**MAIN PACKAGE: PACKAGE COM.IMARTICUS.EMPLOYEE;**

package com.imarticus.Employee;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeApplication.class, args);

}

}

**CONTROLLER CLASSES:**

**EMPLOYEECONTROLLER:**

package com.imarticus.Employee.controller;

import java.util.List;

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

import org.springframework.data.domain.Page;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

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

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

import com.imarticus.Employee.Model.Employee;

import com.imarticus.Employee.service.EmployeeService;

@Controller

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

// display list of employees

@GetMapping("/")

public String viewHomePage(Model model) {

return findPaginated(1, "firstName", "asc", model);

}

@GetMapping("/showNewEmployeeForm")

public String showNewEmployeeForm(Model model) {

// create model attribute to bind form data

Employee employee = new Employee();

model.addAttribute("employee", employee);

return "new\_employee";

}

@PostMapping("/saveEmployee")

public String saveEmployee(@ModelAttribute("employee") Employee employee) {

// save employee to database

employeeService.saveEmployee(employee);

return "redirect:/";

}

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

public String showFormForUpdate(@PathVariable ( value = "id") long id, Model model) {

// get employee from the service

Employee employee = employeeService.getEmployeeById(id);

// set employee as a model attribute to pre-populate the form

model.addAttribute("employee", employee);

return "update\_employee";

}

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

public String deleteEmployee(@PathVariable (value = "id") long id) {

// call delete employee method

this.employeeService.deleteEmployeeById(id);

return "redirect:/";

}

@GetMapping("/page/{pageNo}")

public String findPaginated(@PathVariable (value = "pageNo") int pageNo,

@RequestParam("sortField") String sortField,

@RequestParam("sortDir") String sortDir,

Model model) {

int pageSize = 5;

Page<Employee> page = employeeService.findPaginated(pageNo, pageSize, sortField, sortDir);

List<Employee> listEmployees = page.getContent();

model.addAttribute("currentPage", pageNo);

model.addAttribute("totalPages", page.getTotalPages());

model.addAttribute("totalItems", page.getTotalElements());

model.addAttribute("sortField", sortField);

model.addAttribute("sortDir", sortDir);

model.addAttribute("reverseSortDir", sortDir.equals("asc") ? "desc" : "asc");

model.addAttribute("listEmployees", listEmployees);

return "index";

}

}

**MAINCONTROLLER:**

package com.imarticus.Employee.controller;

import org.springframework.stereotype.Controller;

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

@Controller

public class MainController {

@GetMapping("/login")

public String login() {

return "login";

}

}

**USERREGISTERCONTROLLER:**

package com.imarticus.Employee.controller;

import org.springframework.stereotype.Controller;

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

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

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

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

import com.imarticus.Employee.dto.UserRegistrationDto;

import com.imarticus.Employee.service.UserService;

@Controller

@RequestMapping("/registration")

public class UserRegistrationController {

private UserService userService;

public UserRegistrationController(UserService userService) {

super();

this.userService = userService;

}

@ModelAttribute("user")

public UserRegistrationDto userRegistrationDto() {

return new UserRegistrationDto();

}

@GetMapping

public String showRegistrationForm() {

return "registration";

}

@PostMapping

public String registerUserAccount(@ModelAttribute("user") UserRegistrationDto registrationDto) {

userService.save(registrationDto);

return "redirect:/registration?success";

}

}

**DTO PACKAGE:**

**USER REGISTRATIONDTO:**

**package** com.imarticus.Employee.dto;

**public** **class** UserRegistrationDto {

**private** String firstName;

**private** String lastName;

**private** String email;

**private** String password;

**public** UserRegistrationDto(){

}

**public** UserRegistrationDto(String firstName, String lastName, String email, String password) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.email = email;

**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;

}

**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;

}

}

**PACKAGE MODEL:**

**EMPLOYEE CLASS:**

package com.imarticus.Employee.Model;

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 = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "email")

private String email;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

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;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**USER CLASS:**

package com.imarticus.Employee.Model;

import java.util.Collection;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint;

import javax.persistence.JoinColumn;

@Entity

@Table(name = "user", uniqueConstraints = @UniqueConstraint(columnNames = "email"))

