

```
        <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>
```

OutputScreenshot



Tests:

```

14 @SpringBootTest
15 class AuthenticationApplicationTests {
16
17     @Autowired
18     private UserService userService;
19
20     @Test
21     void contextLoads() {
22     }
23
24     @Test
25     void testServiceCall() {
26         Iterable<User> users = userService.getAllUsers();
27         Integer count = 0;
28
29         for (User u: users)
30             count++;
31
32         assertEquals(count, 0);
33     }
34
35     @Test
36     void countUsers() {
37         Iterable<User> users = userService.getAllUsers();
38         Integer count = 0;
39
40         for (User u: users)
41             count++;
42
43         assertEquals(count, 4);
44     }
45
46     @Test
47     void checkAllUsers() {
48         Iterable<User> users = userService.getAllUsers();
49
50         for (User u: users)
51             assertNotNull(u);
52     }
53 }

```

Finished after 7.88 seconds

Runs: 4/4 Errors: 0 Failures: 0

AuthenticationApplicationTests [Runner: JUnit 5] (0.558 s)

- countUsers() (0.520 s)
- checkAllUsers() (0.016 s)
- contextLoads() (0.006 s)
- testServiceCall() (0.016 s)

Authentication Tests

```

25 @Autowired
26 private UserRepository userRepository;
27
28 @Test
29 public void returnUserFromName() {
30     User testUser = new User();
31     testUser.setName("newTest");
32     testUser.setEmail("test@email.com");
33     testUser.setPassword("testpw");
34     entityManager.persist(testUser);
35     entityManager.flush();
36
37     User found = userRepository.findByName(testUser.getName());
38     assertEquals(found.getName(), testUser.getName());
39 }
40
41 @Test
42 public void passwordNotNull() {
43     Iterable<User> users = userRepository.findAll();
44     for (User u: users)
45         assertNotNull(u.getPassword());
46 }
47
48 @Test
49 public void nameNotNull() {
50     Iterable<User> users = userRepository.findAll();
51     for (User u: users)
52         assertNotNull(u.getName());
53 }
54
55 @Test
56 public void emailNotNull() {
57     Iterable<User> users = userRepository.findAll();
58     for (User u: users)
59         assertNotNull(u.getEmail());
60 }
61 }

```

Finished after 5.462 seconds

Runs: 4/4 Errors: 0 Failures: 0

AuthenticationTests [Runner: JUnit 5] (0.467 s)

- emailNotNull() (0.358 s)
- nameNotNull() (0.013 s)
- passwordNotNull() (0.013 s)
- returnUserFromName() (0.083 s)

Web Tests

```
1 package com.example.Authentication;
2
3 import com.example.Authentication.controllers.LoginController;
4
5 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
6 @AutoConfigureMockMvc
7 public class AuthenticationWebTests {
8
9     @LocalServerPort
10     private int port;
11
12     @Autowired
13     private LoginController controller;
14
15     @Autowired
16     private MockMvc mockMvc;
17
18     @Test
19     public void shouldReturnDefaultMessage() throws Exception {
20         mockMvc.perform(get("/")).andExpect(status().isOk());
21     }
22
23     @Test
24     public void checkLoginPage() throws Exception {
25         mockMvc.perform(get("/login")).andExpect(status().is4xxClientError());
26     }
27
28     @Test
29     public void checkUsersPage() throws Exception {
30         mockMvc.perform(get("/allusers")).andExpect(status().isOk());
31     }
32 }
33
```

com.example.Auth
AuthenticationWeb
port: int
controller: LoginController
mockMvc: MockMvc
shouldReturnDefaultMessage
checkLoginPage
checkUsersPage

Markers Properties Servers Data Source Explorer Snippets Console Progress JUnit

Finished after 8.633 seconds

Runs: 3/3 Errors: 0 Failures: 0

AuthenticationWebTests [Runner: JUnit 5] (0.554 s)

- checkUsersPage() (0.513 s)
- checkLoginPage() (0.021 s)
- shouldReturnDefaultMessage() (0.020 s)

Failure Trace

User Class Tests

```
15 @Test
16 public void getAndSetPassword() {
17     User testUser = new User();
18     testUser.setPassword("mypassword");
19     assertEquals(testUser.getPassword(), "mypassword");
20 }
21
22 @Test
23 public void getAndSetName() {
24     User testUser = new User();
25     testUser.setName("joe");
26     assertEquals(testUser.getName(), "joe");
27 }
28
29 @Test
30 public void getAndSetEmail() {
31     User testUser = new User();
32     testUser.setEmail("joe@email.com");
33     assertEquals(testUser.getEmail(), "joe@email.com");
34 }
35
36 @Test
37 public void checkToString() {
38     User testUser = new User();
39     assertNotNull(testUser.toString());
40 }
41
42 @Test
43 public void checkConstructor() {
44     User testUser = new User("joe", "joe@email.com", "123");
45
46     User checkUser = new User();
47     checkUser.setName("joe");
48     checkUser.setEmail("joe@email.com");
49     checkUser.setPassword("123");
50
51     assertEquals(testUser.getName(), checkUser.getName());
52 }
53
```

com.example.Auth
EntityTests
getAndSetPassword()
getAndSetName()
getAndSetEmail()
checkToString()
checkConstructor()
testDefaultConstructor()

Markers Properties Servers Data Source Explorer Snippets Console Progress JUnit

Finished after 8.633 seconds

Runs: 3/3 Errors: 0 Failures: 0

AuthenticationWebTests [Runner: JUnit 5] (0.554 s)

- checkUsersPage() (0.513 s)
- checkLoginPage() (0.021 s)
- shouldReturnDefaultMessage() (0.020 s)

Failure Trace