

LẬP TRÌNH WEB (WEBPR330479)

Thymeleaf

THS. NGUYỄN HỮU TRUNG

- Ths. Nguyễn Hữu Trung
- Khoa Công Nghệ Thông Tin
- Trường Đại học Sư Phạm Kỹ Thuật TP.HCM
- 090.861.7108
- trungnh@hcmute.edu.vn
- <https://www.youtube.com/@baigiai>



- Giới thiệu Thymeleaf
- Biến trong Thymeleaf
- Standard Expression trong Thymeleaf
- Câu lệnh điều kiện th
- Vòng lặp th
- Toán tử trong Thymeleaf
- Fragment trong Thymeleaf
- Vòng lặp trong Thymeleaf
- Cấu hình view trong spring sử dụng Thymeleaf

- Thymeleaf là một thư viện mã nguồn mở và được coi là một Java Template Engine, được sử dụng để xử lý và tạo ra các trang HTML, XML, JavaScript, CSS, text, và raw content.
- Thymeleaf sử dụng các thẻ HTML làm view và tích hợp trực tiếp vào các tệp HTML. Điều này đồng nghĩa rằng bạn không cần phải thêm các thẻ "non-HTML" vào tệp HTML của mình.
 - ▣ Ví dụ, khi bạn sử dụng JSP, bạn cần phải sử dụng thẻ JSTL và khai báo taglib. Tuy nhiên, với Thymeleaf, bạn chỉ cần sử dụng các thẻ HTML chuẩn mà không cần thêm bất kỳ thẻ nào khác.

- ❑ **Biến đối tượng (th:object):** Biến đối tượng cho phép bạn đặt một đối tượng Java vào phạm vi của Thymeleaf. Ví dụ, nếu bạn có một đối tượng Person với thuộc tính name,.

```
<div th:object="${person}">
  <p>Name: <span th:text="*{name}"></span></p>
</div>
```

- ❑ Ở đây, **`${person}`** đại diện cho đối tượng **Person**, và bạn có thể truy cập thuộc tính name của nó bằng cách sử dụng **`*{name}`**

- ❑ **Biến trường (th:field):** Biến trường làm cho việc xử lý các trường dữ liệu trong các biểu mẫu HTML trở nên dễ dàng hơn. Ví dụ, nếu bạn có một biểu mẫu đăng ký với trường **username**

```
<form th:object="${user}" th:action="@{/register}" method="post">
  <label for="username">Username</label>
  <input type="text" id="username" th:field="*{username}" />
</form>
```

- ❑ Trong ví dụ ***\${user}*** đại diện cho đối tượng ***User***, và ****{username}*** là cách Thymeleaf liên kết trường ***username*** trong biểu mẫu HTML với thuộc tính ***username*** của đối tượng ***User***

- **Hiển thị Giá trị Biến:** Khi bạn đã định nghĩa biến trong Thymeleaf, bạn có thể hiển thị giá trị của chúng bằng cách sử dụng các thuộc tính Thymeleaf như **th:text**, **th:value**, **th:src**, và nhiều thuộc tính khác.

- ✓ **Hiển thị giá trị văn bản (th:text)**

```
<p th:text="${user.name}"></p>
```

Trong ví dụ này, **th:text** sẽ hiển thị giá trị của thuộc tính name của đối tượng user

- ✓ **Gán giá trị vào trường (th:value)**

```
<input type="text" th:value="${user.email}" />
```

Trong ví dụ này, **th:value** cho phép bạn gán giá trị của thuộc tính email vào trường nhập liệu.

❑ Standard Expression dùng để lấy thông tin trong Model

❑ Biểu thức `${...}` - Truy cập Giá Trị Biến

Trong Controller đặt một giá trị:

```
model.addAttribute("name", "Nguyễn Hữu Trung")
```

Trong Thymeleaf, để lấy giá trị biến "name" ta dùng:

```
<p>Họ tên: <span th:text="${name}"></span>.</p>
```

❑ Biểu thức `*{...}` - Truy cập giá trị biến trên các phần tử, nó sẽ lấy giá trị của biến trong ngữ cảnh của **th:object**

```
<div th:object="${session.user}">
```

```
<!-- th:object tồn tại trong phạm vi của thẻ div này Lấy tên từ đối tượng session.user -->
```

```
<p>Name: <span th:text="*{firstName}"></span></p>
```

```
<!-- Lấy lastName từ đối tượng session.user -->
```

```
<p>Surname: <span th:text="*{lastName}"></span></p>
```

```
</div>
```

❑ Biểu thức `#{...}` - Lấy Thông Báo (Message)

Ví dụ, trong tập cấu hình **.properties** của bạn, bạn có một thông báo chào đón người dùng bằng nhiều ngôn ngữ

```
home.welcome=Hello bạn
```

Thì có thể lấy nó ra:

```
<p th:utext="#{home.welcome}">Xin chào các bạn!</p>
```

Đoạn văn bản tiếng việt bên trong thẻ **p** sẽ bị thay thế bởi **Thymeleaf** khi render **`#{home.welcome}`**. Điều này cho phép bạn tạo các ứng dụng hỗ trợ nhiều ngôn ngữ một cách dễ dàng.