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

private String email;

private String password;

@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

@JoinTable(

name = "users\_roles",

joinColumns = @JoinColumn(

name = "user\_id", referencedColumnName = "id"),

inverseJoinColumns = @JoinColumn(

name = "role\_id", referencedColumnName = "id"))

private Collection<Role> roles;

public User() {

}

public User(String firstName, String lastName, String email, String password, Collection<Role> roles) {

super();

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.password = password;

this.roles = roles;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

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;

}

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 Collection<Role> getRoles() {

return roles;

}

public void setRoles(Collection<Role> roles) {

this.roles = roles;

}

}

**ROLE CLASS:**

package com.imarticus.Employee.Model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "role")

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

public Role() {

}

public Role(String name) {

super();

this.name = name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**PACKAGE REPOSITORY:**

**EMPLOYEEREPOSITORY:**

package com.imarticus.Employee.repository;

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

import org.springframework.stereotype.Repository;

import com.imarticus.Employee.Model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**USERREPOSITORY:**

package com.imarticus.Employee.repository;

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

import org.springframework.stereotype.Repository;

import com.imarticus.Employee.Model.User;

@Repository

public interface UserRepository extends JpaRepository<User, Long>{

User findByEmail(String email);

}

**PACKAGE SERVICE:**

**EMPLOYEESERVICE:**

import java.util.List;

import org.springframework.data.domain.Page;

import com.imarticus.Employee.Model.Employee;

public interface EmployeeService {

List<Employee> getAllEmployees();

void saveEmployee(Employee employee);

Employee getEmployeeById(long id);

void deleteEmployeeById(long id);

Page<Employee> findPaginated(int pageNo, int pageSize, String sortField, String sortDirection);

}

**EMPLOYEESERVICEIMPL:**

package com.imarticus.Employee.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.stereotype.Service;

import com.imarticus.Employee.Model.Employee;

import com.imarticus.Employee.repository.EmployeeRepository;

@Service

public class EmployeeServiceImpl implements EmployeeService{

@Autowired

private EmployeeRepository employeeRepository;

@Override

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@Override

public void saveEmployee(Employee employee) {

this.employeeRepository.save(employee);

}

@Override

public Employee getEmployeeById(long id) {

Optional<Employee> optional = employeeRepository.findById(id);

Employee employee = null;

if (optional.isPresent()) {

employee = optional.get();

} else {

throw new RuntimeException(" Employee not found for id :: " + id);

}

return employee;

}

@Override

public void deleteEmployeeById(long id) {

this.employeeRepository.deleteById(id);

}

@Override

public Page<Employee> findPaginated(int pageNo, int pageSize, String sortField, String sortDirection) {

Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortField).ascending() :

Sort.by(sortField).descending();

Pageable pageable = PageRequest.of(pageNo - 1, pageSize, sort);

return this.employeeRepository.findAll(pageable);

}

}

**USERSERVICE:**

**package** com.imarticus.Employee.service;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** com.imarticus.Employee.Model.User;

**import** com.imarticus.Employee.dto.UserRegistrationDto;

**public** **interface** UserService **extends** UserDetailsService{

User save(UserRegistrationDto registrationDto);

}

**USERSERVICEIMPL:**

package com.imarticus.Employee.service;

import java.util.Arrays;

import java.util.Collection;

import java.util.stream.Collectors;

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

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

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

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

import org.springframework.stereotype.Service;

import com.imarticus.Employee.Model.Role;

import com.imarticus.Employee.Model.User;

import com.imarticus.Employee.dto.UserRegistrationDto;

import com.imarticus.Employee.repository.UserRepository;

@Service

public class UserServiceImpl implements UserService{

private UserRepository userRepository;

@Autowired

private BCryptPasswordEncoder passwordEncoder;

public UserServiceImpl(UserRepository userRepository) {

super();

this.userRepository = userRepository;

}

@Override

public User save(UserRegistrationDto registrationDto) {

User user = new User(registrationDto.getFirstName(),

registrationDto.getLastName(), registrationDto.getEmail(),

passwordEncoder.encode(registrationDto.getPassword()), Arrays.asList(new Role("ROLE\_USER")));

return userRepository.save(user);

