

## Bài 9. RESTFUL WEB SERVICE

- **Mục đích, yêu cầu:** Cung cấp cho sinh viên kiến thức về Restful web service và Web API. Sau bài học này sinh viên có khả năng sử dụng LINQ để tạo Web API với HttpGet, HttpPost.

- **Hình thức tổ chức dạy học:** Lý thuyết, trực tiếp + trực tuyến + tự học

- **Thời gian:** Lý thuyết( trên lớp: 3; online: 5) Tự học, tự nghiên cứu: 16

- **Nội dung chính:**

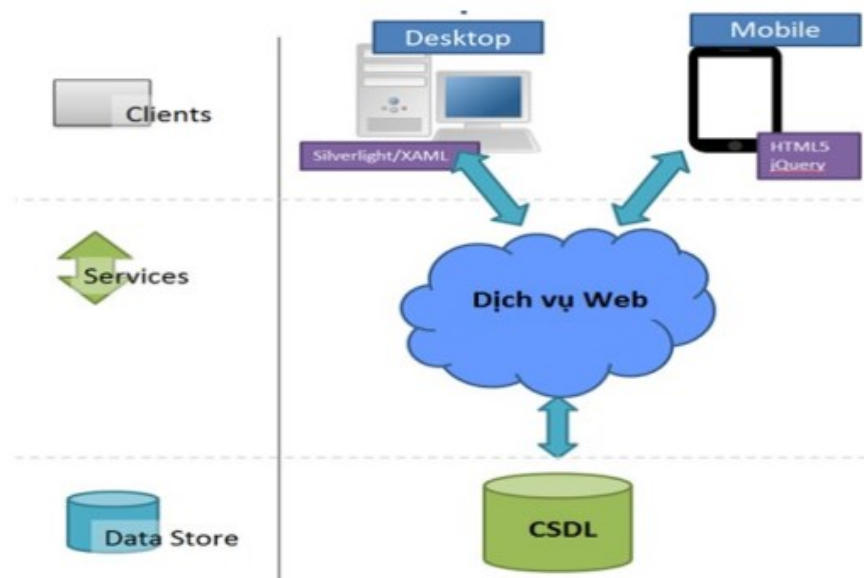
1. Giới thiệu về .NET Web Service .....	2
1.1 Web service là gì? .....	2
1.2 Kiến trúc chung của Web service .....	2
1.3 Ưu nhược điểm của Web service .....	2
2. Giới thiệu về RESTful Web service .....	3
3. Giới thiệu về Web API.....	4
4. Xây dựng Web API.....	8
5. Tạo Web API với HttpGet .....	13
6. Tạo Web API với HttpPost .....	17
7. Tạo Web API với HttpPut .....	17
1.1 8. Tạo Web API với HttpDelete .....	18
Tài liệu tham khảo: .....	20

## 1. Giới thiệu về .NET Web Service

### 1.1 Web service là gì?

Theo định nghĩa của W3C, Web service là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng Internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML, JSON. Web service là tài nguyên phần mềm có thể xác định bằng địa chỉ URL, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu. Một Web service được tạo nên bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng nhìn thấy và có thể truy cập đến những dịch vụ mà nó thực hiện, đồng thời có thể yêu cầu thông tin từ Web service khác. Nó bao gồm các mô đun độc lập cho hoạt động của khách hàng và doanh nghiệp và bản thân nó được thực thi trên server.

### 1.2 Kiến trúc chung của Web service



### 1.3 Ưu nhược điểm của Web service

#### ❖ Ưu điểm

✓ Cung cấp khả năng hoạt động rộng lớn với các ứng dụng phần mềm khác nhau chạy trên những nền tảng khác nhau.

- ✓ Sử dụng các giao thức và chuẩn mở.
- ✓ Nâng cao khả năng tái sử dụng.
- ✓ Thúc đẩy đầu tư các hệ thống phần mềm đã tồn tại.
- ✓ Tạo môi quan hệ tương tác lẫn nhau và mềm dẻo giữa các thành phần trong hệ thống, dễ dàng cho việc phát triển các ứng dụng phân tán.
- ✓ Thúc đẩy hệ thống tích hợp, giảm sự phức tạp của hệ thống, hạ giá thành hoạt động, phát triển hệ thống nhanh và tương tác hiệu quả với hệ thống của các doanh nghiệp khác.

