# PHIẾU HỌC TẬP ASP.NET CORE [02] - TẠO API ĐƠN GIẢN (CATEGORIES MANAGEMENT)

## Họ và tên: Lê Sỹ Mạnh Cường

## Lớp: CNTT 18-09

## Video đã xem (Link):https://youtu.be/JUvsdFWL7WM?si=UNPzODD65SANIxWF

## ⏰ Thời gian hoàn thành: 25/12/2025

## 1. Kiến thức cốt lõi (Tổng quan tạo API)

*Sau khi xem video, hãy điền vào các chỗ trống dưới đây:*

### 1. Controller trong ASP.NET Core là gì?

* Controller là một **class** (class/interface/method) chứa **các phương thức** để xử lý HTTP requests.
* Tên Controller thường kết thúc bằng: **Controller**
* Ví dụ: ProductsController Controller quản lý products

### 2. Các HTTP Verbs cơ bản:

| **HTTP Verb** | **Mục đích** | **Ví dụ** |
| --- | --- | --- |
| **GET** | Lấy dữ liệu (đọc) | GET /api/products |
| **POST** | Tạo mới dữ liệu | POST /api/products |
| **PUT** | Cập nhật toàn bộ dữ liệu | PUT /api/products/1 |
| **DELETE** | Xóa dữ liệu | DELETE /api/products/1 |

### 3. Route trong ASP.NET Core:

Hãy nhìn vào ví dụ sau:

[ApiController]

[Route("api/[controller]")]

public class ProductsController : ControllerBase

{

[HttpGet]

public ActionResult GetAll() { ... }

[HttpGet("{id}")]

public ActionResult GetById(int id) { ... }

}

* [Route("api/[controller]")] có nghĩa là: **định nghĩa đường dẫn (URL) cho controller** bằng *Attribute Routing*.
* Placeholder [controller] sẽ được thay thế bằng: Tên controller không bao gồm phần "Controller"
* Kết quả route cho GetAll() sẽ là: /api/products (ví dụ: /api/...)
* Kết quả route cho GetById() sẽ là: /api/products/{id} (ví dụ: /api/.../...)

### 4. ActionResult vs Task:

* ActionResult<List<Product>> dùng cho: Synchronous Task<ActionResult<List<Product>>> dùng cho: Asynchronous
* Trong ASP.NET Core hiện đại, nên dùng: Task<ActionResult<T>> (Async) vì cải thiện hiệu suất, có thể xử lý nhiều request đồng thời, không block thread

### 5. HTTP Status Codes:

| **Status Code** | **Ý Nghĩa** | **Khi nào dùng** |
| --- | --- | --- |
| **200 OK** | Yêu cầu thành công | Khi lấy dữ liệu thành công (GET) |
| **201 Created** | Tạo mới thành công | Khi tạo mới resource thành công (POST) |
| **204 No Content** | Thành công nhưng không có nội dung trả về | Khi xóa thành công (DELETE) |
| **404 Not Found** | Không tìm thấy resource | Khi resource không tồn tại |
| **400 Bad Request** | Yêu cầu không hợp lệ | Khi client gửi dữ liệu không hợp lệ |

### 6. Service Pattern là gì?

* Service Pattern giúp bạn: Tách biệt business logic khỏi controller, dễ dàng quản lý, bảo trì và test code
* Tại sao cần tách Service khỏi Controller?
  + Tăng tính tái sử dụng code
  + Dễ dàng viết unit test
  + Tuân thủ nguyên tắc Single Responsibility Principle (SRP)

## 2. Ví dụ thực hành (Tạo API Categories)

*Yêu cầu: Thực hiện đầy đủ các TODO dưới đây. Hãy viết code (không copy-paste), không dùng AI hoặc nhờ bạn khác viết!*

### 📝 TODO 1: Tạo Model Category.cs

**Yêu cầu:** Tạo class Category với các properties:

* Id (int)
* Name (string)
* Description (string)
* IsActive (bool)
* CreatedAt (DateTime)

**Hãy viết code dưới đây:**

// File: Category.cs

// (Viết class Category của bạn ở đây)

**Chứng minh:** (Hãy chụp ảnh hoặc paste đoạn code đã viết)

namespace Categories.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public string Description { get; set; } = string.Empty;

    public bool IsActive { get; set; } = true;

    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

}

### 📝 TODO 2: Tạo Interface ICategoryService

**Yêu cầu:** Tạo interface có 5 methods:

* GetAllCategories() → trả về List<Category>
* GetCategoryById(int id) → trả về Category?
* CreateCategory(Category category) → trả về Category
* UpdateCategory(int id, Category category) → trả về Category?
* DeleteCategory(int id) → trả về bool

**Hãy viết code dưới đây:**

// File: Services/ICategoryService.cs

// (Viết interface của bạn ở đây)

**Chứng minh:** (Hãy chụp ảnh hoặc paste đoạn code đã viết)

using Categories.Models;

namespace Categories.Services;

public interface ICategoryService

{

    List<Category> GetAllCategories();

    Category? GetCategoryById(int id);

    Category CreateCategory(Category category);

    Category? UpdateCategory(int id, Category category);

    bool DeleteCategory(int id);

}

### 📝 TODO 3: Tạo CategoryService Implementation

**Yêu cầu:** Tạo class CategoryService : ICategoryService với:

* In-memory Listchứa 3 categories mẫu (Electronics, Books, Clothing)
* Implement tất cả 5 methods

**Gợi ý:**

* Cho GetAllCategories(): return danh sách categories
* Cho GetCategoryById(): dùng FirstOrDefault() để tìm category
* Cho CreateCategory(): tạo ID mới = Max Id + 1
* Cho UpdateCategory(): tìm category, update properties, return
* Cho DeleteCategory(): xóa category khỏi list

**Hãy viết code dưới đây:**

// File: Services/CategoryService.cs

// (Viết implementation của bạn ở đây)

**Chứng minh:** (Hãy chụp ảnh hoặc paste đoạn code đã viết)

using Categories.Models;

namespace Categories.Services;

public class CategoryService : ICategoryService

{

    private readonly List<Category> \_categories = new()

