

TRƯỜNG ĐẠI HỌC HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN



TÀI LIỆU THỰC HÀNH
SPRING BOOT

**CHƯƠNG 05: XÂY DỰNG CÁC CHỨC NĂNG HIỂN
THỊ/THÊM/XÓA/SỬA**

Giảng viên biên soạn: Ths. Nguyễn Minh Tân

2024

BUỔI 5: XÂY DỰNG CÁC CHỨC NĂNG HIỂN THỊ/THÊM/XÓA/SỬA

I. MỤC TIÊU

Sau khi học xong bài này, học viên có thể nắm được:

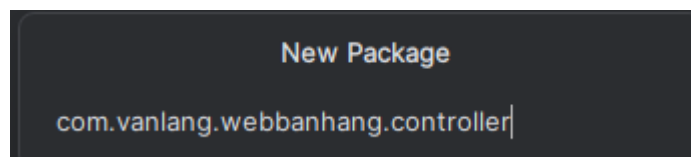
- Hiểu rõ nguyên lý hoạt động của mô hình MVC và vai trò của Controllers trong kiến trúc này.
- Có kỹ năng thiết kế và cài đặt Controllers để xử lý các yêu cầu từ người dùng (HTTP requests).
- Học cách sử dụng annotations để định nghĩa các routes và actions trong Controller.
- Hiểu cách truyền dữ liệu từ Controller đến View sử dụng Model hoặc ModelAndView.
- Biết cách nhận và xử lý dữ liệu đầu vào từ người dùng thông qua form submissions.
- Phát triển kỹ năng xử lý các tình huống lỗi trong ứng dụng web.
- Biết cách gửi phản hồi thích hợp đến người dùng thông qua HTTP status codes và messages.
- Học cách tích hợp Controllers với các Services để xử lý nghiệp vụ như xác thực người dùng, quản lý giỏ hàng, và thực hiện thanh toán.
- Hiểu được cách sử dụng Dependency Injection để quản lý các phụ thuộc trong ứng dụng.

II. NỘI DUNG THỰC HÀNH:

5.1 Tạo các controller class cho các chức năng CRUD và quản lý người dùng

Để xử lý các yêu cầu đến từ người dùng, bạn cần tạo các controller class trong Spring Boot:

- **Khởi tạo Package:** Tạo một package mới trong dự án của bạn và đặt tên là controller.



- **Tạo Controller Classes:** Trong package controller, tạo các class như ``ProductController``, ``CategoryController``, và ``OrderController``.
- **Xác Định Các Phương Thức Xử Lý:** Định nghĩa các phương thức trong mỗi class để xử lý các chức năng tương ứng.

Ví dụ, `ProductController` có thể bao gồm các phương thức như `listProducts()`, `showProductForm()`, `saveProduct()`, `showUpdateForm()`, và `deleteProduct()` để quản lý sản phẩm.

5.2 Xác định các route và phương thức xử lý yêu cầu

Mỗi controller class cần các annotation để xác định cách nó xử lý các yêu cầu:

- **Sử Dụng @Controller:** Áp dụng annotation này trên mỗi class để đánh dấu nó như một phần của bộ điều khiển giao diện người dùng.
- **Định Nghĩa Route Với @RequestMapping:** Sử dụng annotation này trên mỗi phương thức để chỉ định đường dẫn mà phương thức đó xử lý. Nếu không chỉ định, route mặc định sẽ là `/`.
- **Xác Định Phương Thức HTTP:** Sử dụng các annotation như `@GetMapping` cho các yêu cầu GET, `@PostMapping` cho POST, và các annotation như `@ModelAttribute`, `@PathVariable` để xử lý dữ liệu đầu vào.

Ví dụ, trong class `ProductController`, bạn có thể sử dụng annotation như sau:

Ảnh minh họa

5.3 Gọi phương thức từ service class và truyền dữ liệu qua model

Trong mỗi phương thức xử lý yêu cầu của controller, việc tương tác với các service class là cần thiết để xử lý logic nghiệp vụ và truy xuất dữ liệu. Sau đó, dữ liệu được truyền đến view thông qua model để hiển thị cho người dùng.

