**1. HTTP Method Types & Naming Conventions**

| **HTTP Method** | **Usage** | **Annotation** | **Example URL** |
| --- | --- | --- | --- |
| GET | Fetch resources | @GetMapping | /countries, /countries/IN |
| POST | Create a new resource | @PostMapping | /countries |
| PUT | Update an existing resource | @PutMapping | /countries |
| DELETE | Delete a specific resource | @DeleteMapping | /countries/IN |

**Guidelines:**

* Use plural nouns for collections: /countries
* Use hyphens for multi-word resources: /menu-item
* Avoid using verbs in URLs.

**2. POST Method - Add Country**

**Controller:**

@PostMapping

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

LOGGER.info("Start");

return country;

}

**Sample cURL:**

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

This will return:

{"code":"IN","name":"India"}

If a property is mistyped (e.g., nae instead of name), Spring will map it as null.

**3. Country Validation**

**In Country.java:**

@NotNull

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

private String code;

**Manual Validation (Temporary – not recommended for reuse):**

ValidatorFactory factory = Validation.buildDefaultValidatorFactory();

Validator validator = factory.getValidator();

Set<ConstraintViolation<Country>> violations = validator.validate(country);

if (!violations.isEmpty()) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, "Country code should be 2 characters");

}

**4. Global Exception Handling for Validation**

**Create GlobalExceptionHandler.java:**

@ControllerAdvice

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

@Override

protected ResponseEntity<Object> handleMethodArgumentNotValid(

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

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

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

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

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(FieldError::getDefaultMessage)

.collect(Collectors.toList());

body.put("errors", errors);

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

}

}

**Change in Controller:**

@PostMapping

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

return country;

}

**5. PUT Method - Update Employee**

**Add Validations**

**In Employee.java:**

@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;

**In Department.java and Skill.java**: Add similar validations on id and name.

**Controller Method**

@PutMapping("/employees")

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

employeeService.updateEmployee(employee);

}

**6. Handle Invalid Format Errors**

**In 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 cause = (InvalidFormatException) ex.getCause();

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

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

}

}

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

}

**7. DELETE Method - Delete Employee**

**Controller:**

@DeleteMapping("/employees/{id}")

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

employeeService.deleteEmployee(id);

}

**Service:**

public void deleteEmployee(int id) throws EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

**DAO:**

public void deleteEmployee(int id) throws EmployeeNotFoundException {

Employee emp = EMPLOYEE\_LIST.stream()

.filter(e -> e.getId() == id)

.findFirst()

.orElseThrow(() -> new EmployeeNotFoundException("Employee not found"));

EMPLOYEE\_LIST.remove(emp);

}