JOHN ABBOTT
CEGEP/COLLEGE

Spring Boot MVC Login with @Unit testing

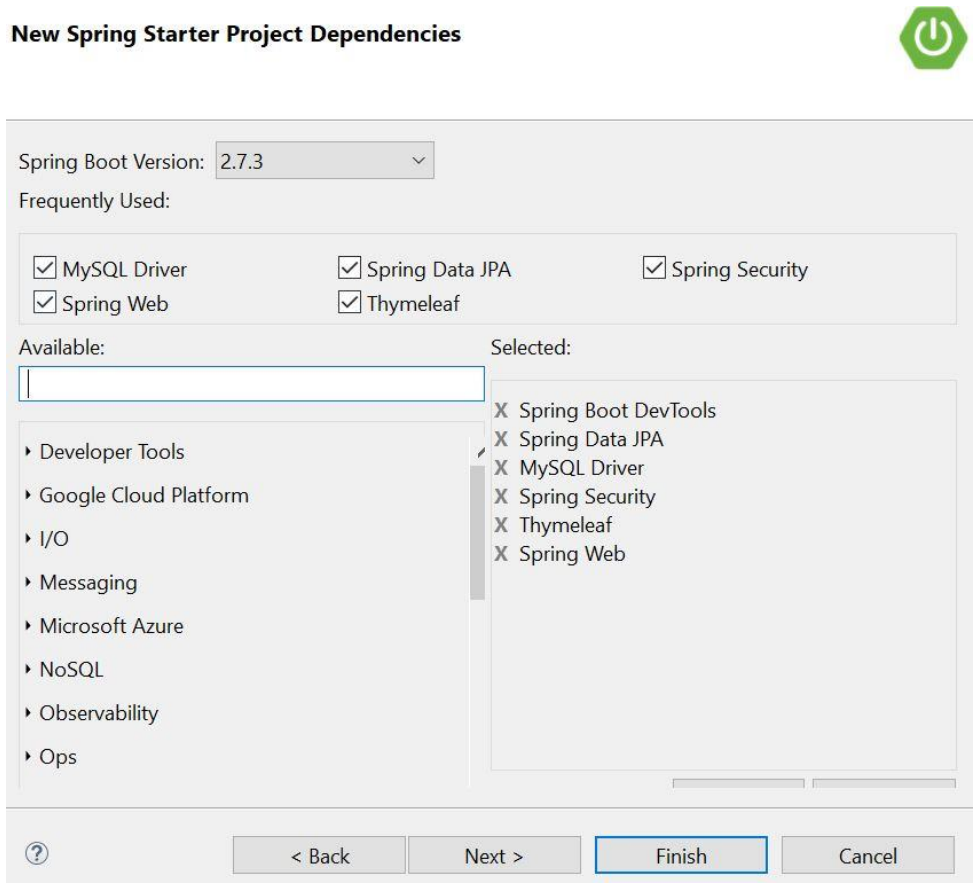
Elie Mambou, Ph.D.

