

# SPRING BOOT VALIDATION

---

Nguyễn Huy Cường - [nh.cuong@hutech.edu.vn](mailto:nh.cuong@hutech.edu.vn)

04/2023

# Nội dung

1. Validation là gì?
2. Validation trong Spring boot
3. Demo

# 1. Validation là gì?

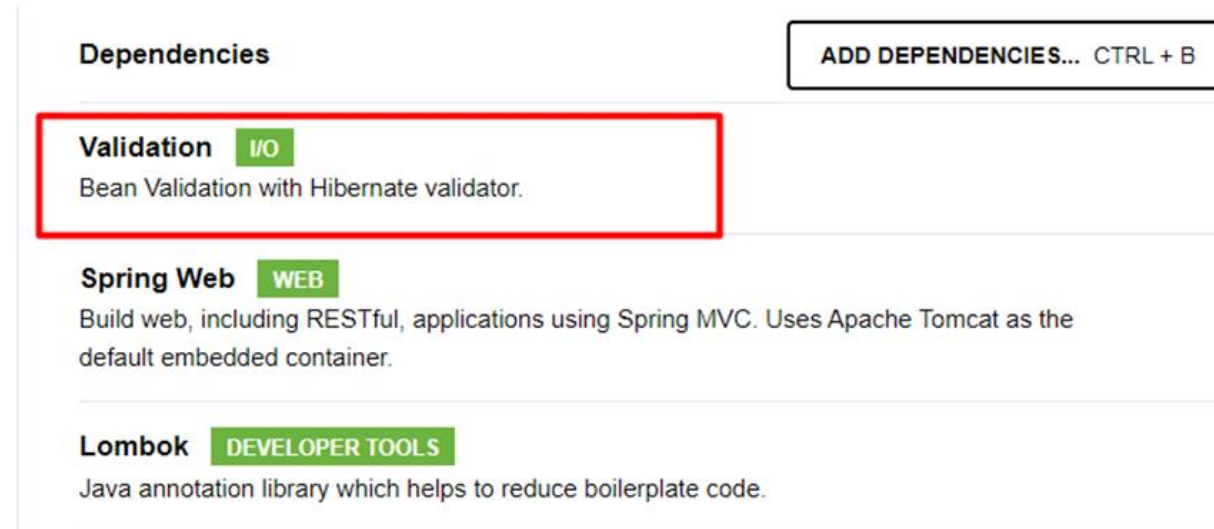
- Validation là hành động kiểm tra tính hợp lệ của dữ liệu.
  - Lợi ích:
    - ❑ Đảm bảo hệ thống hoạt động an toàn, tránh xử lý dữ liệu lỗi,
    - ❑ Ngăn chặn một số cuộc tấn công phổ biến: SQL injection, XSS...
- => Tất cả những việc kiểm tra dữ liệu trước xử lý đều được gọi là **validation**.
- Validation nên được thực hiện cả 2 phía client và server

Name:	<input type="text"/>	<i>Customer name should be between 2 and 30 characters long</i>
Email:	<input type="text" value="abc"/>	<i>Email address is not valid</i>
Age:	<input type="text"/>	<i>Customer age should be in years</i>
Gender:	<input type="text" value="Select Gender"/>	<i>This is a required field</i>
Birthday:	<input type="text" value="2014/12/12"/>	<i>birthday is of invalid format</i>
Phone:	<input type="text" value="sdsd"/>	<i>Invalid format, valid formats are 1234567890, 123-456-7890 x1234</i>
<input type="button" value="Save Customer"/>		

## 2. Validation trong Spring boot

- Thêm **validation** trong spring boot (thêm Validation dependency).

