

Lab5-cloud

Lab Program - 5

Write a sample REST App to Validate the REST API POST & PUT request.

- Design a custom response with appropriate validation errors to the caller

OR

Create a Spring Boot Application using Maven Plugin.

Write a REST Controller API using Spring Annotations to validate the user input for

- Product details using POST, PUT methods.

- Using Postman invoke the REST Controller to demonstrate end to end working

Step 1: Setting Up the Project on Spring Initializer:

Navigate to 'start.spring.io', choose Java, Spring Boot 3.2.2, and configure the project details. Hit generate and download the project zip. Import it into Eclipse.

Example configuration:

Project: Maven

Language: Java

Spring Boot: 3.2.2 (SNAPSHOT)

Group: com.lab5

Artifact: my_lab5

Name: my_lab5

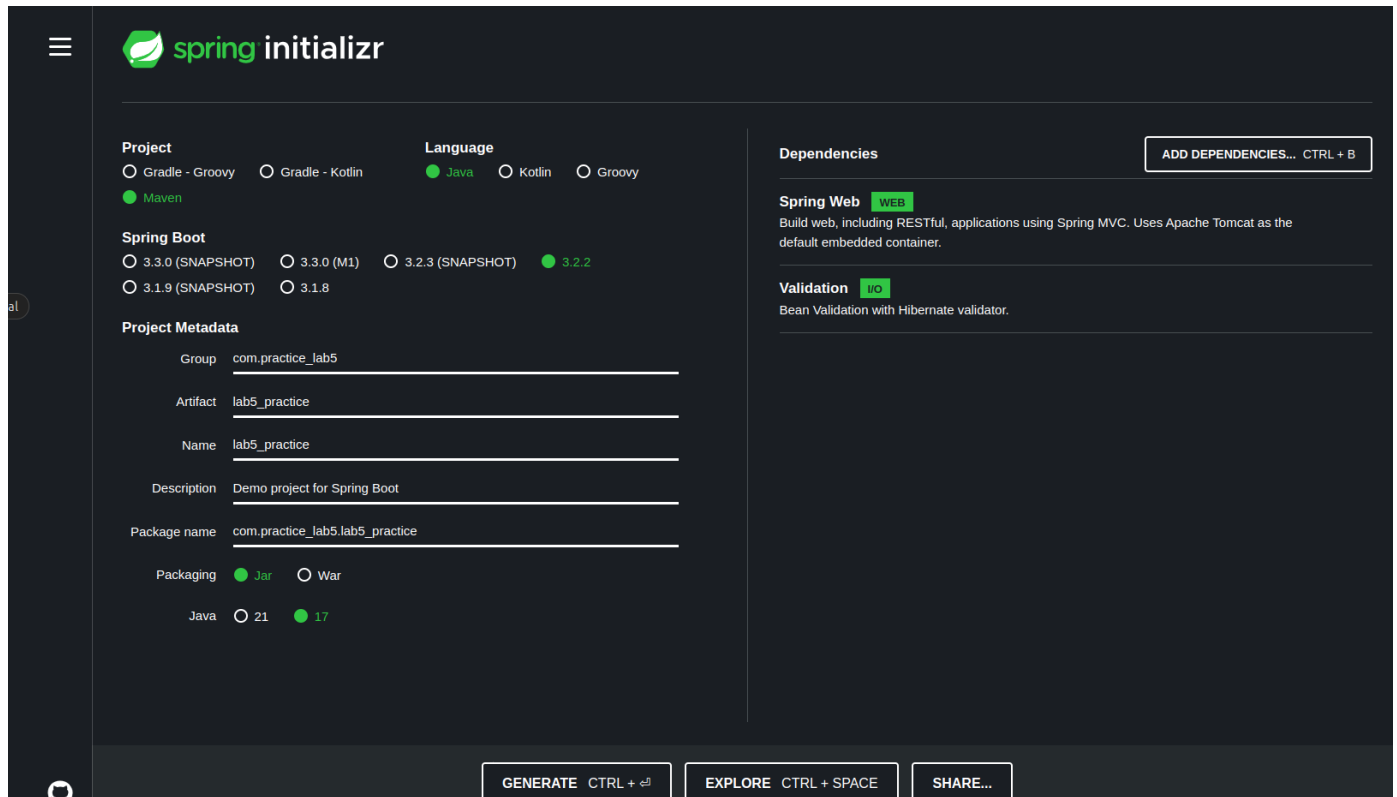
Description: Lab Program 5

Package name: com.lab5.my_lab5

Packaging: Jar

Java: 17

And add dependency - Spring Web (Web)and Validation (I/O)(Bean Validation and Hibernate Validator)



The image shows the Spring Initializr web interface. On the left, there's a sidebar with a hamburger menu icon. The main area is divided into sections: 'Project' with radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven (selected); 'Language' with radio buttons for Java (selected), Kotlin, and Groovy; 'Spring Boot' with radio buttons for 3.3.0 (SNAPSHOT), 3.3.0 (M1), 3.2.3 (SNAPSHOT), 3.2.2 (selected), and 3.1.9 (SNAPSHOT), 3.1.8; 'Project Metadata' with input fields for Group (com.practice_lab5), Artifact (lab5_practice), Name (lab5_practice), Description (Demo project for Spring Boot), and Package name (com.practice_lab5.lab5_practice); and 'Packaging' with radio buttons for Jar (selected) and War, and 'Java' with radio buttons for 21 and 17 (selected). On the right, the 'Dependencies' section has a button 'ADD DEPENDENCIES... CTRL + B' and two selected dependencies: 'Spring Web' (WEB) and 'Validation' (I/O). At the bottom, there are buttons for 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

