**1. CountryController.java**

java

Copy code

@RestController

@RequestMapping("/countries")

public class CountryController {

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

// POST - Add Country

@PostMapping

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

LOGGER.info("Start");

return country;

}

// PUT - Update Country

@PutMapping

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

LOGGER.info("Update Start");

return country;

}

// DELETE - Delete Country by Code

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

LOGGER.info("Deleted Country with code: " + code);

}

// GET - Get All Countries

@GetMapping

public List<Country> getAllCountries() {

return List.of(new Country("IN", "India"), new Country("US", "United States"));

}

// GET - Get Country by Code

@GetMapping("/{code}")

public Country getCountryByCode(@PathVariable String code) {

return new Country(code, "DummyCountry");

}

}

**2. Country.java**

java

Copy code

public class Country {

@NotNull

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

private String code;

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

@Size(min = 1, message = "Name must not be blank")

private String name;

// Constructors

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**3. GlobalExceptionHandler.java**

java

Copy code

@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) {

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

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

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

}

}

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

}

}