Spring Rest

**Country Controller Implementation**

@RestController

@RequestMapping("/countries")

public class CountryController {

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

// GET all countries

@GetMapping

public List<Country> getAllCountries() {

// Implementation to return all countries

}

// GET specific country by code

@GetMapping("/{code}")

public Country getCountryByCode(@PathVariable String code) {

// Implementation to return specific country

}

// POST to create a new country

@PostMapping

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

LOGGER.info("Start");

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

LOGGER.info("End");

return country;

}

// PUT to update a country

@PutMapping

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

LOGGER.info("Start");

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

LOGGER.info("End");

return country;

}

// DELETE a country by code

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

LOGGER.info("Start");

LOGGER.debug("Deleting country with code: {}", code);

LOGGER.info("End");

}

}

**Country Model with Validation**

public class Country {

@NotNull

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

private String code;

@NotBlank

@Size(min=2, max=50, message="Country name should be between 2 and 50 characters")

private String name;

// Getters and setters

}

**Global Exception Handler**

@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("Start");

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);

LOGGER.info("End");

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

}

@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) {

final Throwable cause = ex.getCause() == null ? ex : ex.getCause();

for (InvalidFormatException.Reference reference : ((InvalidFormatException) cause).getPath()) {

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

}

}

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

}

@ExceptionHandler(EmployeeNotFoundException.class)

public ResponseEntity<Object> handleEmployeeNotFound(

EmployeeNotFoundException ex, WebRequest request) {

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

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

body.put("status", HttpStatus.NOT\_FOUND.value());

body.put("error", "Not Found");

body.put("message", ex.getMessage());

return new ResponseEntity<>(body, HttpStatus.NOT\_FOUND);

}

}

**Employee Controller Implementation**

@RestController

@RequestMapping("/employees")

public class EmployeeController {

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

@Autowired

private EmployeeService employeeService;

// PUT to update an employee

@PutMapping

public ResponseEntity<Object> updateEmployee(@RequestBody @Valid Employee employee)

throws EmployeeNotFoundException {

LOGGER.info("Start");

Employee updatedEmployee = employeeService.updateEmployee(employee);

LOGGER.info("End");

return ResponseEntity.ok(updatedEmployee);

}

// DELETE an employee by id

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable int id)

throws EmployeeNotFoundException {

LOGGER.info("Start");

employeeService.deleteEmployee(id);

LOGGER.info("End");

return ResponseEntity.noContent().build();

}

}

**Employee Model with Validation**

public class Employee {

@NotNull(message = "Employee ID must not be null")

private Integer id;

@NotBlank(message = "Employee name must not be blank")

@Size(min = 1, max = 30, message = "Employee name must be between 1 and 30 characters")

private String name;

@NotNull(message = "Salary must not be null")

@Min(value = 0, message = "Salary must be 0 or above")

private Double salary;

@NotNull(message = "Permanent status must not be null")

private Boolean permanent;

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

private Date dateOfBirth;

@Valid

private Department department;

@Valid

private List<Skill> skills;

// Getters and setters

}

**Department Model with Validation**

public class Department {

@NotNull(message = "Department ID must not be null")

private Integer id;

@NotBlank(message = "Department name must not be blank")

@Size(min = 1, max = 30, message = "Department name must be between 1 and 30 characters")

private String name;

// Getters and setters

}

**Skill Model with Validation**

public class Skill {

@NotNull(message = "Skill ID must not be null")

private Integer id;

@NotBlank(message = "Skill name must not be blank")

@Size(min = 1, max = 30, message = "Skill name must be between 1 and 30 characters")

private String name;

// Getters and setters

}

**Testing with cURL Commands**

curl -i -H 'Content-Type: application/json' -X POST -s -d '{"code":"IN","name":"India"}' <http://localhost:8090/countries>

curl -i -H 'Content-Type: application/json' -X POST -s -d '{"code":"IND","name":"India"}' <http://localhost:8090/countries>

curl -i -H 'Content-Type: application/json' -X PUT -s -d '{

"id": 1,

"name": "John Doe",

"salary": 50000.0,

"permanent": true,

"dateOfBirth": "01/01/1990",

"department": {

"id": 1,

"name": "Engineering"

},

"skills": [

{

"id": 1,

"name": "Java"

},

{

"id": 2,

"name": "Spring"

}

]

}' <http://localhost:8090/employees>

curl -i -X DELETE -s http://localhost:8090/employees/1