    {

        new Category { Id = 1, Name = "Electronics", Description = "Electronic devices and gadgets", IsActive = true, CreatedAt = DateTime.UtcNow.AddDays(-10) },

        new Category { Id = 2, Name = "Books", Description = "All kinds of books", IsActive = true, CreatedAt = DateTime.UtcNow.AddDays(-5) },

        new Category { Id = 3, Name = "Clothing", Description = "Fashion and clothing items", IsActive = true, CreatedAt = DateTime.UtcNow.AddDays(-2) }

    };

    public List<Category> GetAllCategories() => \_categories;

    public Category? GetCategoryById(int id) => \_categories.FirstOrDefault(c => c.Id == id);

    public Category CreateCategory(Category category)

    {

        category.Id = \_categories.Count > 0 ? \_categories.Max(c => c.Id) + 1 : 1;

        category.CreatedAt = DateTime.UtcNow;

        \_categories.Add(category);

        return category;

    }

    public Category? UpdateCategory(int id, Category category)

    {

        var existingCategory = GetCategoryById(id);

        if (existingCategory == null) return null;

        existingCategory.Name = category.Name;

        existingCategory.Description = category.Description;

        existingCategory.IsActive = category.IsActive;

        return existingCategory;

    }

    public bool DeleteCategory(int id)

    {

        var category = GetCategoryById(id);

        if (category == null) return false;

        return \_categories.Remove(category);

    }

}

### 📝 TODO 4: Tạo CategoriesController

**Yêu cầu:** Tạo Controller với 5 endpoints:

#### Endpoint 1: GET /api/categories

* HTTP Method: GET
* Trả về: ActionResult<List>
* Hàm trả về danh sách tất cả categories

#### Endpoint 2: GET /api/categories/{id}

* HTTP Method: GET
* Parameter: id (int)
* Trả về: ActionResult
* Nếu không tìm thấy → return NotFound()

#### Endpoint 3: POST /api/categories

* HTTP Method: POST
* Body: Category object
* Trả về: ActionResult
* Trả về HTTP 201 (CreatedAtAction)

#### Endpoint 4: PUT /api/categories/{id}

* HTTP Method: PUT
* Parameter: id (int)
* Body: Category object
* Trả về: ActionResult
* Nếu không tìm thấy → return NotFound()

#### Endpoint 5: DELETE /api/categories/{id}

* HTTP Method: DELETE
* Parameter: id (int)
* Trả về: IActionResult
* Thành công → return NoContent() (204)
* Không tìm thấy → return NotFound()

**Hãy viết code dưới đây:**

// File: Controllers/CategoriesController.cs

// (Viết Controller của bạn ở đây)

**Chứng minh:** (Hãy chụp ảnh hoặc paste đoạn code đã viết)

using Microsoft.AspNetCore.Mvc;

using Categories.Models;

using Categories.Services;

[ApiController]

[Route("api/[controller]")]

public class CategoriesController : ControllerBase

{

    private readonly ICategoryService \_categoryService;

    public CategoriesController(ICategoryService categoryService)

    {

        \_categoryService = categoryService;

    }

    // GET: api/categories

    [HttpGet]

    public ActionResult<List<Category>> GetAll()

    {

        var categories = \_categoryService.GetAllCategories();

        return Ok(categories);

    }

    // GET: api/categories/{id}

    [HttpGet("{id}")]

    public ActionResult<Category> GetById(int id)

    {

        var category = \_categoryService.GetCategoryById(id);

        if (category == null) return NotFound();

        return Ok(category);

    }

    // POST: api/categories

    [HttpPost]

    public ActionResult<Category> Create(Category category)

    {

        var createdCategory = \_categoryService.CreateCategory(category);

        return CreatedAtAction(nameof(GetById), new { id = createdCategory.Id }, createdCategory);

    }

    // PUT: api/categories/{id}

    [HttpPut("{id}")]

    public ActionResult<Category> Update(int id, Category category)

    {

        var updatedCategory = \_categoryService.UpdateCategory(id, category);

        if (updatedCategory == null) return NotFound();

        return Ok(updatedCategory);

    }

    // DELETE: api/categories/{id}

    [HttpDelete("{id}")]

    public IActionResult Delete(int id)

    {

        var result = \_categoryService.DeleteCategory(id);

        if (!result) return NotFound();

        return NoContent();

    }

}

### 📝 TODO 5: Đăng ký Service trong Program.cs

**Yêu cầu:** Sửa file Program.cs để:

1. Import namespace using CategoryAPI.Services;
2. Thêm dòng builder.Services.AddScoped<ICategoryService, CategoryService>(); đúng vị trí
3. Đảm bảo app.MapControllers(); được gọi

**Hãy viết code dưới đây (chỉ cần phần AddScoped):**

// File: Program.cs

// (Thêm dòng đăng ký service)

builder.Services.AddScoped<ICategoryService, CategoryService>();

// (Các dòng khác của Program.cs)

**Chứng minh:** (Hãy paste toàn bộ file Program.cs sau khi sửa)

using Categories.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Đăng ký CategoryService

builder.Services.AddScoped<ICategoryService, CategoryService>();

var app = builder.Build();

// Configure the HTTP request pipeline

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

## 3. Bài tập thực hành (Chạy & Test API)

### 🏃 Bước 1: Chạy ứng dụng

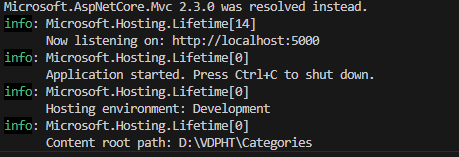
**TODO:** Chạy lệnh dưới đây trong Terminal/PowerShell:

dotnet run

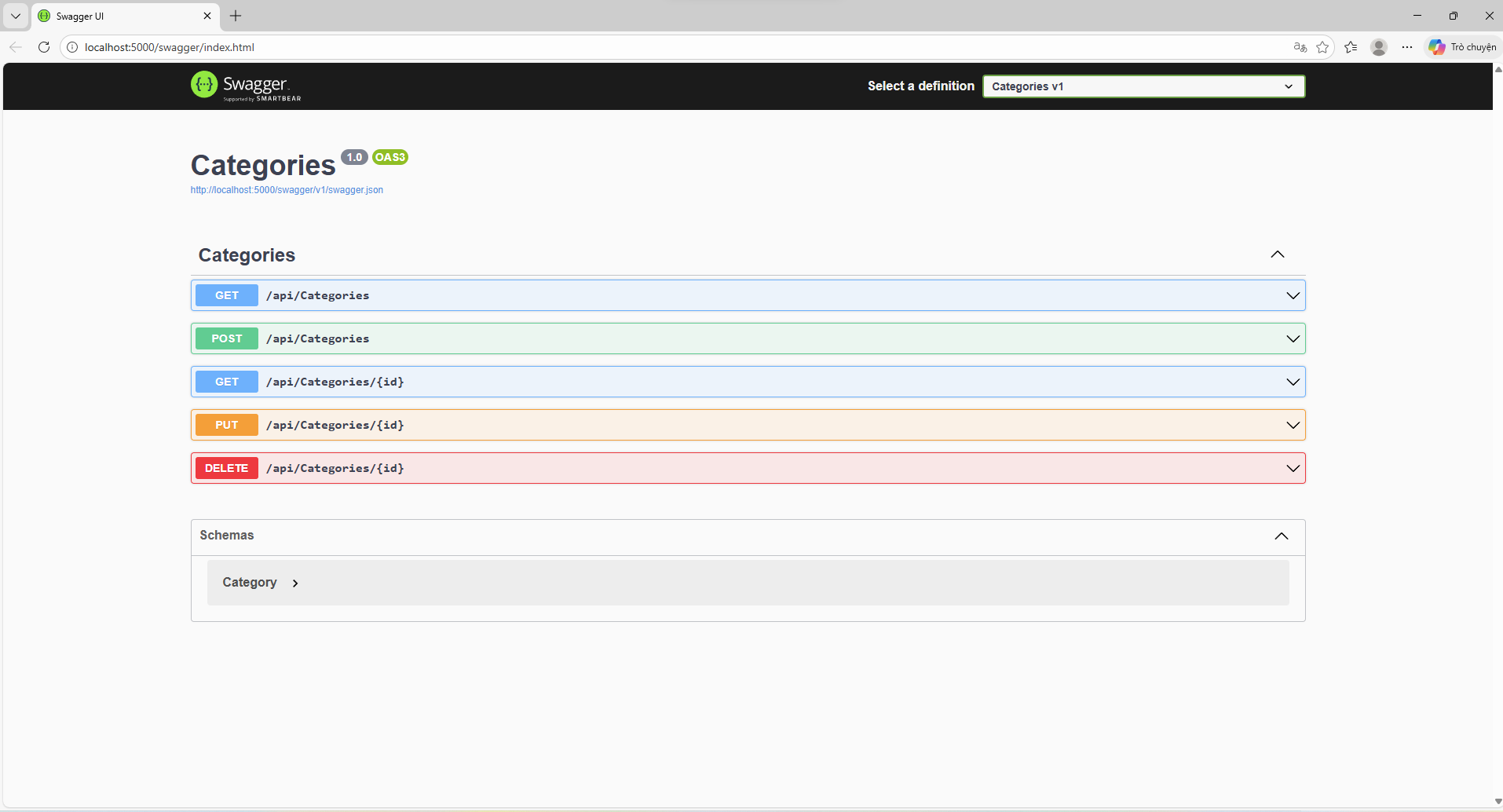
**Kết quả dự kiến:**

* ✅ Ứng dụng chạy thành công (không báo lỗi)
* ✅ Hiển thị: "Application started. Press Ctrl+C to shut down."
* ✅ Swagger UI: <https://localhost:5001/swagger/index.html>

**Hãy chụp ảnh hoặc ghi lại kết quả:**



**Swagger UI:**



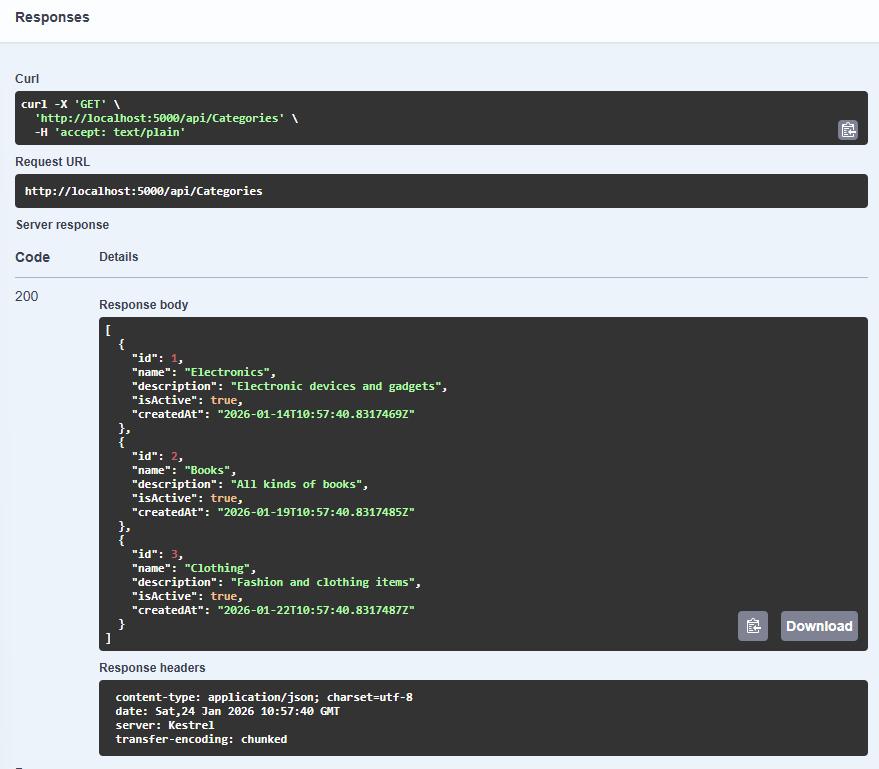
### 🧪 Bước 2: Test 5 Endpoints bằng Swagger UI