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-validation</artifactId>  
</dependency>
```



- Sử dụng validation
  - ❑ Thêm **annotation** trên các **field** của class
  - ❑ Khi sử dụng class có ràng buộc cần đảm bảo hợp lệ
- Quản lý error validation
  - ❑ Dùng tham số: **BindingResult**. Nếu validation fail, method vẫn sẽ được gọi

```
if (result.hasErrors()) { .... }
```

## Validation trong Spring boot

### Sử dụng validation – thêm trên field của class

- Thêm **annotation** trên các **field** của class

- ❑ **@NotNull** – kiểm tra giá trị null
- ❑ **@AssertTrue** – kiểm tra giá trị thuộc tính là true
- ❑ **@Size** – kiểm tra độ dài min,max
- ❑ **@Min** – kiểm tra GTNN @Max – Kiểm tra GTLN
- ❑ **@Email** – kiểm tra email có hợp lệ
- ❑ **@NotEmpty** – kiểm tra không được trống và empty
- ❑ **@NotBlank** – kiểm tra giá trị không được null hoặc khoảng trắng
- ❑ **@ and @PositiveOrZero** – kiểm tra chỉ được phép là số nguyên dương từ 0 trở đi
- ❑ **@NegPositiveative** and **@NegativeOrZero** – kiểm tra số âm
- ❑ **@Past** and **@PastOrPresent** – kiểm tra ngày từ quá khứ đến hiện tại.
- ❑ **@Future** and **@FutureOrPresent** – kiểm tra ngày từ hiện tại đến tương lai

```
12 usages
public class Product {
    3 usages
    private int id;
    3 usages
    @NotBlank(message = "tên sản phẩm không được để trống")
    private String name;
    3 usages
    @Length(min = 0, max = 50, message = "tên hình ảnh không quá 50 ký tự")
    private String image;
    3 usages
    @NotNull(message = "giá sản phẩm không được để trống")
    @Min(value = 1, message = "giá sản phẩm không được nhỏ hơn 1")
    @Max(value = 999999999, message = "giá sản phẩm không được lớn hơn 999999999")
    private long price;
```

Validation trong Spring boot

## Sử dụng validation – đảm bảo hợp lệ trên controller

- Khi sử dụng class có ràng buộc cần đảm bảo hợp lệ
  - ❑ **@Valid**
  - ❑ **@Validated** trên tham số (thuộc class đó)

## Validation trong Spring boot

### Quản lý error validation

- Dùng tham số: **BindingResult** để kiểm tra valid, và xử lý phù hợp.

Nếu validation fail, method vẫn sẽ được gọi

```
if (result.hasErrors())
{
    //VD: xử lý quay lại...
}
```

```
@GetMapping("/create")
public String create(Model model)
{
    model.addAttribute(attributeName: "product", new Product());
    return "products/create";
}

@PostMapping("/create")
public String create(@Valid Product newProduct,
                    BindingResult result,
                    Model model) {
    if (result.hasErrors())
    {
        model.addAttribute(attributeName: "product", newProduct);
        return "products/create";
    }
    //save image to static/images folder
```

- Sử dụng `@ControllerAdvice` các lỗi tổng quát



## 3- Chương trình thêm tìm kiếm, thêm, xóa sửa phẩm

1. Viết chương trình cho phép: lấy sản phẩm, thêm, sửa, xóa sản phẩm và có validation tương ứng.

Home Product		Search	
Create New Product			
#	Name	Image	Price
1	Sản phẩm 1	1.jpg	29312
2	Sản phẩm 2	2.jpg	124246

Mỗi sản phẩm có: Mã sản phẩm, Tên sản phẩm, Hình ảnh (tên hình), Giá

2. Có thể upload hình ảnh lên server

3. Thực hiện chức năng tìm kiếm theo tên sản phẩm



## Thực hiện Demo

B0: Thiết lập các **dependency** cần thiết cho project

B1: Thêm các lớp

models/**Product**

service/**ProductService**

controller/**ProductController**

B2: Thiết kế trang layout (master page)

B3: Thực hiện **Create** : Validation model, GET, POST

B4: Thực hiện **Products** (GET): lấy ds các sản phẩm

B5: Chỉnh sửa lại theo yêu cầu logic

- Sử dụng hình ảnh upload (từ local) và lưu vào folder **/images** ở server

- Tăng id sản phẩm lên 1 sau khi **create**

B6: Thực hiện Edit   B7: Thực hiện Delete, Search [Tự thực hiện]

## Demo

## Bo: Thiết lập dependency (có validation)

## Bo: Thiết lập các dependency cần thiết cho project

