Employee.java:  
package com.cognizant.springlearn;

import com.fasterxml.jackson.annotation.JsonFormat;

import javax.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

@NotNull

@Min(0)

private Double salary;

@NotNull

private Boolean permanent;

@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

@NotNull

private Department department;

private List<Skill> skillList;

// Getters, Setters, toString...

}

Department.java:

package com.cognizant.springlearn;

import javax.validation.constraints.\*;

public class Department {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

// Getters, Setters...

}

Skill.java:

package com.cognizant.springlearn;

import javax.validation.constraints.\*;

public class Skill {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

// Getters, Setters...

}

EmployeeNotFoundException.java:

package com.cognizant.springlearn.service.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(value = HttpStatus.NOT\_FOUND, reason = "Employee Not Found")

public class EmployeeNotFoundException extends Exception {

public EmployeeNotFoundException(String message) {

super(message);

}

}

EmployeeDao.java:

package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.Employee;

import com.cognizant.springlearn.service.exception.EmployeeNotFoundException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.ArrayList;

import java.util.List;

@Repository

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST;

public EmployeeDao() {

ApplicationContext context = new ClassPathXmlApplicationContext("employee.xml");

EMPLOYEE\_LIST = (ArrayList<Employee>) context.getBean("employeeList");

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

public void updateEmployee(Employee employee) throws EmployeeNotFoundException {

boolean found = false;

for (int i = 0; i < EMPLOYEE\_LIST.size(); i++) {

if (EMPLOYEE\_LIST.get(i).getId().equals(employee.getId())) {

EMPLOYEE\_LIST.set(i, employee);

found = true;

break;

}

}

if (!found) {

throw new EmployeeNotFoundException("Employee with ID " + employee.getId() + " not found.");

}

}

}

**EmployeeService.java:**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Employee;

import com.cognizant.springlearn.dao.EmployeeDao;

import com.cognizant.springlearn.service.exception.EmployeeNotFoundException;

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

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

@Transactional

public void updateEmployee(Employee employee) throws EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

}

EmployeeController.java:

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Employee;

import com.cognizant.springlearn.service.EmployeeService;

import com.cognizant.springlearn.service.exception.EmployeeNotFoundException;

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

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@PutMapping

public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

}

GlobalExceptionHandler.java:

@Override

protected ResponseEntity<Object> handleHttpMessageNotReadable(

HttpMessageNotReadableException ex, HttpHeaders headers, HttpStatus status, WebRequest request) {

Map<String, Object> body = new LinkedHashMap<>();

body.put("timestamp", new Date());

body.put("status", status.value());

body.put("error", "Bad Request");

if (ex.getCause() instanceof InvalidFormatException) {

InvalidFormatException ife = (InvalidFormatException) ex.getCause();

for (InvalidFormatException.Reference ref : ife.getPath()) {

body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");

}

} else {

body.put("message", "Malformed JSON request");

}

return new ResponseEntity<>(body, headers, status);

}