❑ Biểu thức `@{...}` - Lấy và tạo các URL động

```
<!-- Tương đương 'http://localhost:8080/order/details?orderId=3' -->  
<a href="details.html" th:href="@{http://localhost:8080/order/details(orderId=${o.id})}" >view</a>  
<!-- Tương đương '/order/details?orderId=3' -->  
<a href="details.html" th:href="@{/order/details(orderId=${o.id})}">view</a>  
<!-- Tương đương '/gtvg/order/3/details' -->  
<a href="details.html" th:href="@{/order/{orderId}/details(orderId=${o.id})}" >view</a>
```

- Sử dụng câu lệnh điều kiện **th:if**, cùng với câu lệnh **th:switch** và **th:case** để làm cho việc kiểm soát dữ liệu trên trang web.
- Câu lệnh **th:if** trong Thymeleaf cho phép bạn thực hiện các kiểm tra điều kiện đơn giản trên dữ liệu và hiển thị hoặc ẩn nội dung dựa trên kết quả của kiểm tra này.

```
<p th:if="{user.isAdmin()}">This user is an admin.</p>
```

- Câu lệnh **th:switch** và **th:case** giúp bạn xử lý nhiều lựa chọn và hiển thị nội dung tương ứng với lựa chọn đã chọn.

```
<div th:switch="{user.role}">
  <p th:case="'ADMIN'">Welcome, Admin!</p>
  <p th:case="'USER'">Welcome, User!</p>
  <p th:case="*">Unknown Role</p>
</div>
```

- **th:each** là một cú pháp trong Thymeleaf cho phép bạn lặp qua danh sách hoặc bất kỳ tập hợp dữ liệu nào, và áp dụng một phần tử HTML cho mỗi phần tử trong danh sách.

```
<div th:each="item : ${items}">
    <!-- Nội dung bạn muốn hiển thị cho mỗi item -->
</div>
```

Trong đó:

- "item" là biến mà bạn sẽ sử dụng để tham chiếu đến mỗi phần tử trong danh sách.
- "\${items}" là danh sách hoặc tập hợp dữ liệu bạn muốn lặp qua.

```
<ul>
    <li th:each="product : ${products}">
        <span th:text="${product.name}"></span>:
        <span th:text="${product.price}"></span> USD
    </li>
</ul>
```

Sử Dụng Vòng Lặp **th:each** Với Điều Kiện **th:if**

12

- Trong ví dụ này, chúng ta chỉ hiển thị các sản phẩm có sẵn trong kho (có thuộc tính `inStock` là `true`).

```
<ul>
  <li th:each="product : ${products}" th:if="${product.inStock}">
    <span th:text="${product.name}"></span>:
    <span th:text="${product.price}"></span> USD
  </li>
</ul>
```

```
<div th:if="${not empty users}">
  <ul>
    <li th:each="user : ${users}">
      <span th:if="${user.isAdmin()}" th:text="${user.name + ' (Admin)'}" >
      </span> <span th:unless="${user.isAdmin()}" th:text="${user.name + ' (User)'}" >
      </span>
    </li>
  </ul>
</div>
```

- Trong ví dụ này, chúng ta kiểm tra xem danh sách `users` có phần tử nào không.
- Nếu danh sách không trống, chúng ta duyệt qua từng `user` trong danh sách.
- Với mỗi `user`, chúng ta kiểm tra xem họ có phải là `admin` không, và hiển thị tên của họ với tên vai trò tương ứng.

- **Toán tử toán học:** Thymeleaf hỗ trợ tất cả các toán tử toán học cơ bản, bao gồm cộng (+), trừ (-), nhân (*), chia (/), và modulo (%).

```
<p th:text="${5 + 3}"></p> <!-- Kết quả: 8 -->
<p th:text="${totalPrice = price * quantity}"></p>
```

- **Toán tử so sánh và logic:** Bạn có thể sử dụng các toán tử so sánh như ==, !=, <, >, <=, >= và and, or, not để so sánh các giá trị và điều kiện trong Thymeleaf .

```
<p th:text="${isAdult = age >= 18}"></p>
<!-- Kết quả: true hoặc false tùy thuộc vào giá trị của age -->
```

```
<p th:text="${(hasPermission and isActive) or isAdmin}"></p>
<!-- Kết quả: Giá trị của biểu thức logic -->
```