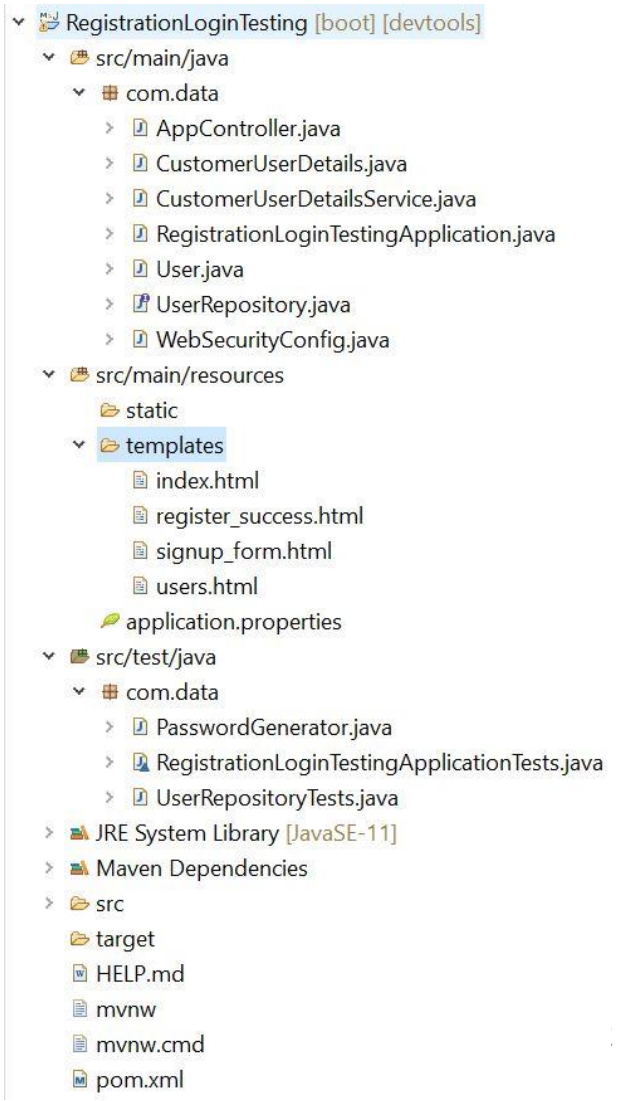
(420-JD5-AB) Programming III

Summer 2022

Create a new Spring boot project

- Create a new project named **RegistrationLoginTesting**
- Add dependencies as below
- Here is the full hierarchy of the web app





Pom.xml file

- Spring Boot DevTools is optional, but it triggers the Spring Boot automatic reload feature to save development time
- Spring Boot uses Junit 5 (Junit Jupiter) by default, and the exclusion means that no support for older versions of JUnit

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
        <exclusion>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Application.properties file

```
spring.datasource.url=jdbc:mysql://localhost:3306/springdatabase?autoReco
nnect true&useSSL=false
spring.datasource.username=root
spring.datasource.password=eliemambou
spring.jpa.hibernate.ddl-auto= create
server.port=8081
```

User.java class under com.data package

JOHN ABBOTT CEGEP/COLLEGE

```java
package com.data;

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

@Entity
@Table(name = "users")
public class User {

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

@Column(nullable = false, unique = true, length = 45)
private String email;

@Column(nullable = false, length = 64)
private String password;

@Column(name = "first_name", nullable = false, length = 20)
private String firstName;

@Column(name = "last_name", nullable = false, length = 20)
private String lastName;

public Long getId() {
return id;
}

public void setId(Long id) {
this.id = id;
}

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;
}

public String getFirstName() {
return firstName;
}

public void setFirstName(String firstName) {
this.firstName = firstName;
}

public String getLastName() {
return lastName;
}

public void setLastName(String lastName) {
this.lastName = lastName;
}
}
```

UserRepository.java interface under com.data

```
package com.data;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long>{

}
```

UserRepositoryTests.java class

```java
package com.data;

import static org.assertj.core.api.Assertions.assertThat;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;
import org.springframework.test.annotation.Rollback;

@DataJpaTest
@AutoConfigureTestDatabase(replace = Replace.NONE)
@Rollback(false)
public class UserRepositoryTests {

@Autowired
private TestEntityManager entityManager;

@Autowired
private UserRepository repo;

@Test
public void testCreateUser() {
User user = new User();
user.setEmail("elie@gmail.com");
user.setPassword("elie");
user.setFirstName("Elie");
user.setLastName("Mambou");

User savedUser = repo.save(user);

User existUser = entityManager.find(User.class, savedUser.getId());

assertThat(user.getEmail()).isEqualTo(existUser.getEmail()); } }
```

This is a basic test class for testing Spring Data JPA repositories. It is configured to work with the actual database (`@AutoConfigureTestDatabase(replace = Replace.NONE)`) and commit the changes (`@Rollback(false)`). `TestEntityManager` is a wrapper of JPA's `EntityManager` so we can use it in test class like a standard `EntityManager`.

Run the test method

- Right-click on testCreateuser method and run it as Junit.

```
Hibernate:

    create table users (
       id bigint not null auto_increment,
        email varchar(45) not null,
        first_name varchar(20) not null,
        last_name varchar(20) not null,
        password varchar(64) not null,
        primary key (id)
    ) engine=InnoDB

Hibernate:

    alter table users
       add constraint UK_6dotkott2kjsp8vw4d0m25fb7 unique (email)

Hibernate:
    insert
    into
        users
        (email, first_name, last_name, password)
    values
        (?, ?, ?, ?)
```

That means Hibernate actually created the table `users` and insert a new row into it (you don't have to create the table manually, right?)