#### ❖ **Nhược điểm**

- ✓ Vào những khoảng thời gian chết của Web service sẽ dẫn đến những thiệt hại lớn:
  - Giao diện không thay đổi
  - Có thể lỗi nếu một máy khách không được nâng cấp
  - Thiếu các giao thức cho việc vận hành
- ✓ Có quá nhiều chuẩn cho Web Service khiến người dùng khó nắm bắt.
- ✓ Phải quan tâm nhiều hơn đến vấn đề an toàn và bảo mật.

## 2. Giới thiệu về RESTful Web service

RESTful Web Service là các Web Service được viết dựa trên kiến trúc REST. REST đã được sử dụng rộng rãi thay thế cho các Web Service dựa trên SOAP và WSDL. RESTful Web Service nhẹ (lightweigh), dễ dàng mở rộng và bảo trì.

Những khái niệm đầu tiên về REST (REpresentational State Transfer) được đưa ra vào năm 2000 trong luận văn tiến sĩ của *Roy Thomas Fielding* (đồng sáng lập giao thức HTTP). Trong luận văn ông giới thiệu khá chi tiết về các ràng buộc, quy ước cũng như cách thức thực hiện với hệ thống để có được một hệ thống REST.

REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web services, chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được truyền tải qua HTTP, và được viết bởi nhiều ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST đã nổi lên trong vài năm qua như

là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều.

REST là một bộ quy tắc để tạo ra một ứng dụng Web Service, mà nó tuân thủ 4 nguyên tắc thiết kế cơ bản sau:

1. Sử dụng các phương thức HTTP một cách rõ ràng
2. Phi trạng thái
3. Hiển thị cấu trúc thư mục như các URLs
4. Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.

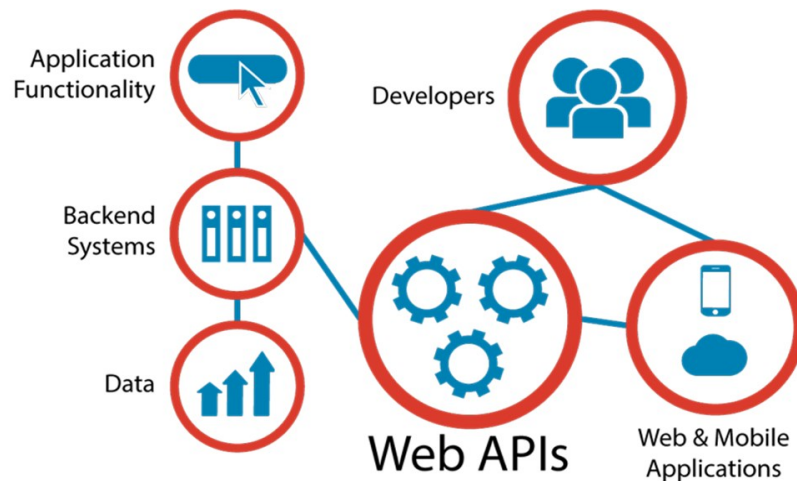
**Lưu ý:** Trong từ **RESTful**, thì từ **ful** chính là hậu tố (suffix) trong tiếng Anh, giống như từ **help** có nghĩa là giúp đỡ thì từ **helpful** là rất hữu ích.

### 3. Giới thiệu về Web API

Hiện nay API nói chung và Web API nói riêng đang được ứng dụng ngày càng nhiều. Kiến trúc ứng dụng hiện đại ngày nay ngày càng phân tán, không phụ thuộc ngôn ngữ đã thúc đẩy việc ứng dụng API. Vậy **API là gì?** Nguồn gốc và ưu điểm của nó là như thế nào?

#### API là gì?

**API** là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của **Application Programming Interface** – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng.