```

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

```

## Dependencies

ADD DEPENDENCIES... CTRL + B

## Validation I/O

Bean Validation with Hibernate validator.

## Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

## Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

## Thymeleaf TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

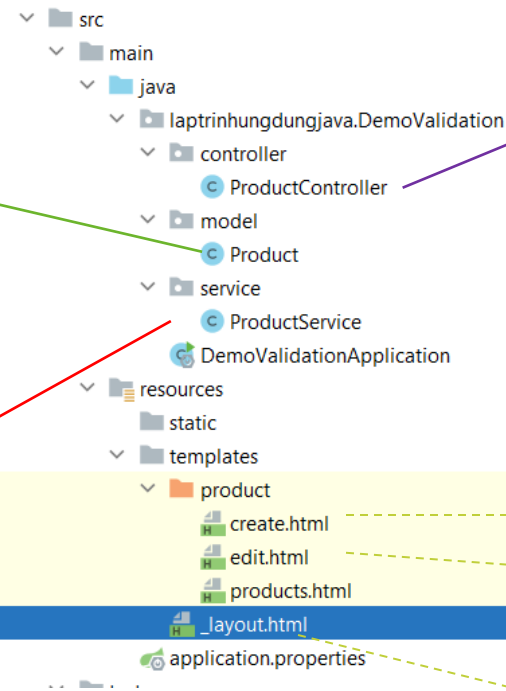
## Demo

## B1: Xây dựng kiến trúc: model – service – controller

```

@Getter @Setter
@NoArgsConstructor
@AllArgsConstructor
public class Product {
    private int id;
    private String name;
    private String image;
    private long price;
}

```



Bước 3

Bước 4

Bước 2

```

@Controller
@RequestMapping("/products")
public class ProductController {
    @Autowired
    private ProductService productService;

    @GetMapping("/create")
    public String Create(Model model) {
        model.addAttribute("product", new Product());
        return "product/create";
    }

    @PostMapping("/create")
    public String Create(@Valid Product newProduct,
        BindingResult result, Model model) {
        if (result.hasErrors()) {
            model.addAttribute("course", newProduct);
            return "product/create";
        }
        productService.add(newProduct);
        return "redirect:/products";
    }

    @GetMapping()
    public String Index(Model model)
    {
        model.addAttribute("listproduct", productService.getAll());
        return "product/products";
    }
}

```

```

@Service
public class ProductService {
    private List<Product> listProduct = new ArrayList<>();

    public List<Product> getAll()
    {
        return listProduct;
    }

    public void add(Product newProduct)
    {
        listProduct.add(newProduct);
    }

    //Edit, Delete, Search ...
}

```

## Demo

## B2: Thiết kế layout (master page) cho ứng dụng

- Tìm kiếm mẫu master page phù hợp & bố cục lại layout

VD: tìm ở getbootstrap & bố cục lại layout

- ❑ Thêm <div> có **layout:fragment** để có thể sử dụng ở content page

Home Products

Search

Search

```
<div layout:fragment="content" class="container body-content">
```

```
</div>
```

Các content Page như: Xem danh sách, Thêm, Xóa, Sửa, ...

```
<!DOCTYPE html>
<html lang="en"
  xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="BigSchool - Home page with courses and search functionality">
  <title>Demo Products</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMHjY6hW+ALEwIH"
    crossorigin="anonymous">
</head>
<body>
  <nav class="navbar bg-dark border-bottom border-body" data-bs-theme="dark">
    <div style="padding-left: 40px">
      <a class="navbar-brand" th:href="@{/products}">Home</a>
      <a class="navbar-brand" th:href="@{/products/create}">Products</a>
    </div>
    <form class="d-flex" role="search" th:action="@{/home/search}" th:method="get">
      <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success" type="submit">Search</button>
    </form>
  </nav>

  <div layout:fragment="content" class="container body-content" ...>
    <div class="card-footer text-center">
      <hr/>
      <p>&copy; Nguyễn Huy Cường - Lập trình ứng dụng với Java</p>
    </div>
  </div>
</body>
</html>
```

## Demo

## B3-1: Thực hiện CREATE (GET &amp; POST) ở controller

- Thực hiện Validation cho models/product
- Thực hiện **Create** ở ProductController