- **Toán tử gán:** Toán tử "=" trong Thymeleaf dùng để gán giá trị cho biến hoặc thuộc tính.

```
<p th:with="x=10" th:text="${x}"></p>
<!-- Kết quả: 10 -->
```

- Fragments cho phép bạn tái sử dụng các phần của trang web của bạn, giúp tạo ra giao diện thú vị, dễ quản lý và hiệu quả.
- Trong Thymeleaf, fragment là một phần của trang web, chẳng hạn như header, footer, menu, hoặc bất kỳ phần nào bạn muốn tái sử dụng. Fragments cho phép bạn đóng gói các phần này thành các module độc lập mà bạn có thể sử dụng lại trên nhiều trang web khác nhau. Điều này giúp giảm lặp lại mã nguồn và tạo ra các trang web dễ bảo trì.
- Để bắt đầu sử dụng fragments, bạn cần định nghĩa chúng trong các tệp template HTML của bạn bằng cách sử dụng các thuộc tính đặc biệt của Thymeleaf như **th:fragment**:

```
<!-- Trong header.html -->
<div th:fragment="header">
    <!-- Nội dung của header -->
</div>
<!-- Trong footer.html -->
<div th:fragment="footer">
    <!-- Nội dung của footer -->
</div>
```

- Khi bạn đã định nghĩa các fragment, bạn có thể sử dụng chúng trong các trang web khác nhau bằng cách sử dụng các hàm **th:replace**, **th:include**, hoặc **th:insert**. Ví dụ:

```
<!-- Trong trang web index.html -->
<!DOCTYPE html>
<html> <head> <title>Trang Chủ</title> </head>
<body>
<header th:replace="fragments/header :: header"></header>
<div class="content">
    <!-- Nội dung của trang web -->
</div>
<footer th:replace="fragments/footer :: footer"></footer>
</body>
</html>
```

- **Cấu hình Maven:** thêm dependency spring boot và Thymeleaf vào file pom.xml

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

- **Cấu hình Thymeleaf:** Cấu hình ứng dụng thông qua tệp **application.properties**

```
# Vô hiệu hóa tính năng cache của Thymeleaf để phát triển nhanh hơn
spring.thymeleaf.cache=false
```

- ❑ **Tạo Trang Index:** Trang index.html sẽ là trang mặc định mà Thymeleaf tìm khi không có đường dẫn cụ thể. Chúng ta sẽ sử dụng nó để hiển thị thông báo chào mừng và một nút liên kết tới trang profile.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<title>Hello World</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<!-- CSS -->
<link th:href="@{/css/bootstrap.css}" rel="stylesheet" />
<!-- JavaScript -->
<script th:src="@{/js/bootstrap.js}"></script>
</head>
<body>
<h1 th:text="#{test.hello}"></h1>
<a th:href="@{/profile}" class="btn btn-primary">User Profile</a>
</body>
</html>
```

□ Tạo @Controller cho /profile

```
@Controller
public class WebController {
    @GetMapping("/profile")
    public String profile(Model model) {
        // Tạo danh sách thông tin cá nhân
        List<Info> profile = new ArrayList<>();
        profile.add(new Info("fullname", "Hữu Trung"));
        profile.add(new Info("nickname", "HuuTrung"));
        profile.add(new Info("email", "trungnhspkt@gmail.com"));
        profile.add(new Info("website", "https://iotstar.vn"));
        // Đưa danh sách vào Model
        model.addAttribute("profile", profile);
        return "profile"; // Trả về template profile.html    }
```

□ Tạo Trang Profile

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <title>User Profile</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <!-- CSS -->
    <link th:href="@{/css/bootstrap.css}" rel="stylesheet" />
    <!-- JavaScript -->
    <script th:src="@{/js/bootstrap.js}"></script>
  </head>
  <body>
    <h1 th:text="#{test.hello}"></h1>
    <h2>User Profile</h2>
    <ul>
      <!-- Duyệt qua toàn bộ danh sách UserProfile -->
      <li th:each="info : ${profile}"> <!-- Lấy key và value từ danh sách -->
        <span th:text="*{info.key}"></span> : <span th:text="*{info.value}"></span> </li>
    </ul>
  </body>
</html>
```

- ❑ **Bước 1:** Thêm các dependency sau vào file pom.xml để tạo dự án Spring Boot.