### API thường ứng dụng vào đâu?

- Web API: là hệ thống API được sử dụng trong các hệ thống website. Hầu hết các website đều ứng dụng đến Web API cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. Ví dụ: Bạn thiết kế chức năng login thông Google, Facebook, Twitter, Github... Điều này có nghĩa là bạn đang gọi đến API của. Hoặc như các ứng dụng di động đều lấy dữ liệu thông qua API.
- API trên hệ điều hành: Windows hay Linux có rất nhiều API, họ cung cấp các tài liệu API là đặc tả các hàm, phương thức cũng như các giao thức kết nối. Nó giúp lập trình viên có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành.
- API của thư viện phần mềm hay framework: API mô tả và quy định các hành động mong muốn mà các thư viện cung cấp. Một API có thể có nhiều cách triển khai khác nhau và nó cũng giúp cho một chương trình viết bằng ngôn ngữ này có thể sử dụng thư viện được viết bằng ngôn ngữ khác. Ví dụ bạn có thể dùng Php để yêu cầu một thư viện tạo file PDF được viết bằng C++.

Một số khái niệm khác:

- API hiện nay đều tuân thủ theo tiêu chuẩn REST và HTTP, tạo sự thân thiện dễ sử dụng với nhà phát triển. Giúp người dùng dễ dàng truy cập, dễ hiểu hơn. Web API hiện đại dùng cho các đối tượng cụ thể, chẳng hạn như mobile developer với document, version khác nhau.

- **API key:** Đây là loại code (string) được truyền tải bởi các chương trình máy tính gọi là API để xác định chương trình, nhà phát triển hoặc người dùng nó tới trang web. Các API key được sử dụng với mục đích nhằm giới hạn, kiểm soát sử dụng API. Chẳng hạn như ngăn chặn sự việc lạm dụng API.

API Key thường hoạt động như một mã định danh duy nhất và mã thông báo bí mật để xác thực và thường sẽ có một bộ quyền truy cập trên API được liên kết với nó. Các API Key có thể dựa trên hệ thống định danh duy nhất toàn cầu (UUID) để đảm bảo chúng sẽ là duy nhất cho mỗi người dùng.

### **Web API là gì?**

**Web API** là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường ở dạng **JSON** hoặc XML thông qua giao thức HTTP hoặc HTTPS.

### **Những điểm nổi bật của Web API**

Web API hỗ trợ restful đầy đủ các phương thức: Get/Post/put/delete dữ liệu. Nó giúp bạn xây dựng các HTTP service một cách rất đơn giản và nhanh chóng. Nó cũng có khả năng hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content format.

**Tự động hóa sản phẩm:** Với **web API**, chúng ta sẽ tự động hóa quản lý công việc, cập nhật luồng công việc, giúp tăng năng suất và tạo hiệu quả công việc cao hơn.

**Khả năng tích hợp linh động:** API cho phép lấy nội dung từ bất kỳ website hoặc ứng dụng nào một cách dễ dàng nếu được cho phép, tăng trải nghiệm người dùng. API hoạt động như một chiếc cổng, cho phép các công ty chia sẻ thông tin được chọn nhưng vẫn tránh được những yêu cầu không mong muốn.

**Cập nhật thông tin thời gian thực:** API có chức năng thay đổi và cập nhật thay đổi theo thời gian thực. Với công nghệ này, dữ liệu sẽ được truyền đi tốt hơn, thông tin chính xác hơn, dịch vụ cung cấp linh hoạt hơn.

**Có tiêu chuẩn chung dễ sử dụng:** Bất kỳ người dùng, công ty nào sử dụng cũng có thể điều chỉnh nội dung, dịch vụ mà họ sử dụng.

Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test.

### **Web API hoạt động như thế nào?**

- Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi request dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.
- Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trả về kết quả.
- Server trả về kết quả theo định dạng JSON hoặc XML thông qua giao thức HTTP/HTTPS.
- Tại nơi yêu cầu ban đầu là ứng dụng web hoặc ứng dụng di động , dữ liệu JSON/XML sẽ được parse để lấy data. Sau khi có được data thì thực hiện tiếp các hoạt động như lưu dữ liệu xuống Cơ sở dữ liệu, hiển thị dữ liệu...