Ví dụ về Phương Thức Xử Lý:

Trong `ProductController`, phương thức `listProducts()` là một ví dụ điển hình:

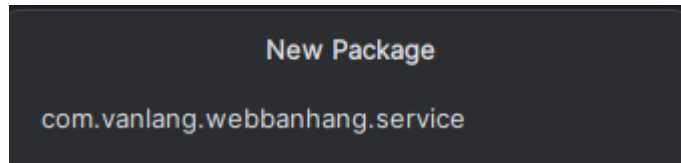
- **Gọi Service Class:** Phương thức này sẽ gọi `getAllProducts()` từ class service tương ứng để lấy danh sách tất cả sản phẩm.
- **Truyền Dữ Liệu Qua Model:** Danh sách sản phẩm thu được sẽ được thêm vào model, để có thể truyền dữ liệu này vào view.

5.4 Hướng dẫn code mẫu

5.4.1 Xây dựng chức năng Hiển thị/Thêm Category

Bước 1: Tạo class CategoryService.java

Tạo một package **service** trong thư mục **src/main/java/com.vanlang.webbanhang**



Tiếp theo, tạo một file mới có tên **CategoryService.java** trong package **src/main/java/com.vanlang.webbanhang /service**

Trong lớp **CategoryService.java**, định nghĩa các phương thức để thao tác với cơ sở dữ liệu, bao gồm thêm, sửa, xóa và truy vấn dữ liệu. Lớp này sẽ cung cấp các chức năng để quản lý danh mục sản phẩm và đảm bảo tính nhất quán của dữ liệu trong toàn bộ ứng dụng web.

Dưới đây là cách viết code cho lớp **CategoryService.java**:

```
package com.vanlang.webbanhang.service;

import com.vanlang.webbanhang.model.Category;
import com.vanlang.webbanhang.repository.CategoryRepository;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

@Service
@RequiredArgsConstructor
@Transactional
public class CategoryService {
    private final CategoryRepository categoryRepository;

    /**
     * Retrieve all categories from the database.
     * @return a list of categories
     */
    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }
}
```

```
    }  
    /**  
     * Retrieve a category by its id.  
     * @param id the id of the category to retrieve  
     * @return an Optional containing the found category or  
     empty if not found  
     */  
    public Optional<Category> getCategoryById(Long id) {  
        return categoryRepository.findById(id);  
    }  
    /** * Add a new category to the database. * @param category  
    the category to add */  
    public void addCategory(Category category) {  
        categoryRepository.save(category);  
    }  
    /** * Update an existing category. * @param category the  
    category with updated information */  
    public void updateCategory(@NonNull Category category) {  
        Category existingCategory =  
categoryRepository.findById(category.getId()).orElseThrow(() ->  
new IllegalStateException("Category with ID " +  
category.getId() + " does not exist."));  
        existingCategory.setName(category.getName());  
        categoryRepository.save(existingCategory);  
    }  
    /** *Delete a category by its id. * @param id the id of the  
    category to delete */  
    public void deleteCategory(Long id) {  
        if(!categoryRepository.existsById(id)) {  
            throw new IllegalStateException("Category with ID " +  
id + " does not exist.");  
        }  
        categoryRepository.deleteById(id);  
    }  
}
```

Bước 2: Tạo class CategoryController.java

Trong đường dẫn `src/main/java/com.vanlang.webbanhang/controller` tạo một class ``CategoryController.java``.

```
package com.vanlang.webbanhang.controller;  
  
import com.vanlang.webbanhang.model.Category;  
import com.vanlang.webbanhang.service.CategoryService;  
import jakarta.validation.Valid;  
import lombok.RequiredArgsConstructor;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller
@RequiredArgsConstructor
public class CategoryController {
    @Autowired
    private final CategoryService categoryService;

    @GetMapping("/categories/add")
    public String showAddForm(Model model) {
        model.addAttribute("category", new Category());
        return "/categories/add-category";
    }

    @PostMapping("/categories/add")
    public String addCategory(@Valid Category category, BindingResult
bindingResult, Model model) {
        if (bindingResult.hasErrors()) {
            return "/categories/add-category";
        }
        categoryService.addCategory(category);
        return "redirect:/categories";
    }

    // Hiện thị danh sách danh mục
    @GetMapping("/categories")
    public String listCategories(Model model) {
        List<Category> categories = categoryService.getAllCategories();
        model.addAttribute("categories", categories);
        return "/categories/categories-list";
    }
}
```