```
public class Product {
    no usages
    private int id;

    no usages
    @NotBlank(message = "Tên sản phẩm không được để trống")
    private String name;

    no usages
    @Length(min= 0,max = 200, message = "Tên hình ảnh không quá 200 kí tự")
    private String image;

    no usages
    @NotNull(message = "Giá sản phẩm không được để trống")
    @Min(value = 1, message = "Giá sản phẩm không được nhỏ hơn 1")
    @Max(value = 9999999, message = "Giá sản phẩm không được lớn hơn 9999999")
    private long price;
}
```

```
@Controller
@RequestMapping("/products")
public class ProductController {
    3 usages
    @Autowired
    private ProductService productService;
    no usages
    @GetMapping("/create")
    public String Create(Model model) {
        model.addAttribute( attributeName: "product", new Product());
        return "product/create";
    }
    no usages
    @PostMapping("/create")
    public String Create(@Valid Product newProduct,
        BindingResult result,
        @RequestParam MultipartFile imageProduct,
        Model model) {
        if (result.hasErrors()) {
            model.addAttribute( attributeName: "product", newProduct);
            return "product/create";
        }
        return "redirect:/products";
    }
}
```

## Demo


## B3-2: Thực hiện Create View ở templates/product/create

## ● Tạo HTML cho create

```

<!DOCTYPE html>
<html lang="en"
  xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
  layout:decorate="_layout">
<head>
  <meta charset="UTF-8">
  <title>Create product</title>
</head>
<body>
  <div layout:fragment="content" class="container body-content">
    <form th:action="@{/products/create}" th:object="${product}" method="post" class="form">
      <div class="form-group">
        <label for="name">Name</label>
        <input class="form-control" type="text" th:field="*{name}" id="name" placeholder="Product Name">
        <div class="alert alert-warning" th:if="${#fields.hasErrors('name')}" th:errors="*{name}"></div>
      </div>
      <div class="form-group">
        <label for="image">Image:</label>
        <input class="form-control" type="text" th:field="*{image}" id="image" placeholder="image">
        <div class="alert alert-warning" th:if="${#fields.hasErrors('image')}" th:errors="*{image}"></div>
      </div>
      <div class="form-group">
        <label for="price">Price:</label>
        <input class="form-control" type="number" th:field="*{price}" id="price" placeholder="price">
        <div class="alert alert-warning" th:if="${#fields.hasErrors('price')}" th:errors="*{price}"></div>
      </div>
      <input type="submit" class="btn btn-success" value="Add Product">
    </form>
  </div>
</body>
</html>

```



Name
Product Name
tên sản phẩm không được để trống
Image:
image
Price:
999999999999999
giá sản phẩm không được lớn hơn 999999999
Add Product

## Demo

## B4-1: Thực hiện products: Lấy danh sách sản phẩm

## ● Thực hiện ProductController

@GetMapping()

public String Index(Model model)

```
{
    model.addAttribute("listproduct", productService.getAll());
    return "product/products";
}
```

Home Products				
Create New Product				
#	Name	Image	Price	
0	Sản phẩm 1	hình 1.jpg	1234567	Edit Delete
0	Sản phẩm 2	hình 2.jpg	2345678	Edit Delete

## ● Thực hiện View

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="$_layout"
      xmlns:custom="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="UTF-8">
  <title>List product</title>
</head>
<body>
  <div layout:fragment="content" class="container body-content">
    <a th:href="@{products/create}" class="btn btn-primary">Create New Product</a>
    <table class="table table-striped">
      <thead>
        <tr>
          <th scope="col">#</th>
          <th scope="col">Name</th>
          <th scope="col">Image</th>
          <th scope="col">Price</th>
          <th scope="col"></th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="product : ${listproduct}">
          <th scope="row" th:text="${product.id}"></th>
          <td th:text="${product.name}"></td>
          <td th:text="${product.image}"></td>
          <td th:text="${product.price}"></td>
          <td>
            <a th:href="@{/products/edit/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-secondary">Edit</a>
            <a th:href="@{/products/delete/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-danger">Delete</a>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

## Demo

## B5-1: Thực hiện upload hình ảnh [ở View]

- Sử dụng type = “file” để cho phép người dùng chọn file  
Nếu chỉ muốn file hình ảnh thêm accept="image/png, image/jpeg"
- Điều chỉnh **enctype** ở form để cho phép submit