### **Ưu và nhược điểm của Web API**

Mỗi một ứng dụng bất kỳ đều có những ưu nhược điểm riêng, hỗ trợ tốt cho các ứng dụng. Vì vậy mà web API cũng không ngoại lệ:

#### **❖ Ưu điểm**

- Web API được sử dụng hầu hết trên các ứng dụng desktop, ứng dụng mobile và ứng dụng website.
- Linh hoạt với các định dạng dữ liệu khi trả về client: Json, XML hay định dạng khác.
- Nhanh chóng xây dựng HTTP service: URI, request/response headers, caching, versioning, content formats và có thể host trong ứng dụng hoặc trên IIS.
- Mã nguồn mở, hỗ trợ chức năng RESTful đầy đủ, sử dụng bởi bất kì client nào hỗ trợ XML, Json.
- Hỗ trợ đầy đủ các thành phần MVC như: routing, controller, action result, filter, model binder, IoC container, dependency injection, unit test.

- Giao tiếp hai chiều được xác nhận trong các giao dịch, đảm bảo độ tin cậy cao.

#### ❖ **Nhược điểm**

Do web API còn khá mới nên chưa thể đánh giá nhiều về nhược điểm của mô hình này. Tuy nhiên, có hai nhược điểm dễ dàng nhận thấy:

- Web API chưa hoàn toàn phải là RESTful service, mới chỉ hỗ trợ mặc định GET, POST
- Để sử dụng hiệu quả cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt
- Tốn thời gian và chi phí cho việc phát triển, nâng cấp và vận hành
- Có thể gặp vấn đề về bảo mật khi hệ thống bị tấn công nếu không giới hạn điều kiện kỹ.

## 4. Xây dựng Web API

### **Bước 1:**

Khởi động Visual Studio → tạo một project ASP.NET Web Application và chọn template Web API.

### **Bước 2:**

Tạo Data Model sử dụng Entity Framework để Web API service có thể tương tác CRUD (Create, Read, Update, Delete) dữ liệu được (không tạo cũng được).

Mỗi bảng là 1 Controller

### **Bước 3:** Tạo ra các Web API

Nhấp chuột phải vào thư mục Controllers và chọn thêm controller.

- Web API 2 Controller Empty: tự viết các phương thức từ đầu.
- Web API 2 Controller with read/write actions: phát sinh các phương thức ví dụ để bạn có thể biết cách viết các service này.

### **Bước 4:** Chạy thử và kiểm tra.

Trước tiên ta tạo cơ sở dữ liệu **CSDLTest** gồm 2 bảng:

- + DanhMuc(**MaDanhMuc**, TenDanhMuc)
- + SanPham(**Ma**, Ten, DonGia, MaDanhMuc)