}

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userRepository.findByEmail(username);

if(user == null) {

throw new UsernameNotFoundException("Invalid username or password.");

}

return new org.springframework.security.core.userdetails.User(user.getEmail(), user.getPassword(), mapRolesToAuthorities(user.getRoles()));

}

private Collection<? extends GrantedAuthority> mapRolesToAuthorities(Collection<Role> roles){

return roles.stream().map(role -> new SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());

}

}

**CONFIG PACKAGE:**

**SECURITYCONFIG CLASS:**

**package** com.imarticus.Employee.config;

**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.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.web.util.matcher.AntPathRequestMatcher;

**import** com.imarticus.Employee.service.UserService;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter {

@Autowired

**private** UserService userService;

@Bean

**public** BCryptPasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** DaoAuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider auth = **new** DaoAuthenticationProvider();

auth.setUserDetailsService(userService);

auth.setPasswordEncoder(passwordEncoder());

**return** auth;

}

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

auth.authenticationProvider(authenticationProvider());

}

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.authorizeRequests().antMatchers(

"/registration\*\*",

"/js/\*\*",

"/css/\*\*",

"/img/\*\*").permitAll()

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.invalidateHttpSession(**true**)

.clearAuthentication(**true**)

.logoutRequestMatcher(**new** AntPathRequestMatcher("/logout"))

.logoutSuccessUrl("/login?logout")

.permitAll();

}

}

**HTML FILES :**

**INDEX.HTML**

<!DOCTYPE html>

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

<head>

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

<title>Employee Management System</title>

<link rel=*"stylesheet"*

href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*

integrity=*"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"*

crossorigin=*"anonymous"*>

</head>

<body>

<!-- create navigation bar ( header) -->

<nav class=*"navbar navbar-inverse navbar-fixed-top"*>

<div class=*"container"*>

<div class=*"navbar-header"*>

<button type=*"button"* class=*"navbar-toggle collapsed"*

data-toggle=*"collapse"* data-target=*"#navbar"* aria-expanded=*"false"*

aria-controls=*"navbar"*>

<span class=*"sr-only"*>Toggle navigation</span> <span

class=*"icon-bar"*></span> <span class=*"icon-bar"*></span> <span

class=*"icon-bar"*></span>

</button>

<a class=*"navbar-brand"* href=*"#"* th:href=*"@{/}"*>Employee Management System</a>

</div>

<div id=*"navbar"* class=*"collapse navbar-collapse"*>

<ul class=*"nav navbar-nav"*>

<li sec:authorize=*"isAuthenticated()"*><a th:href=*"@{/logout}"*>Logout</a></li>

</ul>

</div>

</div>

</nav>

<br>

<br>

<div class=*"container my-2"*>

<h1>Employees List</h1>

<a th:href = *"@{/showNewEmployeeForm}"* class=*"btn btn-primary btn-sm mb-3"*> Add Employee </a>

<table border=*"1"* class = *"table table-striped table-responsive-md"*>

<thead>

<tr>

<th>

<a th:href=*"@{'/page/' + ${currentPage} + '?sortField=firstName&sortDir=' + ${reverseSortDir}}"*>

Employee First Name</a>

</th>

<th>

<a th:href=*"@{'/page/' + ${currentPage} + '?sortField=lastName&sortDir=' + ${reverseSortDir}}"*>

Employee Last Name</a>

</th>

<th>

<a th:href=*"@{'/page/' + ${currentPage} + '?sortField=email&sortDir=' + ${reverseSortDir}}"*>

Employee Email</a>

</th>

<th> Actions </th>

</tr>

</thead>

<tbody>

<tr th:each=*"employee : ${listEmployees}"*>

<td th:text=*"${employee.firstName}"*></td>

<td th:text=*"${employee.lastName}"*></td>

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

<td> <a th:href=*"@{/showFormForUpdate/{id}(id=${employee.id})}"* class=*"btn btn-primary"*>Update</a>

<a th:href=*"@{/deleteEmployee/{id}(id=${employee.id})}"* class=*"btn btn-danger"*>Delete</a>