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.thymeleaf.extras</groupId>
<artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
<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-web</artifactId>
</dependency>
```

```
<!-- MySQL Connector -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
</dependency>
```

□ Bước 2: Tạo Database: test_db trong MySQL và cấu hình file application.properties

file **application.properties**

```
# Chạy ứng dụng trên port 8088
server.port=8088
# Tắt tính năng cache của Thymeleaf để phát triển nhanh hơn
spring.thymeleaf.cache=false
# Định nghĩa thư mục chứa các message đa ngôn ngữ
spring.messages.basename=i18n/messages
# Chỉ định ngôn ngữ mặc định (Tiếng Việt)
spring.mvc.locale=vi_VN
# Cấu hình kết nối đến cơ sở dữ liệu MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/test_db?useSSL=false
spring.datasource.username=root
spring.datasource.password=root
# Hibernate Properties
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto = update
```

□ Bước 3: Tạo Entity

```
*CategoryEntity.java x
1 package vn.iostar.Entity;
2
3 import java.io.Serializable;
4 import java.util.Set;
5
6 import jakarta.persistence.*;
7 import lombok.*;
8
9 @Data
10 @AllArgsConstructor
11 @NoArgsConstructor
12 @Entity
13 @Table(name = "Categories")
14 public class CategoryEntity implements Serializable {
15     private static final long serialVersionUID = 1L;
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Long categoryId;
19     @Column(name = "category_name", length = 200, columnDefinition = "nvarchar(200) not null")
20     private String name;
21
22     @OneToMany(mappedBy = "category", cascade = CascadeType.ALL )
23     private Set<ProductEntity> products;
24 }
```

□ Bước 4: Tầng Repository

```
1 package vn.iostar.Repository;
2
3 import java.util.List;
4
5 import org.springframework.data.domain.Page;
6 import org.springframework.data.domain.Pageable;
7 import org.springframework.data.jpa.repository.JpaRepository;
8 import org.springframework.stereotype.Repository;
9
10 import vn.iostar.Entity.CategoryEntity;
11
12 @Repository
13 public interface CategoryRepository extends JpaRepository<CategoryEntity, Long > {
14     //Tìm Kiểm theo nội dung tên
15     List<CategoryEntity> findByNameContaining(String name);
16     //Tìm kiếm và Phân trang
17     Page<CategoryEntity> findByNameContaining(String name, Pageable pageable);
18 }
19
```

□ Bước 5: Tầng Service chứa các logic xử lý nghiệp vụ và hỗ trợ cho tầng Controller

```
1 package vn.iostar.Service.Impl;
2 import java.util.List;
3 import java.util.Optional;
4 import org.apache.commons.lang3.StringUtils;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.domain.Example;
7 import org.springframework.data.domain.Page;
8 import org.springframework.data.domain.Pageable;
9 import org.springframework.data.domain.Sort;
10 import org.springframework.stereotype.Service;
11 import vn.iostar.Entity.CategoryEntity;
12 import vn.iostar.Repository.CategoryRepository;
13 import vn.iostar.Service.ICategoryService;
14 //khai báo service
15 @Service
16 public class CategoryServiceImpl implements ICategoryService{
17     @Autowired
18     CategoryRepository categoryRepository;
19     //source -> Generate Constructor using Field, xóa super()
20     public CategoryServiceImpl(CategoryRepository categoryRepository)
21     {
22         this.categoryRepository = categoryRepository;
23     }
24     @Override
```

```
23 @Override
24 public <S extends CategoryEntity> S save(S entity) {
25     if(entity.getCategoryId() == null) {
26         return categoryRepository.save(entity);
27     }else {
28         Optional<CategoryEntity> opt = findById(entity.getCategoryId());
29         if(opt.isPresent()) {
30             if (StringUtils.isEmpty(entity.getName())) {
31                 entity.setName(opt.get().getName());
32             }else {
33                 //lấy lại images cũ
34                 entity.setName(entity.getName());
35             }
36         }
37         return categoryRepository.save(entity);
38     }
39 }
40 }
41 @Override
42 public List<CategoryEntity> findAll() {
43     return categoryRepository.findAll();
44 }
```

□ **Bước 5:** Tầng **Service** chứa các logic xử lý nghiệp vụ và hỗ trợ cho tầng Controller

```
45 @Override
46 public Page<CategoryEntity> findAll(Pageable pageable) {
47     return categoryRepository.findAll(pageable);
48 }
49 @Override
50 public List<CategoryEntity> findAll(Sort sort) {
51     return categoryRepository.findAll(sort);
52 }
53 @Override
54 public List<CategoryEntity> findAllById(Iterable<Long> ids) {
55     return categoryRepository.findAllById(ids);
56 }
57 @Override
58 public Optional<CategoryEntity> findById(Long id) {
59     return categoryRepository.findById(id);
60 }
61 public <S extends CategoryEntity> Optional<S> findOne(Example<S> example) {
62     return categoryRepository.findOne(example);
63 }
64 @Override
65 public long count() {
66     return categoryRepository.count();
67 }
```

```
68 @Override
69 public void deleteById(Long id) {
70     categoryRepository.deleteById(id);
71 }
72 @Override
73 public void delete(CategoryEntity entity) {
74     categoryRepository.delete(entity);
75 }
76 @Override
77 public void deleteAll() {
78     categoryRepository.deleteAll();
79 }
80 @Override
81 public List<CategoryEntity> findByNameContaining(String name) {
82     return categoryRepository.findByNameContaining(name);
83 }
84 @Override
85 public Page<CategoryEntity> findByNameContaining(String name, Pageable pageable) {
86     return categoryRepository.findByNameContaining(name, pageable);
87 }
88 }
```

- **Bước 6:** Tầng **Controller** nhận các request từ người dùng và chuyển tiếp cho tầng **Service** xử lý.