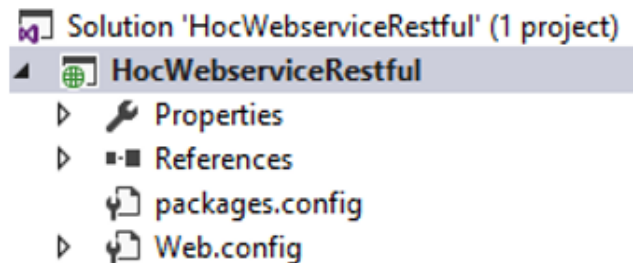


Nhập dữ liệu cho 2 bảng

Khởi động Visual Studio/ vào File/ New/ chọn Project

Chọn Asp.net Web Application, chọn Empty

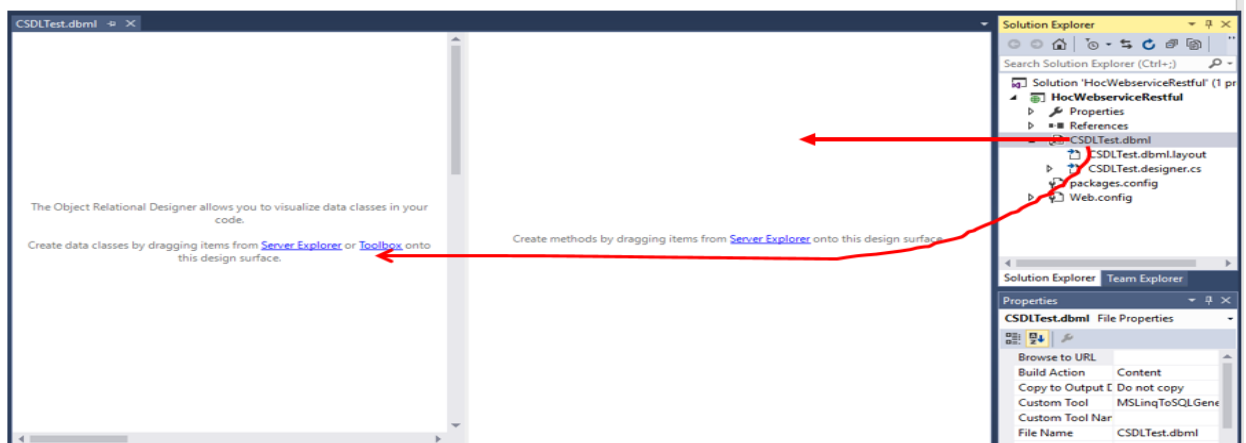
Cấu Project sau khi tạo thành công:



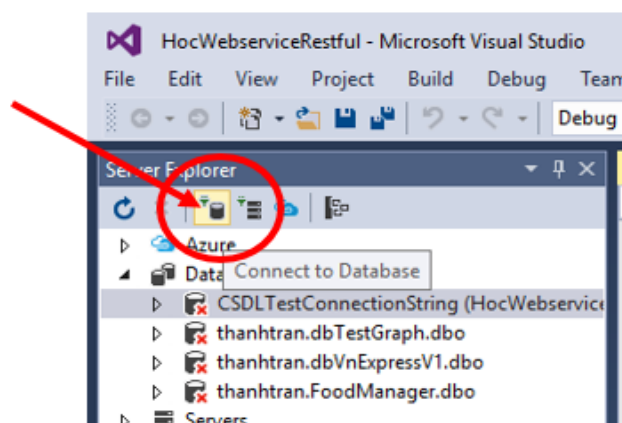
Tiến hành kết nối Cơ sở dữ liệu: Bấm chuột phải vào Project/ add/ LINQ to SQL Classes:

Đặt tên file LinQ, trong trường hợp này Ta nên đặt cùng tên với CSDL: CSDLTest

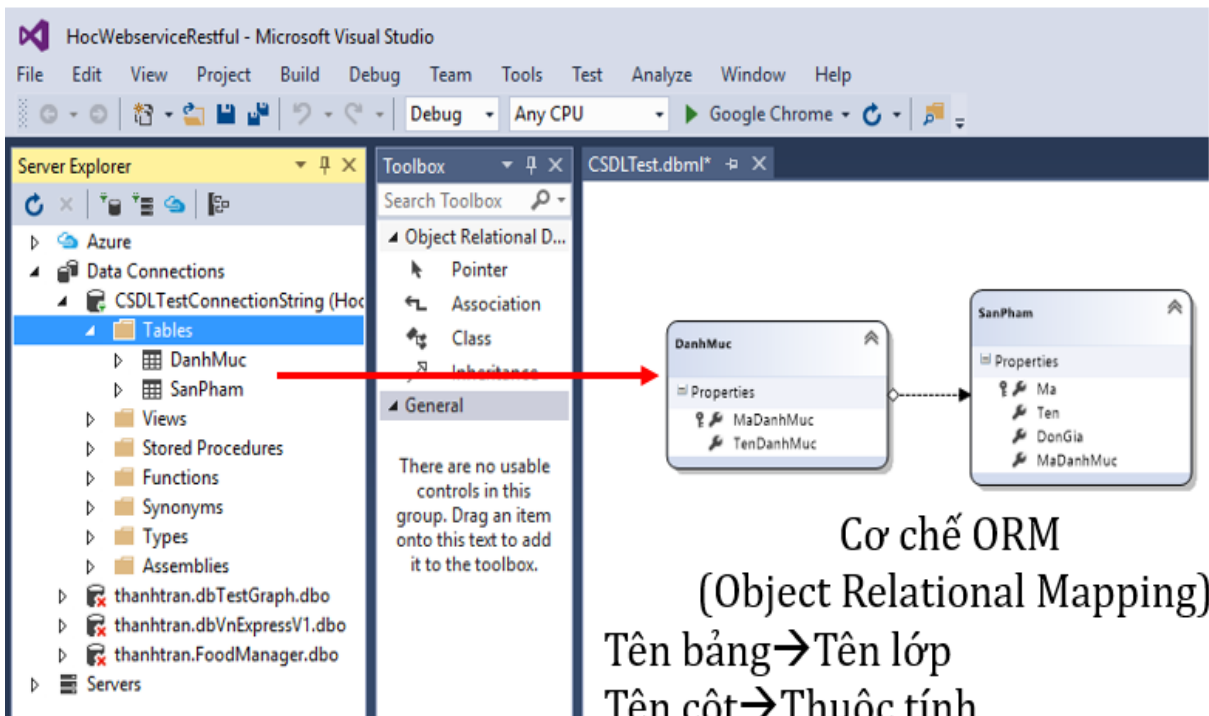
Giao diện LinQ:



Vào Server Explorer để kết nối CSDL



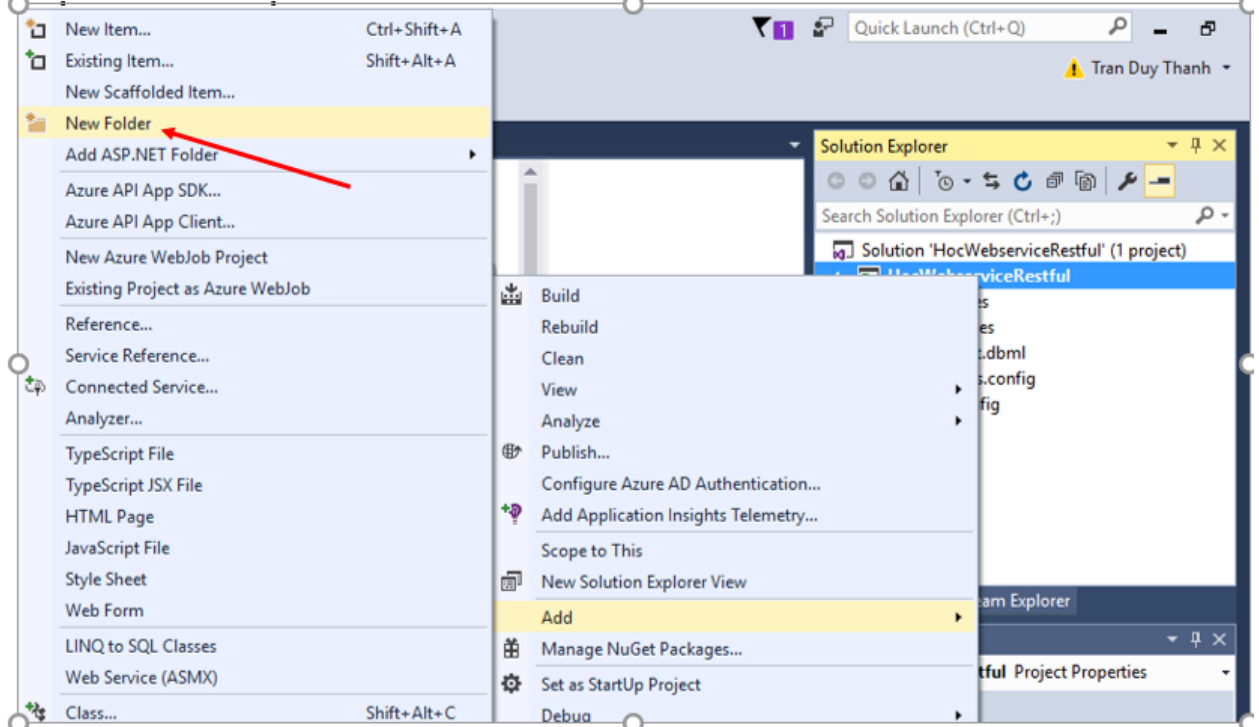
Kéo thả các bảng dữ liệu vào LinQ



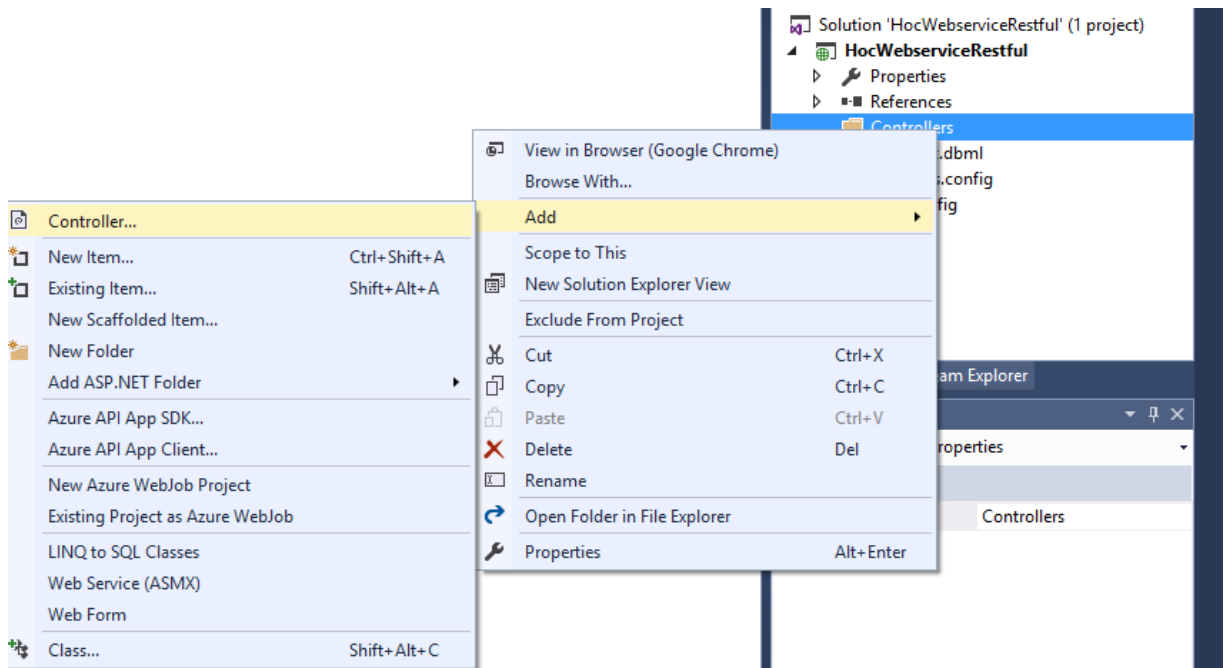
**Cơ chế ORM**  
(Object Relational Mapping)

Tên bảng → Tên lớp  