</td>

</tr>

</tbody>

</table>

<div th:if = *"${totalPages > 1}"*>

<div class = *"row col-sm-10"*>

<div class = *"col-sm-2"*>

Total Rows: [[${totalItems}]]

</div>

<div class = *"col-sm-1"*>

<span th:each=*"i: ${#numbers.sequence(1, totalPages)}"*>

<a th:if=*"${currentPage != i}"* th:href=*"@{'/page/' + ${i}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}"*>[[${i}]]</a>

<span th:unless=*"${currentPage != i}"*>[[${i}]]</span> &nbsp; &nbsp;

</span>

</div>

<div class = *"col-sm-1"*>

<a th:if=*"${currentPage < totalPages}"* th:href=*"@{'/page/' + ${currentPage + 1}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}"*>Next</a>

<span th:unless=*"${currentPage < totalPages}"*>Next</span>

</div>

<div class=*"col-sm-1"*>

<a th:if=*"${currentPage < totalPages}"* th:href=*"@{'/page/' + ${totalPages}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}"*>Last</a>

<span th:unless=*"${currentPage < totalPages}"*>Last</span>

</div>

</div>

</div>

</div>

</body>

</html>

**LOGIN.HTML**

<!DOCTYPE html>

<html>

<head>

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

<title>Registration and Login App</title>

<link rel=*"stylesheet"*

href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*

integrity=*"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"*

crossorigin=*"anonymous"*>

</head>

<body>

<!-- create navigation bar ( header) -->

<nav class=*"navbar navbar-inverse navbar-fixed-top"*>

<div class=*"container"*>

<div class=*"navbar-header"*>

<button type=*"button"* class=*"navbar-toggle collapsed"*

data-toggle=*"collapse"* data-target=*"#navbar"* aria-expanded=*"false"*

aria-controls=*"navbar"*>

<span class=*"sr-only"*>Toggle navigation</span> <span

class=*"icon-bar"*></span> <span class=*"icon-bar"*></span> <span

class=*"icon-bar"*></span>

</button>

<a class=*"navbar-brand"* href=*"#"* th:href=*"@{/}"*>Employee Management System</a>

</div>

</div>

</nav>

<br>

<br>

<div class = *"container"*>

<div class = *"row"*>

<div class = *"col-md-6 col-md-offset-3"*>

<h1> Sign-in </h1>

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

<!-- error message -->

<div th:if=*"${param.error}"*>

<div class=*"alert alert-danger"*>Invalid username or

password.</div>

</div>

<!-- logout message -->

<div th:if=*"${param.logout}"*>

<div class=*"alert alert-info"*>You have been logged out.</div>

</div>

<div class = *"form-group"*>

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

<input type=*"text"* class = *"form-control"* id =*"username"* name = *"username"*