**Test GET /api/categories - Lấy danh sách:**

1. Mở Swagger UI: <https://localhost:5001/swagger/index.html>
2. Click "GET /api/categories"
3. Click "Try it out" → "Execute"
4. Kết quả dự kiến: HTTP 200 + danh sách 3 categories

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc JSON response ở đây)



**Giải thích:** Response có bao nhiêu categories? Tại sao?

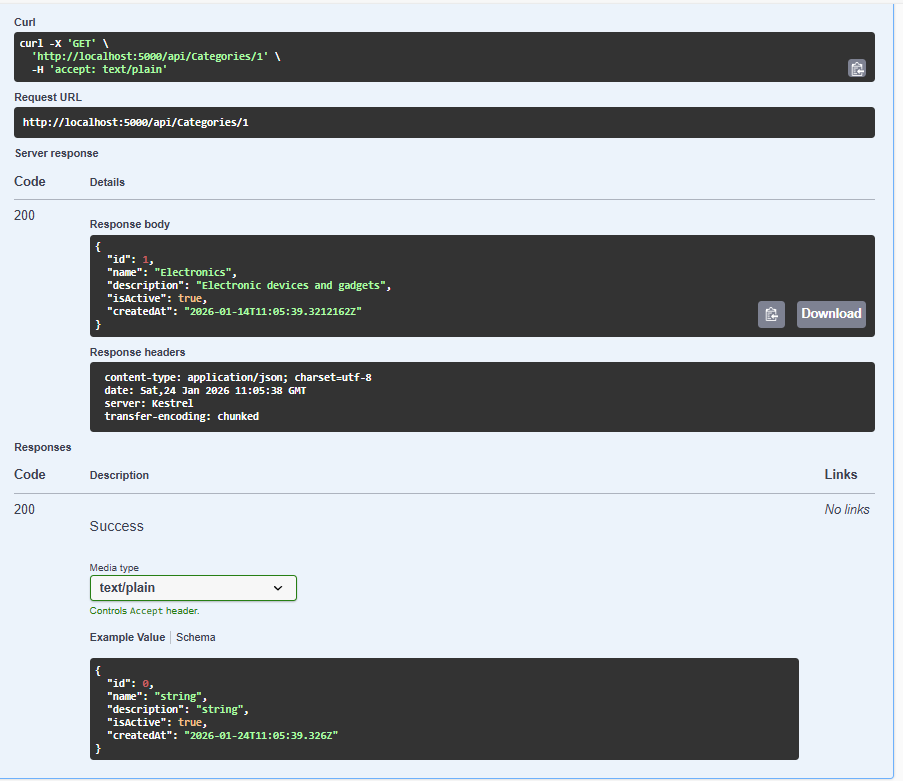
* + Có 3 categories vì
    - Khi [CategoryService](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") được khởi tạo, nó tạo một List [\_categories](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
    - List này **đã chứa sẵn 3 categories** từ lúc bắt đầu
    - Mỗi lần ứng dụng chạy, 3 categories này sẽ xuất hiện

**Test GET /api/categories/{id} - Lấy chi tiết (ID = 1):**

1. Click "GET /api/categories/{id}"
2. Nhập: **1** vào parameter id
3. Click "Execute"
4. Kết quả dự kiến: HTTP 200 + 1 category (Electronics)

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc JSON response ở đây)



**Giải thích:** Tên category với ID = 1 là gì?

* Khi gửi request GET /api/categories/1
* Server tìm category có id=1 trong list
* Tìm thấy category có:
  + **id**: 1
  + **name**: "Electronics" ← **ĐÂY LÀ ĐÁP ÁN**
  + **description**: "Electronic devices and gadgets"
  + **isActive**: true

**Test POST /api/categories - Tạo mới:**

1. Click "POST /api/categories"
2. Click "Try it out"
3. Nhập JSON vào Request body:

{

"name": "Sports",

"description": "Sports equipment and gear"

}

1. Click "Execute"
2. Kết quả dự kiến: HTTP 201 + category mới (ID = 4)

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc JSON response ở đây)



**Giải thích:**

* ID của category mới là bao nhiêu?

Là 4

* Tại sao status code là 201 chứ không phải 200?

 Vì đây là **POST (tạo mới)**, nên:

* **HTTP 200 (OK)** = Thành công, nhưng không tạo resource mới
* **HTTP 201 (Created)** = Thành công VÀ tạo resource mới

**Test PUT /api/categories/{id} - Cập nhật (ID = 1):**

1. Click "PUT /api/categories/{id}"
2. Nhập: **1** vào parameter id
3. Nhập JSON vào Request body:

{

"name": "Digital Devices",

"description": "Updated: Modern digital devices",

"isActive": true

}

1. Click "Execute"
2. Kết quả dự kiến: HTTP 200 + category được update

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc JSON response ở đây)



**Giải thích:** Tên của category ID = 1 đã được cập nhật thành gì?

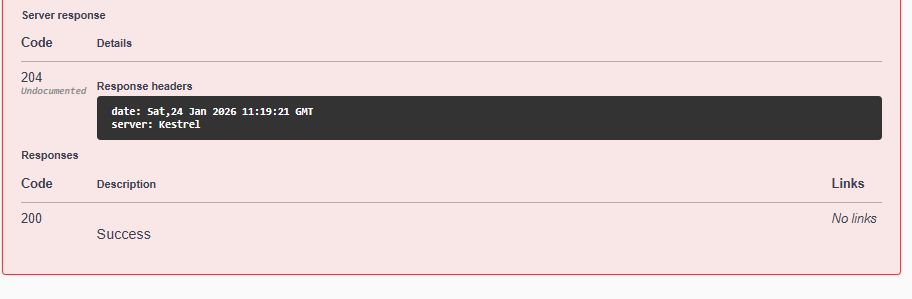
* + Tên ID = 1 được cập nhật thành **Digital Devices**