AppController.java

JOHN ABBOTT CEGEP/COLLEGE

```java
package com.data;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AppController {

    @Autowired
    private UserRepository userRepo;

    @GetMapping("")
    public String viewHomePage() {
        return "index";
    }
}
```

Index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="ISO-8859-1">
    <title>Welcome to Spring Boot Login App</title>
    <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/css/bootstrap.min.css" />
    <script type="text/javascript" src="/webjars/jquery/jquery.min.js"></script>
    <script type="text/javascript" src="/webjars/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
    <div class="container text-center">
        <h1> Welcome to Spring Boot Login App </h1>
        <h3><a th:href="@{/users}">List of Users</a></h3>
        <h3><a th:href="@{/register}">Register</a></h3>
        <h3><a th:href="@{/login}">Login</a></h3>
    </div>

</body>
</html>
```

Update pom.xml to use bootstrap and JQuery

JOHN ABBOTT CEGEP/COLLEGE

```
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>3.4.1</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>4.3.1</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>webjars-locator-
core</artifactId>
</dependency>
```

Update AppController.java

JOHN ABBOTT CEGEP/COLLEGE

```
package com.data;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class AppController {

@Autowired
private UserRepository userRepo;

@GetMapping("")
public String viewHomePage() {
return "index";
}

@GetMapping("/register")
public String showRegistrationForm(Model model) {
model.addAttribute("user", new User());

return "signup_form";
}

@PostMapping("/process_register")
public String processRegister(User user) {
BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
String encodedPassword = passwordEncoder.encode(user.getPassword());
user.setPassword(encodedPassword);

userRepo.save(user);

return "register_success";
}

@GetMapping("/users")
public String listUsers(Model model) {
List <User> listUsers = userRepo.findAll();
model.addAttribute("listUsers", listUsers);

return "users";
}
}
```

signup_form.html

JOHN ABBOTT CEGEP/COLLEGE

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="ISO-8859-1">
    <title>Sign Up – Web App</title>
    <link rel="stylesheet" type="text/css"
href="/webjars/bootstrap/css/bootstrap.min.css" />
    <script type="text/javascript" src="/webjars/jquery/jquery.min.js"></script>
    <script type="text/javascript"
src="/webjars/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
    <div class="container text-center">
        <div>
            <h1>User Registration - Sign Up</h1>
        </div>
        <form th:action="@{/process_register}" th:object="${user}"
            method="post" style="max-width: 600px; margin: 0 auto;">
        <div class="m-3">
            <div class="form-group row">
                <label class="col-4 col-form-label">E-mail: </label>
                <div class="col-8">
                    <input type="email" th:field="*{email}" class="form-control"
required />
                </div>
            </div>

            <div class="form-group row">
                <label class="col-4 col-form-label">Password: </label>
                <div class="col-8">
                    <input type="password" th:field="*{password}" class="form-
control"
                            required minlength="6" maxlength="10"/>
                </div>
            </div>

            <div class="form-group row">
                    <label class="col-4 col-form-label">First Name: </label>
                    <div class="col-8">
                        <input type="text" th:field="*{firstName}"
class="form-control"
                                required minlength="2" maxlength="20"/>
                    </div>
                </div>

                <div class="form-group row">
                    <label class="col-4 col-form-label">Last Name: </label>
                    <div class="col-8">
                        <input type="text" th:field="*{lastName}"
class="form-control"
                                required minlength="2" maxlength="20" />
                    </div>
                </div>

                <div>
                    <button type="submit" class="btn btn-primary">Sign
Up</button>
                </div>
            </div>
            </form>
        </div>
</body>
</html>
```

register_success.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="ISO-8859-1">
    <title>Registration Success</title>
    <link rel="stylesheet" type="text/css"
href="/webjars/bootstrap/css/bootstrap.min.css" />
</head>
<body>
    <div class="container text-center">
        <h3>You have signed up successfully!</h3>
        <h4><a th:href="@{/login}">Click here to
Login</a></h4>
    </div>

</body>
</html>
```

CustomUserDetails.java

JOHN ABBOTT CEGEP/COLLEGE

```java
package com.data;

import java.util.Collection;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetails implements UserDetails {

    private User user;

    public CustomUserDetails(User user) {
        this.user = user;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return null;
    }

    @Override
    public String getPassword() {
        return user.getPassword();
    }

    @Override
    public String getUsername() {
        return user.getEmail();
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

    public String getFullName() {
        return user.getFirstName() + " " + user.getLastName();
    }

}
```

CustomUserDetailsService.java

JOHN ABBOTT CEGEP/COLLEGE

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

public class CustomUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository userRepo;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepo.findByEmail(username);
        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }
        return new CustomUserDetails(user);
    }

}
```

UserRepository.java interface

```
public interface UserRepository extends JpaRepository<User, Long> {
    @Query("SELECT u FROM User u WHERE u.email = ?1")
    public User findByEmail(String email);
}
```

UPDATE THE application.properties file

```
spring.jpa.hibernate.ddl-auto= none
```