```
1 package vn.iostar.Controller.admin;
2 import java.util.List;
3 import java.util.Optional;
4 import java.util.stream.Collectors;
5 import java.util.stream.IntStream;
6 import jakarta.validation.Valid;
7 import org.springframework.beans.BeanUtils;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.data.domain.Page;
10 import org.springframework.data.domain.PageRequest;
11 import org.springframework.data.domain.Pageable;
12 import org.springframework.data.domain.Sort;
13 import org.springframework.stereotype.Controller;
14 import org.springframework.ui.ModelMap;
15 import org.springframework.util.StringUtils;
16 import org.springframework.validation.BindingResult;
17 import org.springframework.web.bind.annotation.GetMapping;
18 import org.springframework.web.bind.annotation.ModelAttribute;
19 import org.springframework.web.bind.annotation.PathVariable;
20 import org.springframework.web.bind.annotation.PostMapping;
21 import org.springframework.web.bind.annotation.RequestMapping;
22 import org.springframework.web.bind.annotation.RequestParam;
23 import org.springframework.web.servlet.ModelAndView;
24 import vn.iostar.Entity.CategoryEntity;
25 import vn.iostar.Model.CategoryModel;
26 import vn.iostar.Service.ICategoryService;
```

```
28 @Controller
29 @RequestMapping("admin/categories")
30 public class CategoryController {
31     @Autowired
32     ICategoryService categoryService;
33
34     @GetMapping("add")
35     public String add(ModelMap model) {
36         CategoryModel cateModel = new CategoryModel();
37         cateModel.setIsEdit(false);
38         //chuyển dữ liệu từ model vào biến category để đưa lên view
39         model.addAttribute("category", cateModel);
40         return "admin/categories/addOrEdit";
41     }
42 }
```

- **Bước 6:** Tầng **Controller** nhận các request từ người dùng và chuyển tiếp cho tầng **Service** xử lý.

```
43 @PostMapping("saveOrUpdate")
44 public ModelAndView saveOrUpdate(ModelMap model,
45     @Valid @ModelAttribute("category") CategoryModel cateMdoel, BindingResult result) {
46     if(result.hasErrors()) {
47         return new ModelAndView("admin/categories/addOrEdit");
48     }
49     CategoryEntity entity = new CategoryEntity();
50     //copy từ Model sang Entity
51     BeanUtils.copyProperties(cateMdoel, entity);
52     //gọi hàm save trong service
53     categoryService.save(entity);
54     //đưa thông báo về cho biến message
55     String message= "";
56     if(cateMdoel.getIsEdit() == true) {
57         message="Category is Edited!!!!!!!";
58     }
59     }else {
60         message="Category is saved!!!!!!!";
61     }
62     model.addAttribute("message",message);
63     //redirect về URL controller
64     return new ModelAndView("forward:/admin/categories/searchpaginated",model);
65 }
```

```
66 @RequestMapping("")
67 public String list(ModelMap model) {
68     //gọi hàm findAll() trong service
69     List<CategoryEntity> list = categoryService.findAll();
70     //chuyển dữ liệu từ list lên biến categories
71     model.addAttribute("categories",list);
72     return "admin/categories/list";
73 }
74 @GetMapping("edit/{categoryId}")
75 public ModelAndView edit(ModelMap model,@PathVariable("categoryId") Long categoryId) {
76     Optional<CategoryEntity> optCategory = categoryService.findById(categoryId);
77     CategoryModel cateModel = new CategoryModel();
78     //kiểm tra sự tồn tại của category
79     if(optCategory.isPresent()) {
80         CategoryEntity entity = optCategory.get();
81         //copy từ entity sang cateModel
82         BeanUtils.copyProperties(entity, cateModel);
83         cateModel.setIsEdit(true);
84         //đẩy dữ liệu ra view
85         model.addAttribute("category",cateModel);
86
87         return new ModelAndView("admin/categories/addOrEdit",model);
88     }
89     model.addAttribute("message","Category is not existed!!!!");
90     return new ModelAndView("forward:/admin/categories",model);
91 }
```

□ Bước 6: Tầng Controller nhận các request từ người dùng và chuyển tiếp cho tầng Service xử lý.