**Test DELETE /api/categories/{id} - Xóa (ID = 2):**

1. Click "DELETE /api/categories/{id}"
2. Nhập: **2** vào parameter id
3. Click "Execute"
4. Kết quả dự kiến: HTTP 204 (No Content)

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc status code ở đây)



**Giải thích:** Tại sao response body trống? (HTTP 204 có ý nghĩa gì?)

* + Vì **HTTP 204 (No Content)** theo quy tắc API REST, khi xóa thành công không cần trả về data.
  + 204 = No Content = Yêu cầu thành công nhưng không có nội dung trả về

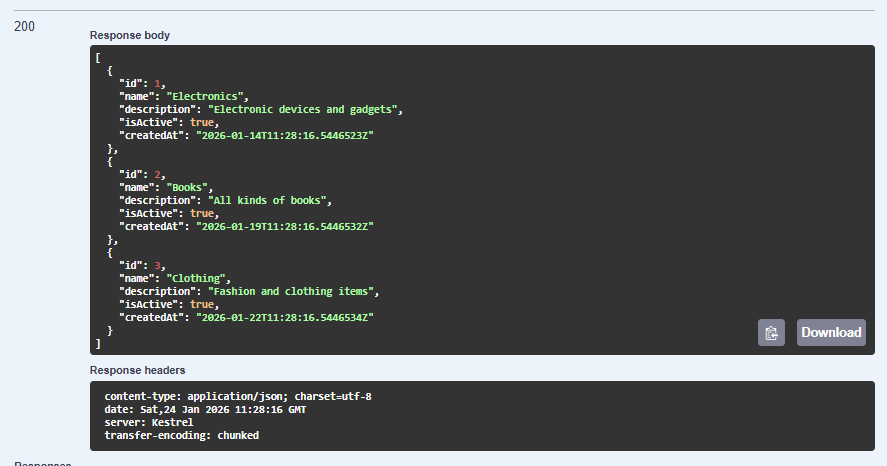
### 🔗 Bước 3: Verify - Gọi GET lại để xác nhận

**TODO:** Gọi GET /api/categories lại để xác nhận:

1. Click "GET /api/categories"
2. Click "Execute"
3. Kiểm tra xem category ID = 2 (Books) còn trong danh sách không?

**Hãy chụp ảnh Response:**

(Dán ảnh hoặc JSON response ở đây)



**Giải thích:** Danh sách bây giờ có bao nhiêu items? (Tại sao?)

* + Vẫn còn 3 items, khả năng do DELETE không hoạt động

## 4. Câu hỏi bắt buộc (Hiểu sâu kiến thức)

### Câu hỏi 1: Dependency Injection

**Hãy nhìn vào code Controller dưới đây:**

public class CategoriesController : ControllerBase

{

private readonly ICategoryService \_categoryService;

public CategoriesController(ICategoryService categoryService)

{

\_categoryService = categoryService;

}

}

**Câu hỏi:**

* A. ASP.NET Core sẽ tự động inject ICategoryService vào constructor
* B. Chúng ta phải tự tạo new CategoryService()
* C. Phải dùng reflection để inject
* D. Không thể inject (lỗi)

**Trả lời:** A (Chọn A, B, C, hay D)

**Giải thích:** Tại sao bạn chọn đáp án đó?

[ASP.NET](https://asp.net/) Core có built-in Dependency Injection container sẽ tự động inject service khi nó được đăng ký trong Program.cs

### Câu hỏi 2: Service Lifetime

**TODO 1:** Trong Program.cs, chúng ta dùng:

builder.Services.AddScoped<ICategoryService, CategoryService>();

**Câu hỏi:** AddScoped có nghĩa là gì?

**Trả lời:**

* Mỗi HTTP request tạo \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ instance CategoryService
* Nếu trong 1 request có 10 Controllers gọi \_categoryService, sẽ có bao nhiêu instances?
  + A. 1 instance
  + B. 10 instances
  + C. Tùy implementation

**Chọn:** A

Mỗi HTTP request tạo **một** instance CategoryService

### Câu hỏi 3: Route

**TODO:** Cho code Controller:

[ApiController]

[Route("api/[controller]")]

public class CategoriesController : ControllerBase

{

[HttpGet]

public ActionResult GetAll() { ... }

[HttpGet("{id}")]

public ActionResult GetById(int id) { ... }

}

**Hãy viết ra đúng route cho 2 methods:**

* GetAll() → Route: **/api/categories** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* GetById() → Route: **/api/categories/{id}** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Câu hỏi 4: HTTP Status Codes

**Hãy gán đúng HTTP status code cho mỗi scenario:**

| **Scenario** | **Status Code** |
| --- | --- |
| Tạo mới category thành công | 201 (200 / 201 / 204) |
| Xóa category thành công | 204 (200 / 201 / 204) |
| Cập nhật category thành công | 200 (200 / 201 / 204) |
| Category không tồn tại | 404 (400 / 404 / 500) |
| Request body không hợp lệ | 400 (400 / 404 / 500) |

### Câu hỏi 5: So sánh Code

**Cách A - ❌ SAI:**

[HttpPost]

public ActionResult CreateCategory(Category category)

{

// Logic tạo category trực tiếp trong Controller

\_categories.Add(category);

\_categories.Max(c => c.Id);

return Ok(category);

}

**Cách B - ✅ ĐÚNG:**

[HttpPost]

public ActionResult CreateCategory(Category category)

{

var created = \_categoryService.CreateCategory(category);

return CreatedAtAction(nameof(GetCategoryById), new { id = created.Id }, created);

}

**Câu hỏi:** Tại sao Cách B tốt hơn Cách A? (Chọn 2 lý do)

* ☐ Cách B tách business logic khỏi Controller
* ☐ Cách B dễ test hơn (có thể mock service)
* ☐ Cách B trả về status code chính xác (201 Created)
* ☐ Cách B có ít code hơn

**Chọn 2 lý do là :**

* Cách B tách business logic khỏi Controller
* Cách B dễ test hơn (có thể mock service)

## 5. Bài tập nâng cao (Thêm tính năng)

### 📝 Bài 1: Thêm Filter (Tìm kiếm theo Name)

**Yêu cầu:** Thêm query parameter name vào GET endpoint:

// GET /api/categories?name=electronics

[HttpGet]

public ActionResult<List<Category>> GetAllCategories(string? name = null)