Web security for authentication (login)

JOHN ABBOTT CEGEP/COLLEGE

```
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private DataSource dataSource;

    @Bean
    public UserDetailsService userDetailsService() {
        return new CustomUserDetailsService();
    }

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

```
    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(passwordEncoder());

        return authProvider;
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(authenticationProvider());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/users").authenticated()
            .anyRequest().permitAll()
            .and()
            .formLogin()
                .usernameParameter("email")
                .defaultSuccessUrl("/users")
                .permitAll()
            .and()
            .logout().logoutSuccessUrl("/").permitAll();
    }


}
```

users.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="ISO-8859-1">
    <title>List Users</title>
    <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/css/bootstrap.min.css" />
    <script type="text/javascript" src="/webjars/jquery/jquery.min.js"></script>
    <script type="text/javascript" src="/webjars/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container text-center">
    <div>
        <form th:action="@{/logout}" method="post">
            <p>
                Welcome <b>[[${#request.userPrincipal.principal.fullName}]]</b>
            </p>
            <input type="submit" value="Sign Out" />
        </form>
    </div>
    <div>
        <h1>List of Users</h1>
    </div>

    <div>
        <table class="table table-striped table-bordered">
            <thead class="thead-dark">
                <tr>
                    <th>User ID</th>
                    <th>E-mail</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                </tr>
            </thead>
            <tbody>
                    <tr th:each="user: ${listUsers}">
                        <td th:text="${user.id}">User ID</td>
                        <td th:text="${user.email}">E-mail</td>
                        <td th:text="${user.firstName}">First Name</td>
                        <td th:text="${user.lastName}">Last Name</td>
                    </tr>
                </tbody>
            </table>
        </div>
</div>
</body>
</html>
```

Running web application

Welcome **Elie Mambou**

Sign Out

List of Users

User ID	E-mail	First Name	Last Name
1	elie@gmail.com	Elie	Mambou
5	Elie2@gmail.com	Elie2	Mambou2

User Registration - Sign Up

E-mail: elie@gmail.com

Password: ••••••••••

First Name: Elie

Last Name: Mambou

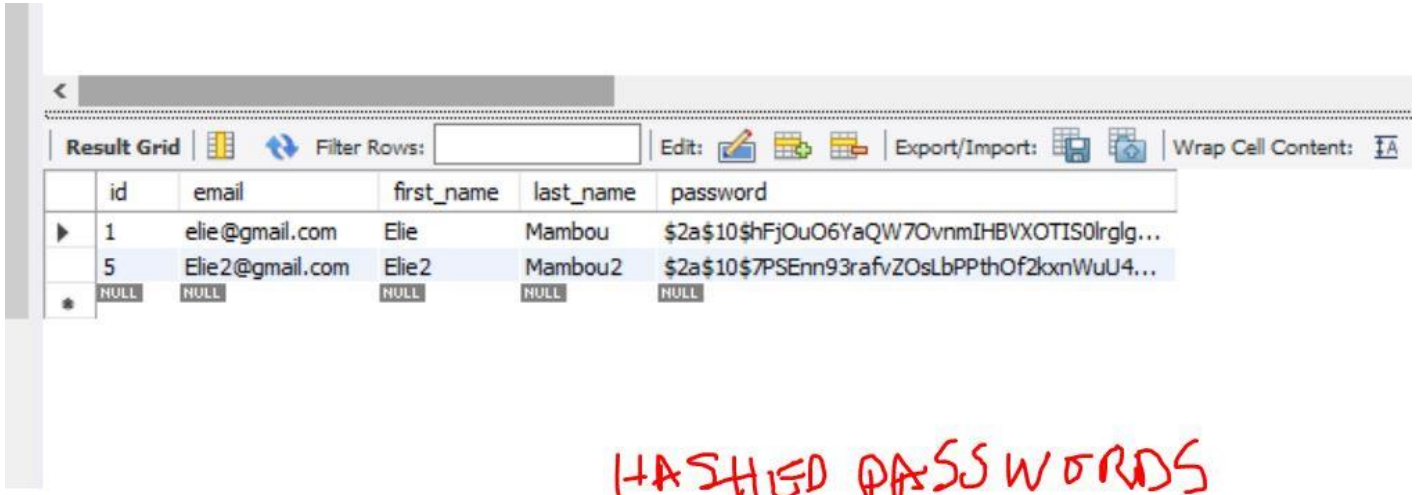
Sign Up

Please sign in

elie@gmail.com

Password

Sign in



HASHED PASSWORDS

Q & A