Tên cột → Thuộc tính  
Từng dòng dữ liệu → tập các đối tượng

## Tạo 1 thư mục Controllers

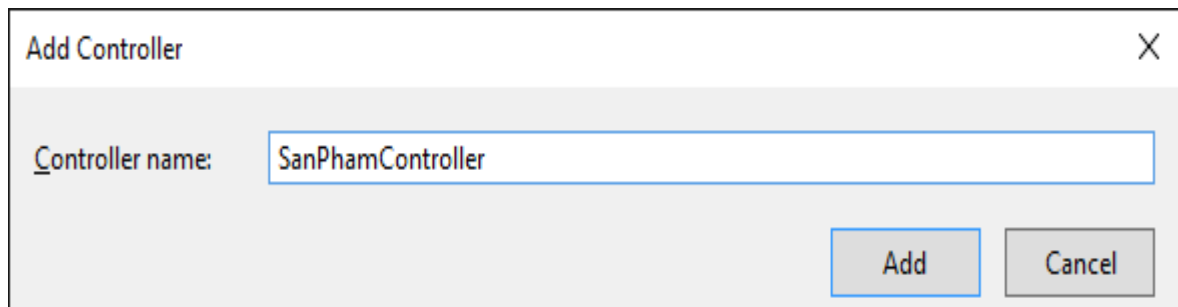


Muốn tương tác Bảng nào trong CSDL thì tạo từng đó Controller (mỗi bảng là 1 Controller độc lập)