{

var categories = \_categoryService.GetAllCategories();

if (!string.IsNullOrEmpty(name))

{

// TODO: Hãy viết LINQ để filter theo name (case-insensitive)

categories = categories.Where(c => **c.name**.Contains(name, StringComparison.OrdinalIgnoreCase

))

.ToList();

}

return Ok(categories);

}

**TODO:** Hoàn thành dòng .Where() để filter categories:

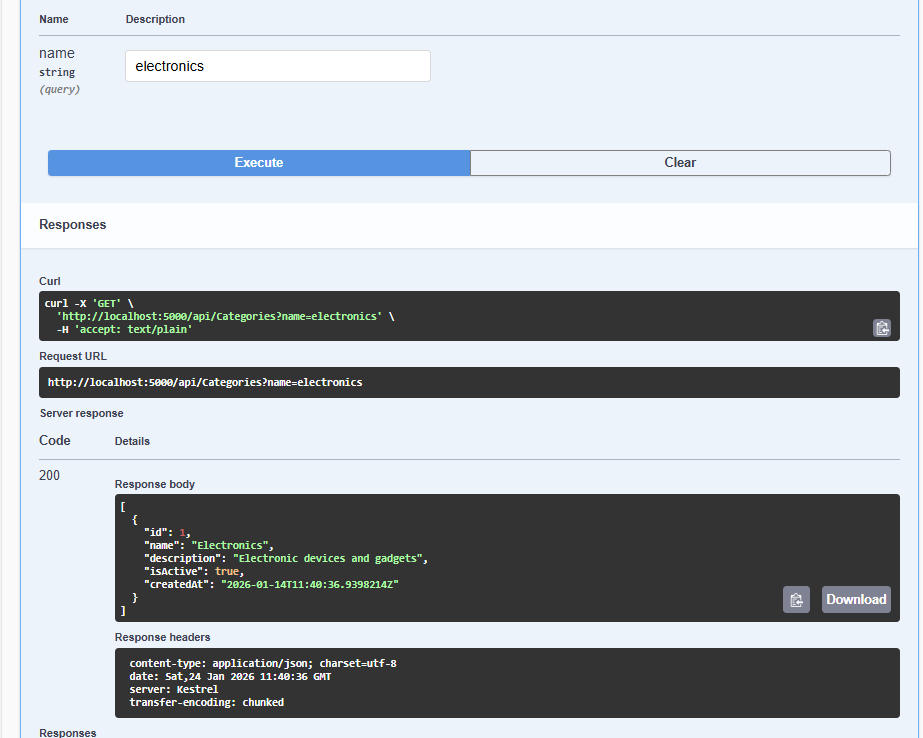
// (Viết code filter của bạn)

**Test:**

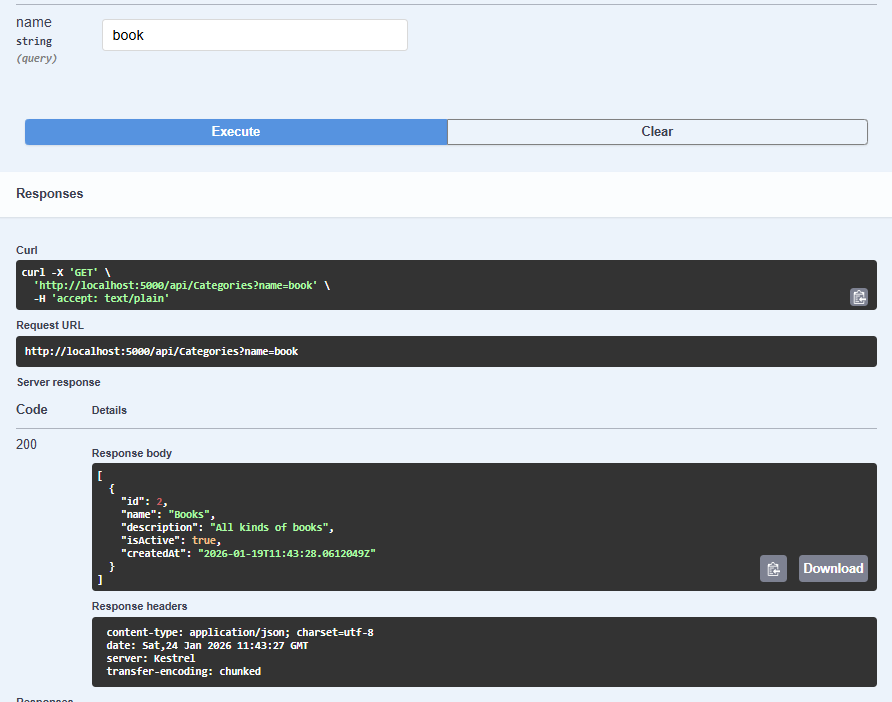
* GET /api/categories?name=book → Kết quả: Books
* GET /api/categories?name=ELEC → Kết quả: Electronics

**Hãy chụp ảnh kết quả test:**

**Elec**



Book



### 📝 Bài 2: Thêm Validation (Validate Name không trống)

**Yêu cầu:** Validate Name property:

* Không được null/trống
* Độ dài tối thiểu 3 ký tự
* Độ dài tối đa 100 ký tự

**TODO:** Sửa model Category.cs để thêm Data Annotation:

public class Category

{

public int Id { get; set; }

// TODO: Thêm [Required] và [StringLength(100, MinimumLength = 3)]

[Required]

[StringLength(100, MinimumLength = 3)]

public string Name { get; set; } = string.Empty;