Bước 4: Tạo view cho category

Sử dụng lại trang layout.html của bài trước. Code mẫu:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    <meta charset="UTF-8">
    <title>Layout</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boo
tstrap.min.css">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-
primary">
```

```
<div class="container-fluid"><a class="navbar-brand"
href="#">Trang Chủ</a>
    <div class="collapse navbar-collapse"
id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item"><a class="nav-link
active" aria-current="page" href="/sinhvien">Web Bán Hàng</a>
            </li>
        </ul>
    </div>
</div>
</nav>
<div class="container mt-4">
    <section layout:fragment="content"> &lt;!&dash;
Content will be replaced by each specific page &dash;&gt;
    </section>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/boots
trap.bundle.min.js"></script>
</body>
</html>
```

Trong đường dẫn `src/main/resources/templates` tạo thêm directory **categories**. Sau đó tạo thêm 2 file html: ``add-category.html`` và ``categories-list.html``

File ``add-category.html``:

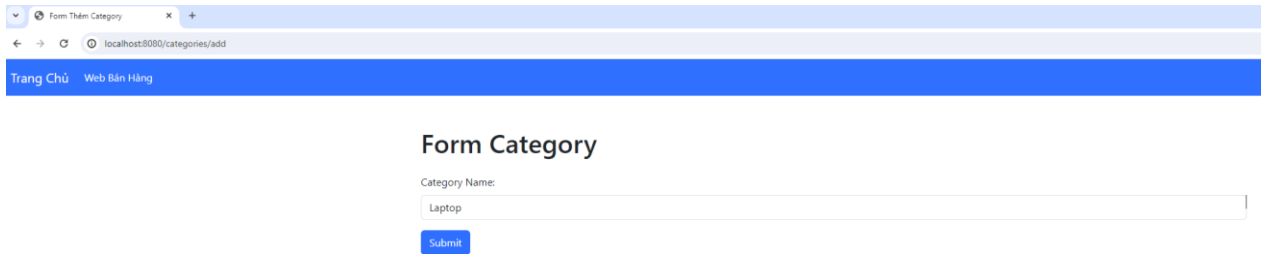
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="~{layout}">
<head><title>Form Thêm Category</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boo
tstrap.min.css">
</head>
<body>
<section layout:fragment="content" class="container mt-5"><h1
class="mb-4">Form Category</h1>
    <form th:action="@{/categories/add}"
th:object="${category}" method="post">
        <div class="mb-3">
            <label for="name" class="form-label">Category
Name:</label>
            <input type="text" id="name" th:field="*{name}"
class="form-control"/>
        </div>
    </form>
</section>
```

```
        <div class="text-danger"
th:if="{#fields.hasErrors('name')}" th:errors="*{name}"></div>
        </div>
        <button type="submit" class="btn btn-
primary">Submit</button>
    </form>
</section>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootst
rap.bundle.min.js"></script>
</body>
</html>
```

