**SECURITY**

package iuh.fit.se.config;

import org.springframework.boot.CommandLineRunner;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import iuh.fit.se.entities.Candidate;

import iuh.fit.se.entities.Experience;

import iuh.fit.se.entities.Roles;

import iuh.fit.se.repository.CandidateRepository;

import java.time.LocalDate;

import java.util.List;

@Configuration

public class DataSeeder {

@Bean

public CommandLineRunner seedData(CandidateRepository candidateRepository) {

return args -> {

Candidate candidate1 = new Candidate();

candidate1.setEmail("user1@example.com");

candidate1.setFullName("Username");

candidate1.setPhone("123456789");

Experience exp1 = new Experience();

exp1.setCompanyName("ABC Corp");

exp1.setFromDate(LocalDate.of(2015, 1, 1));

exp1.setToDate(LocalDate.of(2020, 1, 1));

exp1.setRole(Roles.STAFF);

exp1.setWorkDescription("STAFF software applications");

exp1.setCandidate(candidate1);

candidate1.setExperiences(List.of(exp1));

candidateRepository.save(candidate1);

};

}

}

package iuh.fit.se.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsManager() {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password(passwordEncoder().encode("user"))

.roles("USER")

.build(),

User.withUsername("admin")

.password(passwordEncoder().encode("admin"))

.roles("ADMIN")

.build()

);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

SecurityFilterChain filterChain(HttpSecurity httpSecurity) throws Exception {

httpSecurity.csrf(csrf -> {

csrf.disable();

})

// Require authentication

.authorizeHttpRequests(request -> request

.requestMatchers("/report2").hasRole("ADMIN")

.requestMatchers("/report1", "/candidates").hasAnyRole("USER")

.anyRequest()

.permitAll()

)

// Specify the login page and permit all access to it

.formLogin(form -> {

form.loginProcessingUrl("/login");

form.defaultSuccessUrl("/");

form.permitAll();

})

// Specify the logout request matcher and permit all access to it

.logout(logout -> logout.permitAll())

.httpBasic(Customizer.withDefaults());

return httpSecurity.build();

}

// @Bean

// public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

// http.authorizeHttpRequests(requests -> requests

// .requestMatchers("/report2").hasRole("ADMIN")

// .requestMatchers("/report1", "/candidates").hasRole("USER")

// .anyRequest().authenticated()) // Mọi yêu cầu khác đều cần xác thực

// .formLogin(login -> login.permitAll())

// .logout(logout -> logout.permitAll());

// return http.build();

// }

}

**- ENTITES:**

@Entity

@Table(name = "candidate")

public class Candidate {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "can\_id")

private long id;

@Column(name = "full\_name")

private String fullName;

@Column(name = "email")

private String email;

@Column(name = "phone")

private String phone;

@OneToMany(mappedBy = "candidate", cascade = CascadeType.ALL, orphanRemoval = true)

private List<Experience> experiences;

public Candidate() {

super();

// TODO Auto-generated constructor stub

}

public Candidate(String fullName, String email, String phone, List<Experience> experiences) {

super();

this.fullName = fullName;

this.email = email;

this.phone = phone;

this.experiences = experiences;

}

Get/set/toString

}

@Entity

@Table(name = "experience")

public class Experience {

// column type is BIGINT(20) in MySQL

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "exp\_id")

private long id;

// column type is DATE in MySQL

@Column(name = "from\_date")

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate fromDate;

// column type is DATE in MySQL

@Column(name = "to\_date")

@DateTimeFormat(pattern = "yyyy-MM-dd")

private LocalDate toDate;

@Column(name = "work\_desc")

private String workDescription;

@Column(name = "company")

private String companyName;

// column type is tinyint(4) in MySQL

@Column(name = "role")

@Enumerated(EnumType.STRING)

private Roles role;

@ManyToOne

@JoinColumn(name = "can\_id")

private Candidate candidate;

public Experience() {

super();

// TODO Auto-generated constructor stub

}

**-CONTROLLER**

package iuh.fit.se.controller;

import java.time.temporal.ChronoUnit;

import java.util.List;

import java.util.stream.Collectors;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import iuh.fit.se.entities.Candidate;

import iuh.fit.se.repository.CandidateRepository;

@Controller

public class CandidateController {

private final CandidateRepository candidateRepository;

public CandidateController(CandidateRepository candidateRepository) {

this.candidateRepository = candidateRepository;

}

@GetMapping("/")

public String index() {

return "index";

}

@GetMapping("/candidates")

public String listCandidates(Model model) {

model.addAttribute("candidates", candidateRepository.findAll());

return "candidates";

}

@GetMapping("/candidates/{id}")

public String candidateDetails(@PathVariable Long id, Model model) {

model.addAttribute("candidate", candidateRepository.findById(id).orElse(null));

return "cand\_details";

}

@GetMapping("/report1")

public String report1(@RequestParam(required = false) String role, Model model) {

if (role == null || role.isEmpty()) {

model.addAttribute("error", "Vui lòng nhập vai trò.");

model.addAttribute("candidates", List.of()); // Trả về danh sách trống

return "report1"; // Hiển thị trang với form nhập

}

List<Candidate> candidates = candidateRepository.findAll().stream()

.filter(candidate -> candidate.getExperiences().stream()

.anyMatch(exp -> exp.getRole().toString().equalsIgnoreCase(role)))

.collect(Collectors.toList());

model.addAttribute("candidates", candidates);

return "report1";

}

@GetMapping("/report2")

public String report2(Model model) {

List<Candidate> candidates = candidateRepository.findAll().stream()

.filter(candidate -> candidate.getExperiences().stream()

.mapToLong(exp -> ChronoUnit.YEARS.between(exp.getFromDate(), exp.getToDate()))

.sum() >= 5)

.collect(Collectors.toList());

model.addAttribute("candidates", candidates);

return "report2";

}

}

-HTML

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Candidates List</title>

</head>

<body>

<h1>Candidates</h1>

<table border="1">

<thead>

<tr>

<th>Email</th>

<th>Full Name</th>

<th>Phone</th>

<th>Details</th>

</tr>

</thead>

<tbody>

<tr th:each="candidate : ${candidates}">

<td th:text="${candidate.email}"></td>

<td th:text="${candidate.fullName}"></td>

<td th:text="${candidate.phone}"></td>

<td><a th:href="@{'/candidates/' + ${candidate.id}}">Details</a></td>

</tr>

</tbody>

</table>

<form th:action="@{/logout}" method="post">

<input type="submit" value="Sign Out"/>

</form>

</body>

</html>

-- report

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Report 1</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

h1 {

background-color: #333;

color: #fff;

margin: 0;

padding: 10px;

text-align: center;

}

form {

margin-top: 20px;

text-align: center;

}

label {

margin-right: 10px;

}

input {

margin-right: 10px;

}

#btn {

background-color: red;

width: 90px;

height: 30px;

color: white;

font-weight: bold;

}

ul {

text-align: center;

text-decoration: none;

}

li {

list-style-type: none;

}

</style>

</head>

<body>

<h1>Report 1: Candidates by Role</h1>

<form action="/report1" method="get">

<label for="role">Enter Role:</label>

<input type="text" id="role" name="role" required>

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

</form>

<ul>

<li th:each="candidate : ${candidates}">

<span th:text="${candidate.id}"></span> -

<span th:text="${candidate.fullName}"></span> -

<span th:text="${candidate.email}"></span> -

<span th:text="${candidate.phone}"></span>

</li>

</ul>

<form th:action="@{/logout}" method="post">

<input id="btn" type="submit" value="Sign Out"/>

</form>

</body>

</html>