**Country.java**

package com.cognizant.springlearn.model;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

@NotNull

@Size(min = 2, max = 2, message = "Country code should be 2 characters")

private String code;

@NotNull

@Size(min = 1, max = 50)

private String name;

// Getters and setters

}

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import jakarta.validation.Valid;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

@RequestMapping("/countries")

public class CountryController {

private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);

@PostMapping

public Country addCountry(@RequestBody @Valid Country country) {

LOGGER.info("Start");

LOGGER.debug("Country: {}", country);

return country;

}

}

package com.cognizant.springlearn.exception;

import com.fasterxml.jackson.databind.exc.InvalidFormatException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.\*;

import org.springframework.web.bind.MethodArgumentNotValidException;

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

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.\*;

import java.util.stream.Collectors;

@ControllerAdvice

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

private static final Logger LOGGER = LoggerFactory.getLogger(GlobalExceptionHandler.class);

@Override

protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

LOGGER.info("Validation failed");

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

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

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

List<String> errors = ex.getBindingResult().getFieldErrors().stream()

.map(x -> x.getDefaultMessage())

.collect(Collectors.toList());

body.put("errors", errors);

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

}

@Override

protected ResponseEntity<Object> handleHttpMessageNotReadable(

org.springframework.http.converter.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() + "'");

}

}

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

}

}

**Employee.java**

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

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;

@NotNull

private Date dateOfBirth;

@Valid

private Department department;

@Valid

private List<Skill> skills;

// Getters and Setters

}

**Department.java**

public class Department {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

}

**Skill.java**

public class Skill {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

}

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PutMapping

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

employeeService.updateEmployee(employee);

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

public void updateEmployee(Employee employee) throws EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

public void deleteEmployee(int id) throws EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

}

**EmployeeDao.java**

@Repository

public class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST = new ArrayList<>();

public void updateEmployee(Employee emp) throws EmployeeNotFoundException {

boolean updated = false;

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

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

EMPLOYEE\_LIST.set(i, emp);

updated = true;

break;

}

}

if (!updated) throw new EmployeeNotFoundException("Employee not found");

}

public void deleteEmployee(int id) throws EmployeeNotFoundException {

boolean removed = EMPLOYEE\_LIST.removeIf(e -> e.getId() == id);

if (!removed) throw new EmployeeNotFoundException("Employee not found");

}

}