// ... properties khác

}

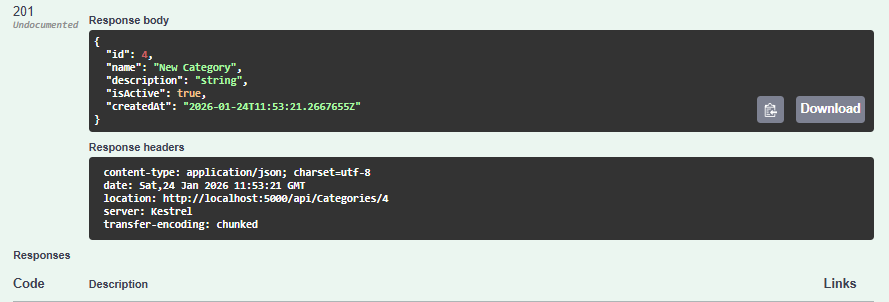
**Test:**

1. POST /api/categories với {"name": ""} → Kết quả:  HTTP **400 Bad Request** (Status code)
2. POST /api/categories với {"name": "AB"} → Kết quả: HTTP **400 Bad Request** (Status code)
3. POST /api/categories với {"name": "New Category"} → Kết quả: HTTP **201 Created** (Status code)

**Hãy chụp ảnh kết quả test:**







### 📝 Bài 3: Thêm Pagination

**Yêu cầu:** Thêm pagination vào GET endpoint:

// GET /api/categories?page=1&pageSize=2

[HttpGet]

public ActionResult<List<Category>> GetAllCategories(int page = 1, int pageSize = 10)

{

var categories = \_categoryService.GetAllCategories();

// TODO: Dùng Skip() và Take() để implement pagination

var paginated = categories.Skip((page - 1) \* pageSize )

.Take (pageSize)

.ToList();

return Ok(paginated);

}

**TODO:** Viết công thức Skip/Take:

// Skip = (page - 1) \* pageSize

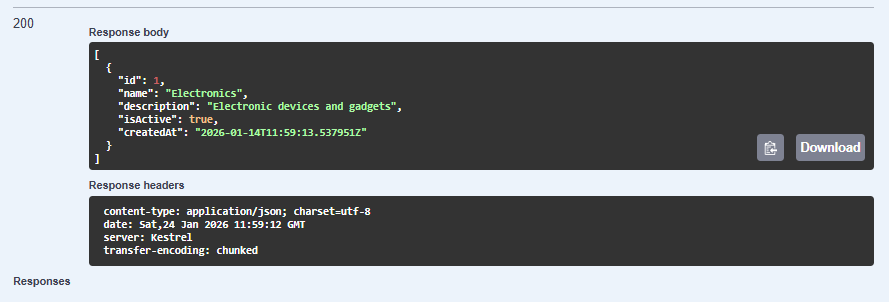
// Take = pageSize

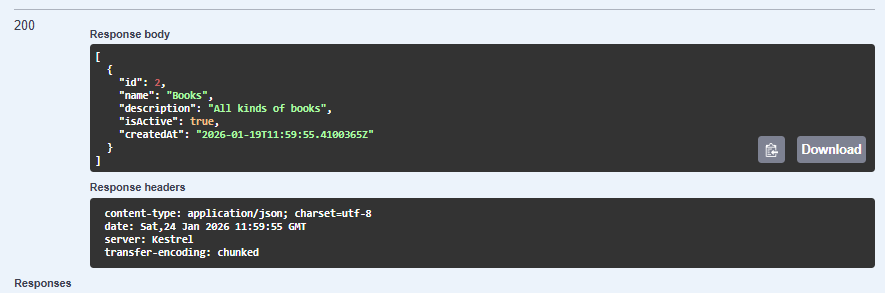
// (Viết code của bạn)

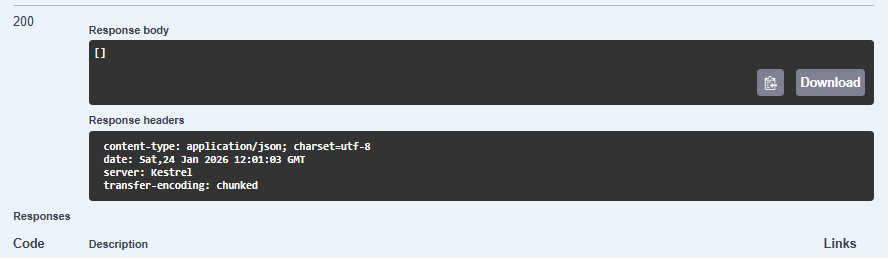
**Test:**

* GET /api/categories?page=1&pageSize=2 → Lấy: Electronics, Books
* GET /api/categories?page=2&pageSize=2 → Lấy: Clothing

**Hãy chụp ảnh kết quả test:**







## 6. Lỗi thường gặp & Giải pháp

### ❌ Lỗi 1: "InvalidOperationException: Unable to resolve service"

**Lỗi xảy ra ở dòng nào?**

Lỗi xảy ra khi **chạy ứng dụng**, thường ở dòng khởi tạo dependency:

**Nguyên nhân:**

* **Quên đăng ký service** trong [Program.cs](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")
* Service [ICategoryService](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") chưa được thêm vào DI Container
* ASP.NET Core không biết làm sao để inject [ICategoryService](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") vào constructor

**Cách sửa:**

Thêm dòng này vào [Program.cs](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") **trước khi** [var app = builder.Build();](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

builder.Services.AddScoped<ICategoryService, CategoryService>();

**Chứng minh cách sửa:** (Paste dòng code đã sửa)

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// ← THÊM DÒNG NÀY

builder.Services.AddScoped<ICategoryService, CategoryService>();

var app = builder.Build();

### ❌ Lỗi 2: "404 Not Found" khi gọi API

**Các bước check:**

☐ Kiểm tra Controller có attribute [ApiController] không? ☐ Kiểm tra có attribute [Route("api/[controller]")] không? ☐ Kiểm tra có dòng app.MapControllers(); trong Program.cs không? ☐ Kiểm tra HTTP method (GET/POST/PUT/DELETE) có đúng không?

**Lỗi của bạn là gì? (Chọn):** ☐ ☐ ☐ ☐

☐ **Quên [ApiController]** → Thêm vào class

☐ **Quên [Route("api/[controller]")]** → Thêm vào

☐  **Quên app.MapControllers()** → Thêm vào [Program.cs](vscode-file://vscode-app/c:/Users/NC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")

☐  **HTTP method sai** → Kiểm tra [HttpGet]/[HttpPost]/[HttpPut]/[HttpDelete]

**Cách sửa:**

**Nếu quên [ApiController] hoặc [Route]:**

[ApiController] // ← Thêm

[Route("api/[controller]")] // ← Thêm

public class CategoriesController : ControllerBase

{

// ...