```
<form th:action="@{/products/create}" th:object="${product}" method="post" class="form" enctype = "multipart/form-data">
  <div class="form-group">
    <label for="name">Name</label>
    <input class="form-control" type="text" th:field="*{name}" id="name" placeholder="Product Name">
    <div class="alert alert-warning" th:if="${#fields.hasErrors('name')}" th:errors="*{name}"></div>
  </div>
  <div class="form-group">
    <label for="image">Image:</label>
    <input class="form-control" type="text" th:field="*{image}" id="image" placeholder="image">-->
    <input class="form-control" type="file" id="image" name="imageProduct" accept="image/png, image/jpeg">
    <div class="alert alert-warning" th:if="${#fields.hasErrors('image')}" th:errors="*{image}"></div>
  </div>
```



## Demo

## B5-2: Thực hiện upload hình ảnh [ở controller - create]

- Khi create, lấy MultipartFile và lưu vào static/**images** với tên file Random

```

@PostMapping("/create")
public String Create(@Valid Product newProduct,
                    BindingResult result,
                    @RequestParam MultipartFile imageProduct,
                    Model model) {
    if (result.hasErrors()) {
        model.addAttribute("product", newProduct);
        return "product/create";
    }
    productService.updateImage(newProduct, imageProduct);
    productService.add(newProduct);
    return "redirect:/products";
}

```

```

public void add(Product newProduct)
{
    int maxId = listProduct.stream().mapToInt(Product::getId).max().orElse(0);
    newProduct.setId(maxId+1);
    listProduct.add(newProduct);
}

public void updateImage(Product newProduct, MultipartFile imageProduct)
{
    if (!imageProduct.isEmpty()) {
        try
        {
            Path dirImages = Paths.get("static/images");
            if (!Files.exists(dirImages)) {
                Files.createDirectories(dirImages);
            }
            String newFileName = UUID.randomUUID() + "_" +
            imageProduct.getOriginalFilename();
            Path pathFileUpload = dirImages.resolve(newFileName);
            Files.copy(imageProduct.getInputStream(), pathFileUpload,
            StandardCopyOption.REPLACE_EXISTING);
            newProduct.setImage(newFileName);
        }
        catch (IOException e) {
            e.printStackTrace(); // Handle the exception appropriately
        }
    }
}



```

## Demo

## B5-3: Hiển thị danh sách sản phẩm có hình ảnh - [ở View]

- Hiển thị hình ảnh lấy từ *static/images/{product.image}*

```
<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Name</th>
      <th scope="col">Image</th>
      <th scope="col">Price</th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="product : ${listproduct}" >
      <th scope="row" th:text="${product.id}"></th>
      <td th:text="${product.name}"></td>
      <!-- <td th:text="${product.image}"></td-->
      <td>
        
      </td>
      <td th:text="${product.price}"></td>
      <td>
        <a th:href="@{/products/edit/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-secondary">Edit</a>
        <a th:href="@{/products/delete/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-danger">Delete</a>
      </td>
    </tr>
  </tbody>
</table>
```

Create New Product				
#	Name	Image	Price	
1	Sản phẩm 1		29312	<a href="#">Edit</a> <a href="#">Delete</a>
2	Sản phẩm 2		124246	<a href="#">Edit</a> <a href="#">Delete</a>

## Demo

## B5-4: Tăng id của sản phẩm khi tạo mới

- Tìm **maxid** của ds sản phẩm

- Khi thêm mới

**id = max id + 1**