```
92 @GetMapping("delete/{categoryId}")
93 public ModelAndView delet(ModelMap model, @PathVariable("categoryId") Long categoryId) {
94     categoryService.deleteById(categoryId);
95     model.addAttribute("message", "Category is deleted!!!!");
96     return new ModelAndView("forward:/admin/categories/searchpaginated", model);
97 }
98 @GetMapping("search")
99 public String search(ModelMap model, @RequestParam(name="name", required = false) String name) {
100     List<CategoryEntity> list = null;
101     //có nội dung truyền về không, name là tùy chọn khi required=false
102     if(StringUtils.hasText(name)) {
103         list = categoryService.findByNameContaining(name);
104     } else {
105         list = categoryService.findAll();
106     }
107     model.addAttribute("categories", list);
108     return "admin/categories/search";
109 }
```

```
110 @RequestMapping("searchpaginated")
111 public String search(ModelMap model,
112     @RequestParam(name="name", required = false) String name,
113     @RequestParam("page") Optional<Integer> page,
114     @RequestParam("size") Optional<Integer> size) {
115     int count = (int) categoryService.count();
116     int currentPage = page.orElse(1);
117     int pageSize = size.orElse(3);
118     Pageable pageable = PageRequest.of(currentPage-1, pageSize, Sort.by("name"));
119     Page<CategoryEntity> resultPage = null;
120     if(StringUtils.hasText(name)) { resultPage = categoryService.findByNameContaining(name, pageable);
121         model.addAttribute("name", name);
122     } else { resultPage = categoryService.findAll(pageable);
123     }
124     int totalPages = resultPage.getTotalPages();
125     if(totalPages > 0) {
126         int start = Math.max(1, currentPage-2);
127         int end = Math.min(currentPage + 2, totalPages);
128         if(totalPages > count) {
129             if(end == totalPages) start = end - count;
130             else if (start == 1) end = start + count;
131         }
132         List<Integer> pageNumbers = IntStream.rangeClosed(start, end)
133             .boxed().collect(Collectors.toList());
134         model.addAttribute("pageNumbers", pageNumbers);
135     } model.addAttribute("categoryPage", resultPage);
136     return "admin/categories/searchpaginated";
137 }
138 }
```

- **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang layout_admin.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    . . .
    <body>
        <header class="row">
            <div class="col">
                <div th:replace="~{admin/fragments/header::header}"></div>
            </div>
        </header>
        <main class="container">
            <!-- <nav class="row">
                <div class="col">
                    <div th:replace="~{admin/fragments/nav::nav}"></div>
                </div>
            </nav> -->
            <section class="row mt-4">
                <div class="col mt-5 mb-5">
                    <div layout:fragment="content"></div>
                </div>
            </section>
        </main>
        <footer class="row">
            <div class="col">
                <div th:replace="~{admin/fragments/footer::footer}"></div>
            </div>
        </footer>
    </body>
</html>
```

- **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang list.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{admin/layout-admin.html}">

    <section class="row" layout:fragment="content">
        <div class="col mt-4">
            <div class="card">
                <div class="card-header">List Category</div>
                <div class="card-body">
                    <!-- Hiển thông báo -->
                    <div th:if="@{message != null}" class="alert alert-primary" role="alert">
                        <i>[[${message}]]</i>
                    </div>
                    <!-- Hết thông báo -->
                    <table class="table table-striped table-responsive">
                        <thead class="thead-inverse">
                            <tr>
                                <th>Category ID</th>
                                <th>Category Name</th>
                                <th>Action</th>
                            </tr>
                        </thead>
                        <tbody>
                            <tr th:each="category: ${categories}">
                                <td scope="row">[[${category.categoryId}]]</td>
                                <td th:text="${category.name}"></td>
                                <td>
                                    <a th:href="@{'/admin/categories/view/'+${category.categoryId}}" class="btn btn-outline-info"><i
                                        class="fa fa-info"></i></a>
                                    <a th:href="@{'/admin/categories/edit/'+ ${category.categoryId}}"
                                        class="btn btn-outline-warning"><i class="fa fa-edit"></i></a>
                                    <a th:href="@{'/admin/categories/delete/'+ ${category.categoryId}}" class="btn btn-outline-danger"><i
                                        class="fa fa-trash"></i></a></td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </section>
```

□ **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang AddOrEdit.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{admin/layout-admin.html}">

<section class="row" layout:fragment="content">
  <div class="col-6 offset-3 mt-4">
    <form th:action="@{/admin/categories/saveOrUpdate}" method="POST" th:object="${category}">
      <div class="card">
        <div class="card-header">
          <h2 th:text="${category.isEdit ? 'Edit Category' : 'Add New Category'}">Add New Category</h2>
        </div>
        <div class="card-body">
          <div class="mb-3" th:if="${category.isEdit}">
            <label for="categoryId" class="form-label">Category ID:</label>
            <input type="hidden" th:field="*{isEdit}">
            <input
              type="text" readonly="readonly" class="form-control" th:field="*{categoryId}" id="categoryId" name="categoryId"
              aria-describedby="categoryIdid" placeholder="Category Id">
            <div th:if="${#fields.hasErrors('categoryId')}"
              id="categoryIdid" class="form-text text-muted"><span class="text-danger">Category ID is required!</span></div>
          </div>
          <div class="mb-3">
            <label for="name" class="form-label">Category Name:</label>
            <input
              type="text" class="form-control" th:field="*{name}" id="name" name="name"
              aria-describedby="nameid" placeholder="Category Name">
            <div th:if="${#fields.hasErrors('name')}" id="nameidid" class="form-text text-muted"><span class="text-danger">Category
          </div>
        </div>
      </div>
    </form>
  </div>
</div>
```