}

**Nếu quên app.MapControllers():**

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.MapControllers(); // ← Thêm dòng này

app.Run();

**Nếu HTTP method sai:**

[HttpGet] // ← Phải match với request (GET, POST, PUT, DELETE)

public ActionResult<List<Category>> GetAll()

{

// ...

}

### ❌ Lỗi 3: "POST trả về 400 Bad Request"

**Nguyên nhân thường là:**

1. **JSON keys không match C# property names** (camelCase vs PascalCase)
2. **Request body không hợp lệ** (missing fields, wrong data type)
3. **Validation rules bị vi phạm** (Required, StringLength, etc.)

**Các bước debug:**

1. Kiểm tra JSON keys có match với C# property names không?
   * JSON: "name": "..." ↔ C#: public string Name { get; set; }
   * (Lưu ý: camelCase vs PascalCase)
   * // ❌ WRONG - camelCase không match
   * JSON: { "name": "...", "description": "..." }
   * C#: public string Name { get; set; }
   * // ✅ CORRECT - ASP.NET Core tự convert (case-insensitive)
   * JSON: { "name": "...", "description": "..." }
   * C#: public string Name { get; set; }
   * // ASP.NET Core tự động convert "name" → "Name"
2. Request body có hợp lệ không?

**Hãy paste request body bạn đã gửi:**

**Ví dụ request body sai:**

{

"name": "", // ← Lỗi: Trống (Required)

"description": "...",

"isActive": "yes" // ← Lỗi: String thay vì bool

}

**Ví dụ request body đúng:**

{

"name": "New Category", // ← Đúng: Có giá trị, độ dài 3-100

"description": "Mô tả",

"isActive": true // ← Đúng: boolean

}

**Hãy paste error response:**

{

"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",

"title": "One or more validation errors occurred.",

"status": 400,

"errors": {

"Name": [

"Name không được trống" // ← Validation error từ [Required]

]

}

}

**Cách sửa:**

1. Đảm bảo tất cả required fields được điền
2. Kiểm tra data types (string, int, bool, DateTime)
3. Tuân theo validation rules (StringLength, Range, etc.)
4. Xem error response để biết field nào bị lỗi
5. Sửa request body rồi gửi lại

## 7. Một điều tôi chưa hiểu rõ hoặc muốn hỏi thêm

**(Bắt buộc phải đặt ít nhất 2 câu hỏi)**

Ví dụ câu hỏi tốt:

* "Sự khác biệt giữa AddScoped, AddTransient, AddSingleton là gì?"
* "Tại sao phải dùng Service thay vì viết logic trực tiếp trong Controller?"
* "Khi nào nên dùng Task<ActionResult> (async) vs ActionResult (sync)?"
* "Làm sao để validate input request?"

### Câu hỏi 1 của tôi:

"Tại sao phải dùng Service thay vì viết logic trực tiếp trong Controller?"

### Câu hỏi 2 của tôi:

* "Sự khác biệt giữa AddScoped, AddTransient, AddSingleton là gì?"

### Câu hỏi 3 của tôi (nếu có):

## 8. Dự án hoàn thành

### ✅ Checklist:

*Hãy đánh dấu các hạng mục bạn đã hoàn thành:*

**Phần cơ bản:**

* ☐ Tạo Category.cs model
* ☐ Tạo ICategoryService interface
* ☐ Tạo CategoryService implementation
* ☐ Tạo CategoriesController
* ☐ Đăng ký Service trong Program.cs
* ☐ Chạy dotnet run thành công
* ☐ Swagger UI hiển thị đúng

**Phần Test:**

* ☐ Test GET /api/categories ✅
* ☐ Test GET /api/categories/{id} ✅
* ☐ Test POST /api/categories ✅
* ☐ Test PUT /api/categories/{id} ✅
* ☐ Test DELETE /api/categories/{id} ✅

**Phần nâng cao (tối thiểu 1 bài):**

* ☐ Thêm Filter (Bài 1)
* ☐ Thêm Validation (Bài 2)
* ☐ Thêm Pagination (Bài 3)

**Phần trả lời câu hỏi:**

* ☐ Trả lời 5 câu hỏi bắt buộc
* ☐ Đặt ít nhất 2 câu hỏi của riêng mình
* ☐ Giải quyết ít nhất 1 lỗi

## 📸 Hướng dẫn nộp bài

**Hãy gửi file chứa:**

1. **Ảnh chụp / Code:**
   * Category.cs
   * ICategoryService.cs
   * CategoryService.cs
   * CategoriesController.cs
   * Program.cs (phần AddScoped)
2. **Ảnh chụp Test Results:**
   * Swagger UI 5 endpoints
   * GET danh sách (3 categories)
   * GET chi tiết (ID = 1)
   * POST tạo mới (ID = 4)
   * PUT cập nhật (Electronics → Digital Devices)
   * DELETE xóa (Books)
   * GET verify (2 categories còn lại)
3. **Phần trả lời Phiếu Học Tập:**
   * Tất cả chỗ trống điền đầy đủ
   * Câu hỏi trả lời chi tiết
   * Bài tập nâng cao ít nhất 1 cái

## 🎓 Kết luận

**Sau phiếu học tập này, bạn có thể:**

✅ Tạo API đơn giản từ A-Z  
✅ Hiểu rõ HTTP verbs và routes  
✅ Sử dụng Dependency Injection đúng cách  
✅ Tách business logic khỏi Controller (Service Pattern)  
✅ Test API bằng Swagger UI  
✅ Xử lý lỗi thường gặp  
✅ Mở rộng API với thêm tính năng

**Chúc mừng bạn đã hoàn thành Phiếu Học Tập ASP.NET Core!** 🎉

**Hướng dẫn tiếp theo: Entity Framework Core - Kết nối với Database** 💾