Step 2: Create Product.java under src/main/java

```
package com.lab5.my_lab5;

import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;

public class Product {

    @NotNull(message="id is required")
    @Min(1)
    @Max(100)
    private Long id;

    @NotBlank(message = "Name is required")
    private String name;

    @Min(1)
    @NotNull(message = "Price is required")
    private Double price;

    public Long getId() {
        return id;
    }
}
```

```
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Double getPrice() {
    return price;
}

public void setPrice(Double price) {
    this.price = price;
}
}
```

Step 3: Create ProductController.java under src/main/java

```
package com.lab5.my_lab5;

import java.util.ArrayList;
import java.util.List;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

```

import jakarta.validation.Valid;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    private final List<Product> products = new ArrayList<>();

    /**
     * OR
     *
     * @PostMapping public ResponseEntity<?> addProduct(@Valid @RequestBody
Product
     * product, BindingResult result) {
     *
     * List<String> displayErrors = new ArrayList<String>();
     *
     * if (result.hasErrors()) { List<FieldError> errors =
result.getFieldErrors();
     * for(FieldError err:errors) { displayErrors.add(err.getField() + ": "
+
     * err.getDefaultMessage()); System.out.println(displayErrors); } return
     * ResponseEntity.badRequest().body(displayErrors); }
products.add(product);
     * return ResponseEntity.status(HttpStatus.CREATED).body(product); }
     */

    @PostMapping
    public ResponseEntity<?> addProduct(@Valid @RequestBody Product product,
BindingResult result) {
        List<String> displayErrors = getErrors(result);

        if (!displayErrors.isEmpty()) {
            return ResponseEntity.badRequest().body(displayErrors);
        }

        products.add(product);
        return ResponseEntity.status(HttpStatus.CREATED).body(product);
    }

    @GetMapping
    public List<Product> getProduct(){
        return products;
    }
}

```

```

@PutMapping("/{productId}")
public ResponseEntity<?> updateProduct(@PathVariable Long productId,
@Valid @RequestBody Product updatedProduct, BindingResult result) {
    Product existingProduct = findProductById(productId);

    if (existingProduct == null) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Product
not found");
    }

    List<String> displayErrors = getErrors(result);

    if (!displayErrors.isEmpty()) {
        return ResponseEntity.badRequest().body(displayErrors);
    }

    // Update the existing product
    existingProduct.setName(updatedProduct.getName());
    existingProduct.setPrice(updatedProduct.getPrice());

    return ResponseEntity.ok(existingProduct);
}

// Helper method to extract field errors from BindingResult
private List<String> getErrors(BindingResult result) {
    List<String> displayErrors = new ArrayList<>();
    if (result.hasErrors()) {
        List<FieldError> errors = result.getFieldErrors();
        for (FieldError error : errors) {
            displayErrors.add(error.getField() + ": " +
error.getDefaultMessage());
        }
    }
    return displayErrors;
}

// Helper method to find a product by ID
private Product findProductById(Long productId) {
    for(Product p:products) {
        if(p.getId().equals(productId)) {
            return p;
        }
    }
}

```

```
    }  
    return null;  
}  
  
}
```

O/p:

- POST (<http://localhost:8080/api/products>)

```
{  
  "id": "1",  
  "name": "sugar",  
  "price": 200  
}
```

- PUT (<http://localhost:8080/api/products/1>)

```
{  
  "id": "1",  
  "name": "sugar syrup",  
  "price": 50.0  
}
```

POST



<http://localhost:8080/api/products>

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1  {  
2    ... "id": "1",  
3    ... "name": "sugar",  
4    ... "price": 200  
5  }
```

Body Cookies Headers (5) Test Results

Pretty

Raw

Preview

Visualize

JSON ▼



```
1  {  
2    "id": 1,  
3    "name": "sugar",  
4    "price": 200.0  
5  }
```



GET



<http://localhost:8080/api/products>

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  [  
2    {  
3      "id": 1,  
4      "name": "sugar",  
5      "price": 200.0  
6    }  
7  ]
```


PUT



<http://localhost:8080/api/products/1>

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1  {
2    ... "id": "1",
3    ... "name": "sugar syrup",
4    ... "price": 50.0
5  }
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼



```
1  {
2    "id": 1,
3    "name": "sugar syrup",
4    "price": 50.0
5  }
```

GET



<http://localhost:8080/api/products>

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  [  
2    {  
3      "id": 1,  
4      "name": "sugar syrup",  
5      "price": 50.0  
6    }  
7  ]
```

POST



http://localhost:8080/api/products

Params

Authorization

Headers (8)

Body ●

Pre-request Sc

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ i

```
1 {  
2   ... "id": "",  
3   ... "name": "",  
4   ... "price": -1  
5 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 [  
2   "price: must be greater than or equal to 1",  
3   "id: id is required",  
4   "name: Name is required"  
5 ]
```

POST



http://localhost:8080/api/products

Params

Authorization

Headers (8)

Body ●

Pre-request Sc

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ i

```
1 {  
2   ... "id": "-1",  
3   ... "name": "sugar",  
4   ... "price": 200  
5 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 [  
2   "id: must be greater than or equal to 1"  
3 ]
```