- **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang searchpaging.html

```
<!doctype html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{admin/layout-admin.html}">
    <section class="row" layout:fragment="content">
        <div class="col mt-4">
            <div class="card">
                <div class="card-header">List Category</div>
                <div class="card-body">
                    <!-- Hiển thông báo -->
                    <div th:if="${message != null}" class="alert alert-primary"
                        role="alert">
                        <i>[[${message}]]</i>
                    </div>
                    <!-- Hết thông báo -->
                    <div class="row mt-2 mb-2">
                        <div class="col-md-6">
                            <form th:action="@{/admin/categories/searchpaginated}">
                                <div class="input-group">
                                    <input type="text" class="form-control ml-2" name="name"
                                        id="name" placeholder="Nhập từ khóa để tìm" />
                                    <button class="btn btn-outline-primary ml-2">Search</button>
                                </div>
                            </form>
                        </div>
                        <div class="col-md-6">
                            <div class="float-right">
                                <a class="btn btn-outline-success"
                                    th:href="@{/admin/categories/add}">Add New Category</a>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
```

❑ **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang searchpaging.html

```
<div class="row" th:if="${!categoryPage.hasContent()}">
  <div class="col">
    <div class="alert alert-danger" role="alert">No Category
      Found</div>
  </div>
</div>
<table th:if="${categoryPage.hasContent()}"
  class="table table-striped table-inverse table-responsive">
  <thead class="thead-inverse">
    <tr>
      <th>Category ID</th>
      <th>Category Name</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="category, iStat : ${categoryPage.content}">
      <td scope="row">[[${category.categoryId}]]</td>
      <td th:text="${category.name}"></td>
      <td><a
        th:href="@{'/admin/categories/view/' + ${category.categoryId}}"
        class="btn btn-outline-info"><i class="fas fa-info"></i></a> <a
        th:href="@{'/admin/categories/edit/' + ${category.categoryId}}"
        class="btn btn-outline-warning"><i class="fas fa-edit"></i></a>
        <a th:data-id="${category.categoryId}"
        th:data-name="${category.name}"
        onclick="showconfirmation(this.getAttribute('data-id'),this.getAttribute('data-name'))"
        class="btn btn-outline-danger"><i class="fa fa-trash"></i></a>
      </td>
    </tr>
  </tbody>
</table>
```

```
</tbody>
</table>
<script type="text/javascript">
  function showconfirmation(id,name){
    $('#productName').text(name);
    $('#yesOption').attr('href','/admin/categories/delete/' + id);
    $('#confirmationId').modal('show');
  }
</script>
<!-- Modal -->
<div class="modal fade" id="confirmationId" tabindex="-1"
  aria-labelledby="confirmationLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="confirmationLabel">Confirmation</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
          aria-label="Close"></button>
      </div>
      <div class="modal-body">
        Bạn có muốn xóa "<span id="productName"></span>"?
      </div>
      <div class="modal-footer">
        <a id="yesOption" class="btn btn-primary">Yes</a>
        <button type="button" class="btn btn-secondary"
          data-bs-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

□ **Bước 7:** Sử dụng Thymeleaf Layout làm Template Engine để xây dựng các trang web. Trang searchpaging.html

```
<div class="card-footer text-muted">
  <div class="row">
    <div class="col-3">
      <form action="">
        <div class="mb-3 input-group float-left">
          <label for="size" class="mr-2">Page size:</label>
          <select class="form-select ml-2"
            name="size" aria-label="size" id="size"
            onchange="this.form.submit()">
            <option th:selected="${categoryPage.size == 3}" value="3">3</option>
            <option th:selected="${categoryPage.size == 5}" value="5">5</option>
            <option th:selected="${categoryPage.size == 10}" value="10">10</option>
            <option th:selected="${categoryPage.size == 15}" value="15">15</option>
            <option th:selected="${categoryPage.size == 20}" value="20">20</option>
          </select>
        </div>
      </form>
    </div>
  </div>
```

