Step 1: Create a new Spring Boot project

Use Spring Initializr to create a new project with the following dependencies:

- Spring Web

- Spring Data JPA

- Thymeleaf

- H2 Database (for simplicity)

Step 2: Create the Employee entity

Create a new Java class Employee.java in the com.example.employee package:

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private String department;

// Getters and setters

}

Step 3: Create the EmployeeRepository interface

Create a new Java interface EmployeeRepository.java in the com.example.employee pack…

[10:55 PM, 4/14/2024] Prabhakar Nishad: Sure, I can provide you with a basic example of how to create an Employee Management System using Spring Boot and Thymeleaf. Here's a step-by-step guide:

Step 1: Create a new Spring Boot project

Use Spring Initializr ((link unavailable)) to create a new project with the following dependencies:

- Spring Web

- Spring Data JPA

- Thymeleaf

- H2 Database (for simplicity)

Step 2: Create the Employee entity

Create a new Java class Employee.java in the com.example.employee package:

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private String department;

// Getters and setters

}

Step 3: Create the EmployeeRepository interface

Create a new Java interface EmployeeRepository.java in the com.example.employee package:

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

Step 4: Create the EmployeeService class

Create a new Java class EmployeeService.java in the com.example.employee package:

@Service

public class EmployeeService {

private final EmployeeRepository employeeRepository;

public EmployeeService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public Employee saveEmployee(Employee employee) {

return employeeRepository.save(employee);

}

public Employee getEmployeeById(Long id) {

return employeeRepository.findById(id).orElseThrow();

}

public void deleteEmployee(Long id) {

employeeRepository.deleteById(id);

}

}

Step 5: Create the EmployeeController class

Create a new Java class EmployeeController.java in the com.example.employee package:

@Controller

@RequestMapping("/employees")

public class EmployeeController {

private final EmployeeService employeeService;

public EmployeeController(EmployeeService employeeService) {

this.employeeService = employeeService;

}

@GetMapping

public String getAllEmployees(Model model) {

model.addAttribute("employees", employeeService.getAllEmployees());

return "employees";

}

@GetMapping("/new")

public String newEmployee(Model model) {

model.addAttribute("employee", new Employee());

return "employee-form";

}

@PostMapping

public String saveEmployee(@ModelAttribute Employee employee) {

employeeService.saveEmployee(employee);

return "redirect:/employees";

}

@GetMapping("/{id}")

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

model.addAttribute("employee", employeeService.getEmployeeById(id));

return "employee-details";

}

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

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

model.addAttribute("employee", employeeService.getEmployeeById(id));

return "employee-form";

}

@PostMapping("/{id}")

public String updateEmployee(@PathVariable Long id, @ModelAttribute Employee employee) {

employeeService.saveEmployee(employee);

return "redirect:/employees";

}

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

public String deleteEmployee(@PathVariable Long id) {

employeeService.deleteEmployee(id);

return "redirect:/employees";

}

}

Step 6: Create the Thymeleaf templates

Create the following HTML templates in the src/main/resources/templates directory:

- employees.html: displays a list of all employees

- employee-form.html: a form to create or edit an employee

- employee-details.html: displays the details of an employee