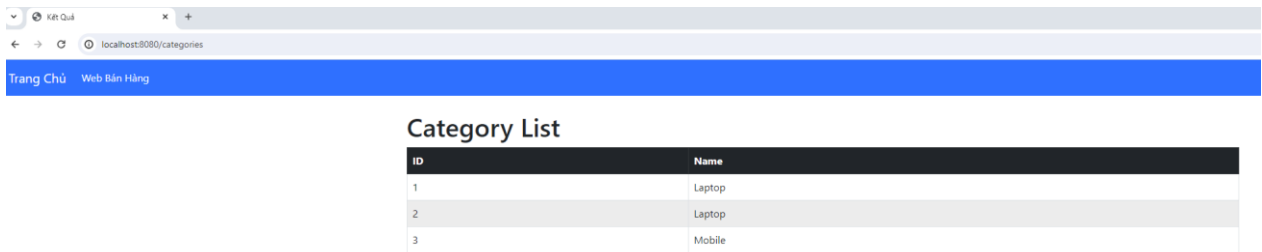
File `categories-list.html`:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="~{layout}">
<head><title th:text="{title} ? : 'Kết Quả'">Kết Quả</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boo
tstrap.min.css">
</head>
<body>
<section layout:fragment="content"><h1>Category List</h1>
    <table class="table table-bordered table-hover">
        <thead class="table-dark">
            <tr>
                <th>ID</th>
                <th>Name</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="category : {categories}">
                <td th:text="{category.id}"></td>
                <td th:text="{category.name}"></td>
            </tr>
        </tbody>
    </table>
</section>
</body>
</html>
```

Bước 5: Tiến hành build dự án và kiểm tra kết quả: <http://localhost:8080/categories/add>

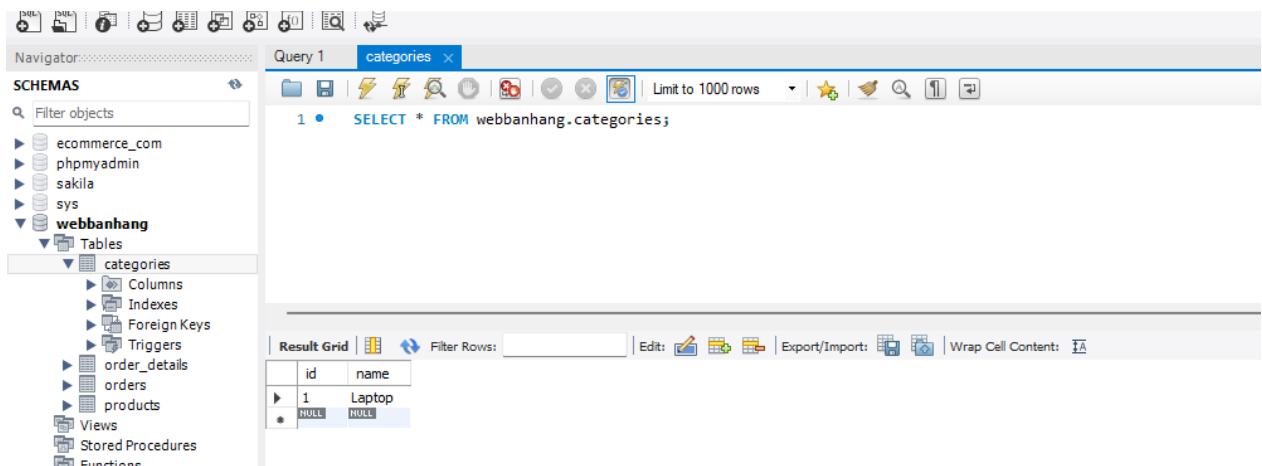


Thêm thành công và hiển thị danh sách category: <http://localhost:8080/categories>



| ID | Name |
|----|--------|
| 1 | Laptop |
| 2 | Laptop |
| 3 | Mobile |

- *Kiểm tra Database:*



| id | name |
|----|--------|
| 1 | Laptop |

- *Thêm thành công category “Laptop” vào database webbanhang:*

5.4.2 Chức năng xóa/ sửa cho Category

Xây dựng hoàn chỉnh chức năng SỬA/XÓA Category, thêm các view tương ứng sử dụng Bootstrap.

Code bổ sung cho **CategoryController.java**:

```
// GET request to show category edit form
@GetMapping("/categories/edit/{id}")
public String showUpdateForm(@PathVariable("id") Long id, Model
```

```
model) {
    Category category =
categoryService.getCategoryById(id).orElseThrow(() -> new
IllegalArgumentException("Invalid category Id:" + id));
    model.addAttribute("category", category);
    return "/categories/update-category";
}

// POST request to update category
@PostMapping("/categories/update/{id}")
public String updateCategory(@PathVariable("id") Long id,
@Valid Category category, BindingResult result, Model model) {
    if (result.hasErrors()) {
        category.setId(id);
        return "/categories/update-category";
    }
    categoryService.updateCategory(category);
    model.addAttribute("categories",
categoryService.getAllCategories());
    return "redirect:/categories";
}