```
<div class="col-7">
  <div aria-label="Page navigation"
    if="${categoryPage.totalPages > 0}"
    class="pagination justify-content-center">
    <li>
      th:class="${1 == categoryPage.number + 1} ? 'page-item active' : 'page-item'"
      <a
        th:href="@{/admin/categories/searchpaginated(name=${name},size=${categoryPage.size},page=${1})}"
        class="page-link">First</a>
      </li>
    <li class="page-item active">
      th:each="pageNumber : ${pageNumbers}"
      th:if="${categoryPage.totalPages > 1}"
      th:class="${pageNumber == categoryPage.number + 1} ? 'page-item active' : 'page-item'"
      <a
        th:href="@{/admin/categories/searchpaginated(name=${name},size=${categoryPage.size},page=${pageNumber})}"
        class="page-link" th:text="${pageNumber}"></a>
      </li>
    <li>
      th:class="${categoryPage.totalPages == categoryPage.number + 1} ? 'page-item active' : 'page-item'"
      <a
        th:href="@{/admin/categories/searchpaginated(name=${name},size=${categoryPage.size},page=${categoryPage.totalPages})}"
        class="page-link">Last</a>
      </li>
    </ul>
  </div>
</div>
```

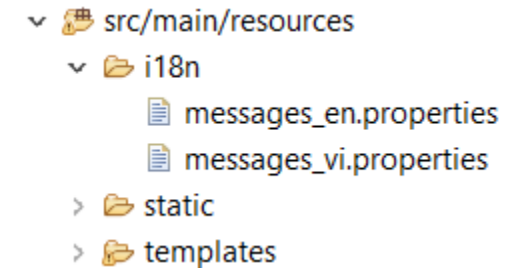
- **Bước 8:** Sử dụng i18n (Internationalization) để hỗ trợ nhiều ngôn ngữ trong ứng dụng. Các message được định nghĩa trong các file properties cho từng ngôn ngữ. Dưới đây là ví dụ về file message tiếng Việt và tiếng Anh: **src/main/resources/ i18n/messages_vi.properties.**

```
iotstar.message.hello=Chào mừng đến với TMDT  
iotstar.message.success=Thêm danh mục thành công!  
iotstar.message.failed=Thêm danh mục thất bại!  
iotstar.value.addCate=Thêm danh mục  
iotstar.value.viewListCate=Xem danh sách danh mục  
iotstar.value.listCate=Danh sách danh mục
```

src/main/resources/ i18n/messages_en.properties

```
iotstar.message.hello=Welcome to TMDTApp  
iotstar.message.success=Add Category Successfully!  
iotstar.message.failed=Add Category Failed!  
iotstar.value.addCate=Add Category  
iotstar.value.viewListCate=View Category list  
iotstar.value.listCate=Category list
```

□ Bước 8: Tạo class cấu hình i18n



```
<span
th:text="#{iotstar.message.hello}"></span>
<a href="?language=en">English</a>|
<a href="?language=vi_VN">Viet Nam</a>|
```

```
1 package vn.iostar.Config;
2 import java.util.Locale;
3 import org.springframework.context.MessageSource;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.context.support.ReloadableResourceBundleMessageSource;
7 import org.springframework.web.servlet.LocaleResolver;
8 import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
9 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
10 import org.springframework.web.servlet.i18n.CookieLocaleResolver;
11 import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
12 @Configuration
13 public class WebMvcConfig implements WebMvcConfigurer {
14     @Bean(name = "localeResolver")
15     public LocaleResolver getLocaleResolver() {
16         CookieLocaleResolver resolver = new CookieLocaleResolver();
17         resolver.setCookieDomain("iotstar.vn");
18         resolver.setDefaultLocale(Locale.ENGLISH);
19         return resolver;
20     }
21     @Bean(name = "messageSource")
22     public MessageSource getMessageResource() {
23         ReloadableResourceBundleMessageSource messageResource = new ReloadableResourceBundleMessageSource();
24         // Đọc vào file src/main/resources/i18n/messages_XXX.properties
25         // Ví dụ: i18n/messages_en.properties
26         messageResource.setBasename("classpath:i18n/messages");
27         messageResource.setDefaultEncoding("UTF-8");
28         return messageResource;
29     }
30     @Override
31     public void addInterceptors(InterceptorRegistry registry) {
32         LocaleChangeInterceptor localeInterceptor = new LocaleChangeInterceptor();
33         localeInterceptor.setParamName("language");
34         registry.addInterceptor(localeInterceptor).addPathPatterns("/");
35     }
36 }
```