placeholder=*"Enter Email ID"* autofocus=*"autofocus"*>

</div>

<div class=*"form-group"*>

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

id=*"password"* name=*"password"* class=*"form-control"*

placeholder=*"Enter Password"* />

</div>

<div class=*"form-group"*>

<div class=*"row"*>

<div class=*"col-sm-6 col-sm-offset-3"*>

<input type=*"submit"* name=*"login-submit"* id=*"login-submit"*

class=*"form-control btn btn-primary"* value=*"Log In"* />

</div>

</div>

</div>

</form>

<div class=*"form-group"*>

<span>New user? <a href=*"/"* th:href=*"@{/registration}"*>Register

here</a></span>

</div>

</div>

</div>

</div>

</body>

</html>

**NEWEMPLOYEE.HTML:**

<!DOCTYPE html>

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

<head>

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

<title>Employee Management System</title>

<link rel=*"stylesheet"*

href=*"https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"*

integrity=*"sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"*

crossorigin=*"anonymous"*>

</head>

<body>

<div class=*"container"*>

<h1>Employee Management System</h1>

<hr>

<h2>Save Employee</h2>

<form action=*"#"* th:action=*"@{/saveEmployee}"* th:object=*"${employee}"*

method=*"POST"*>

<input type=*"text"* th:field=*"\*{firstName}"*

placeholder=*"Employee First Name"* class=*"form-control mb-4 col-4"*>

<input type=*"text"* th:field=*"\*{lastName}"*

placeholder=*"Employee Last Name"* class=*"form-control mb-4 col-4"*>

<input type=*"text"* th:field=*"\*{email}"*

placeholder=*"Employee Email"* class=*"form-control mb-4 col-4"*>

<button type=*"submit"* class=*"btn btn-info col-2"*> Save Employee</button>

</form>

<hr>

<a th:href = *"@{/}"*> Back to Employee List</a>

</div>

</body>

</html>

**REGISTRATION.HTML:**

<!DOCTYPE html>

<html>

<head>

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

<title>Registration</title>

<link rel=*"stylesheet"*

href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*

integrity=*"sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"*

crossorigin=*"anonymous"*>

</head>

<body>

<!-- create navigation bar ( header) -->

<nav class=*"navbar navbar-inverse navbar-fixed-top"*>

<div class=*"container"*>

<div class=*"navbar-header"*>

<button type=*"button"* class=*"navbar-toggle collapsed"*

data-toggle=*"collapse"* data-target=*"#navbar"* aria-expanded=*"false"*

aria-controls=*"navbar"*>

<span class=*"sr-only"*>Toggle navigation</span> <span

class=*"icon-bar"*></span> <span class=*"icon-bar"*></span> <span

class=*"icon-bar"*></span>

</button>

<a class=*"navbar-brand"* href=*"#"* th:href=*"@{/}"*>Employee Management System</a>

</div>

</div>

</nav>

<br>

<br>

<!-- Create HTML registration form -->

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-md-6 col-md-offset-3"*>

<!-- success message -->

<div th:if=*"${param.success}"*>

<div class=*"alert alert-info"*>You've successfully registered

to our awesome app!</div>

</div>

<h1>Registration</h1>

<form th:action=*"@{/registration}"* method=*"post"* th:object=*"${user}"*>

<div class=*"form-group"*>

<label class=*"control-label"* for=*"firstName"*> First Name </label>

<input id=*"firstName"* class=*"form-control"* th:field=*"\*{firstName}"*

required autofocus=*"autofocus"* />

</div>

<div class=*"form-group"*>

<label class=*"control-label"* for=*"lastName"*> Last Name </label> <input

id=*"lastName"* class=*"form-control"* th:field=*"\*{lastName}"*

required autofocus=*"autofocus"* />

</div>

<div class=*"form-group"*>

<label class=*"control-label"* for=*"email"*> Email </label> <input

id=*"email"* class=*"form-control"* th:field=*"\*{email}"* required

autofocus=*"autofocus"* />

</div>

<div class=*"form-group"*>

<label class=*"control-label"* for=*"password"*> Password </label> <input

id=*"password"* class=*"form-control"* type=*"password"*

th:field=*"\*{password}"* required autofocus=*"autofocus"* />

</div>

<div class=*"form-group"*>

<button type=*"submit"* class=*"btn btn-success"*>Register</button>

<span>Already registered? <a href=*"/"* th:href=*"@{/login}"*>Login

here</a></span>

</div>

</form>

</div>

</div>

</div>

</body>

</html>

**UDPADEEMPLOYEE.HTML:**

<!DOCTYPE html>

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

<head>

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

<title>Employee Management System</title>

<link rel=*"stylesheet"*

href=*"https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"*>

</head>

<body>

<div class=*"container"*>

<h1>Employee Management System</h1>

<hr>

<h2>Update Employee</h2>

<form action=*"#"* th:action=*"@{/saveEmployee}"* th:object=*"${employee}"*

method=*"POST"*>

<!-- Add hidden form field to handle update -->

<input type=*"hidden"* th:field=*"\*{id}"* />

<input type=*"text"* th:field=*"\*{firstName}"* class=*"form-control mb-4 col-4"*>

<input type=*"text"* th:field=*"\*{lastName}"* class=*"form-control mb-4 col-4"*>

<input type=*"text"* th:field=*"\*{email}"* class=*"form-control mb-4 col-4"*>

<button type=*"submit"* class=*"btn btn-info col-2"*> Update Employee</button>

</form>

<hr>

<a th:href = *"@{/}"*> Back to Employee List</a>

</div>

</body>

</html>

……………………………………………………………………………………………………………………………………………………………………………………………………………………….