// GET request for deleting category
@GetMapping("/categories/delete/{id}")
public String deleteCategory(@PathVariable("id") Long id, Model
model) {
    Category category =
categoryService.getCategoryById(id).orElseThrow(() -> new
IllegalArgumentException("Invalid category Id:" + id));
    categoryService.deleteCategory(id);
    model.addAttribute("categories",
categoryService.getAllCategories());
    return "redirect:/categories";
}
```

Cập nhật view **category-list**:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="~{layout}">
<head><title th:text="${title} ?: 'Categories List'">
Categories List </title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boo
tstrap.min.css">
</head>
<body>
```

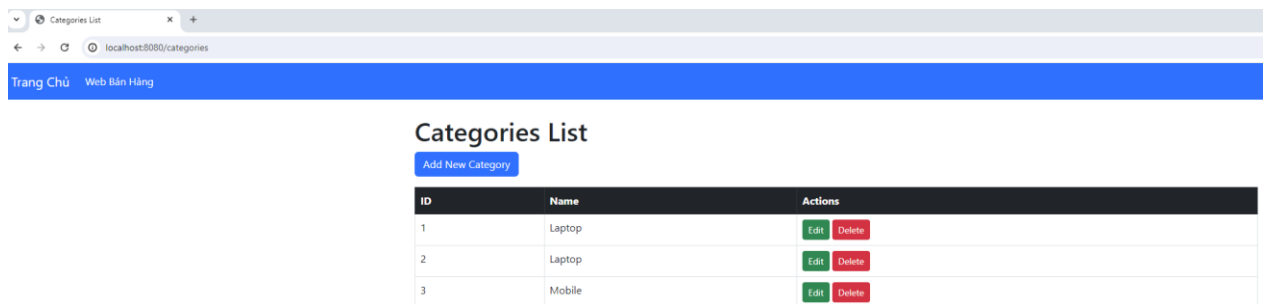
```
<section layout:fragment="content"><h1>Categories List</h1>
  <div><a th:href="@{/categories/add}" class="btn btn-primary
mb-3">Add New Category</a></div>
  <table class="table table-bordered table-hover">
    <thead class="table-dark">
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="category : ${categories}">
        <td th:text="${category.id}"></td>
        <td th:text="${category.name}"></td>
        <td><a
th:href="@{/categories/edit/{id} (id=${category.id})}"
class="btn btn-success btn-sm">Edit</a> <a
th:href="@{/categories/delete/{id} (id=${category.id})}"
class="btn btn-danger btn-sm"
        onclick="return confirm('Are you
sure?') ">Delete</a></td>
      </tr>
    </tbody>
  </table>
</section>
</body>
</html>
```