```
@Service
public class ProductService {
    private List<Product> listProduct = new ArrayList<>();
    public List<Product> getAll()
    {
        return listProduct;
    }
    public void add(Product newProduct)
    {
        int maxId = listProduct.stream().mapToInt(Product::getId).max().orElse(0);
        newProduct.setId(maxId+1);
        listProduct.add(newProduct);
    }
    public Product get(int id)
    {
        return listProduct.stream().filter(p->p.getId() == id).findFirst().orElse(null);
    }

    //Update, Delete, Search ...
}
```

## Demo

## B6: Thực hiện Edit – Get (Lấy sản phẩm theo id)

- Chú ý: ở Edit lấy hình ảnh cũ, và có thể thay thế ảnh mới.
- Tự thực hiện JS để cập nhật ảnh

```
@GetMapping("/edit/{id}")
public String Edit(@PathVariable int id, Model model) {
    Product find = productService.get(id);
    if(find == null)
        return "Product not found with ID: " + id; //error page
    model.addAttribute("product", find);
    return "product/edit";
}
```

```
<!DOCTYPE html>
<html lang="en"
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="$_layout">
<head>
    <meta charset="UTF-8">
    <title>Edit product</title>
</head>
<body>
<div layout:fragment="content" class="container body-content">
    <form th:action="@{/products/edit}" th:object="${product}" method="post" class="form" enctype="multipart/form-data">
        <input type="hidden" th:field="**{id}"/> <!-- Hidden input field for product ID -->
        <div class="form-group">
            <label for="name">Name</label>
            <input class="form-control" type="text" th:field="**{name}" id="name" placeholder="Product Name">
            <div class="alert alert-warning" th:if="${#fields.hasErrors('name')}" th:errors="**{name}"></div>
        </div>
        <div class="form-group">
            <label for="name">Image:</label>
            
        </div>
        <div class="form-group">
            <label for="image">Change Image:</label>
            <input class="form-control-file" type="file" id="image" name="imageProduct" accept="image/png,image/jpeg">
            <div class="alert alert-warning" th:if="${#fields.hasErrors('image')}" th:errors="**{image}"></div>
        </div>
        <div class="form-group">
            <label for="price">Price:</label>
            <input class="form-control" type="number" th:field="**{price}" id="price" placeholder="price">
            <div class="alert alert-warning" th:if="${#fields.hasErrors('price')}" th:errors="**{price}"></div>
        </div>
        <input type="submit" class="btn btn-success" value="Save Product">
    </form>
</div>
</body>
</html>
```

## Demo

## B6: Thực hiện Edit – Post

- Có thể viết chung hàm lưu ảnh vào images cho cả create, update có thể thay thế ảnh mới.

```

@PostMapping("/edit")
public String Edit(@Valid Product editProduct,
                  BindingResult result,
                  @RequestParam MultipartFile imageProduct,
                  Model model) {
    if (result.hasErrors()) {
        model.addAttribute("product", editProduct);
        return "product/edit"; // Return to the edit form with error messages
    }
    productService.updateImage(editProduct, imageProduct);
    productService.update(editProduct);
    return "redirect:/products"; // Redirect to the products page after successful update
}

```

```

1 usage
public void update(Product editProduct)
{
    Product find = get(editProduct.getId());
    if(find != null) {
        find.setImage(editProduct.getImage());
        find.setPrice(editProduct.getPrice());
        find.setName(editProduct.getName());
    }
}

2 usages
public void updateImage(Product newProduct, MultipartFile imageProduct)
{
    if (!imageProduct.isEmpty()) {
        try
        {
            Path dirImages = Paths.get( first: "static/images");
            if (!Files.exists(dirImages)) {
                Files.createDirectories(dirImages);
            }
            // Combine directory and file name to get the complete path
            String newFileName = UUID.randomUUID() + "_" + imageProduct.getOriginalFilename();
            Path pathFileUpload = dirImages.resolve(newFileName);
            Files.copy(imageProduct.getInputStream(), pathFileUpload, StandardCopyOption.REPLACE_EXISTING);
            newProduct.setImage(newFileName);
        }
        catch (IOException e) {
            e.printStackTrace(); // Handle the exception appropriately
        }
    }
}

```