Code View ``update-category``:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
  layout:decorate="~{layout}">
<head>
  <title>Update Category</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boo
tstrap.min.css">
</head>
<body>
  <section layout:fragment="content" class="container mt-
5"><h1 class="mb-4">Update Category</h1>
    <form
th:action="@{/categories/update/{id} (id=${category.id})}"
th:object="${category}" method="post" class="needs-validation"
```

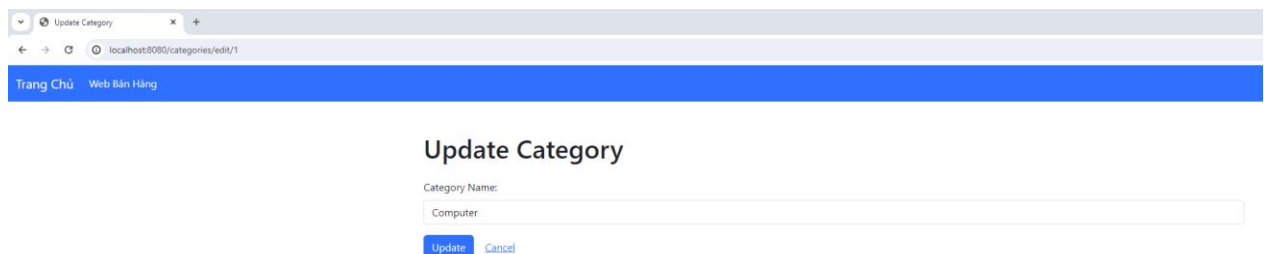
```
novalidate>
    <div class="mb-3">
      <label for="name" class="form-label">Category
Name:</label>
      <input type="text" th:field="*{name}"
class="form-control" id="name" required>
      <div class="invalid-feedback"
th:if="${#fields.hasErrors('name')}" th:errors="*{name}">Valid
name is required.</div>
    </div>
    <button type="submit" class="btn btn-
primary">Update</button>
    <a th:href="@{/categories}" class="btn btn-
link">Cancel</a>
  </form>
</section>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootst
rap.bundle.min.js"></script>
</body>
</html>
```

- Tiến hành build dự án và kiểm tra kết quả:
- Trang hiển thị danh sách category:

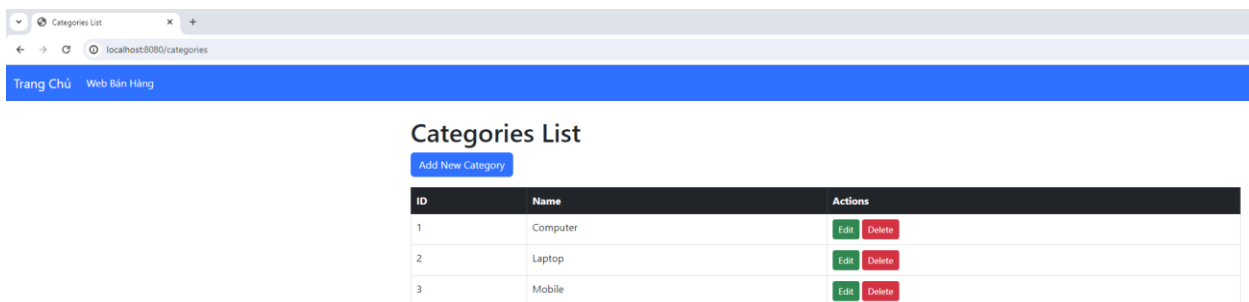


Chức năng Edit:

Nhấn nút Edit và thay category PC thành Computer:

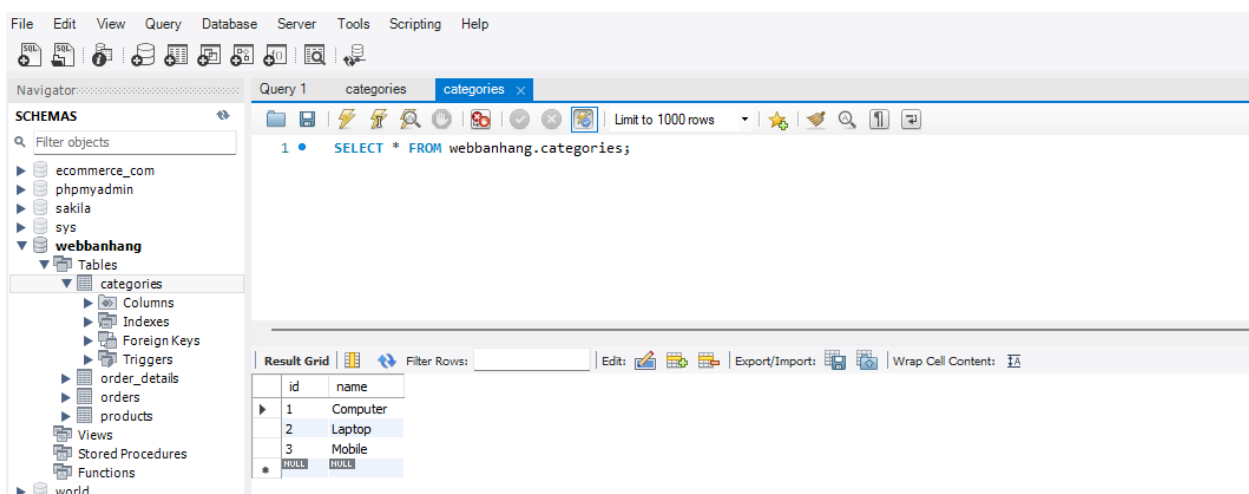


Trang hiển thị danh sách category sau khi đã sửa:



| ID | Name | Actions |
|----|----------|---|
| 1 | Computer | Edit Delete |
| 2 | Laptop | Edit Delete |
| 3 | Mobile | Edit Delete |

Kiểm tra Database cho thấy database đã cập nhật thành công



| id | name |
|------|----------|
| 1 | Computer |
| 2 | Laptop |
| 3 | Mobile |
| NULL | NULL |