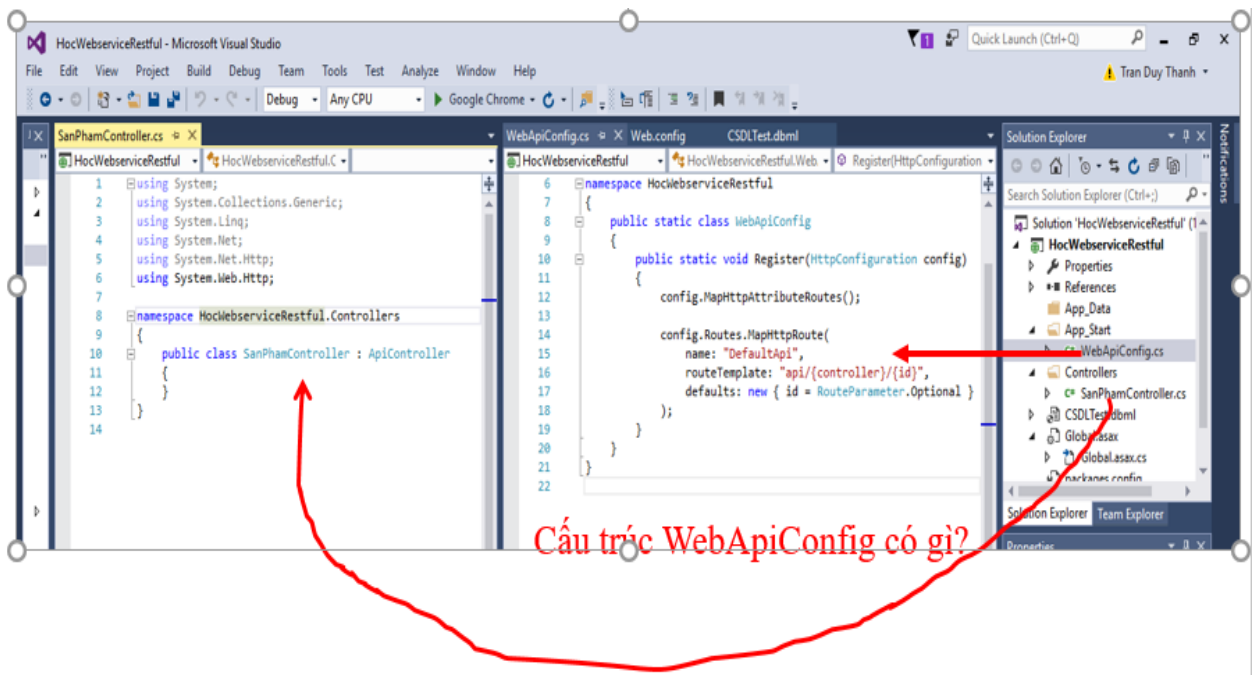


Chọn Web API 2 Controller – Empty rồi nhấn Add

Mặc định là chữ Default, bây giờ bạn đổi lại thành SanPham (thường ta làm API cho bảng nào thì lấy tên bảng đó), đó là lý do vì sao Tui đặt là SanPham:



Tập tin SanPhamController.cs sẽ được tạo ra, nhưng ta chỉ lấy SanPham(bỏ chữ Controller đằng sau đi) để tương tác (đây là cơ chế hoạt động). Ta xem cấu trúc Controller được tạo ra:



File WebApiConfig.cs được sinh ra trong thư mục App\_Start, bạn để ý routeTemplate: “api/{controller}/{id}”, tức là khi ta dùng thì viết: “api/sanpham” để lấy toàn bộ danh sách, hay “api/sanpham/3” để lấy chi tiết 1 Sản phẩm có mã là

- File SanPhamController.cs kế thừa từ ApiController

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.MapHttpAttributeRoutes();

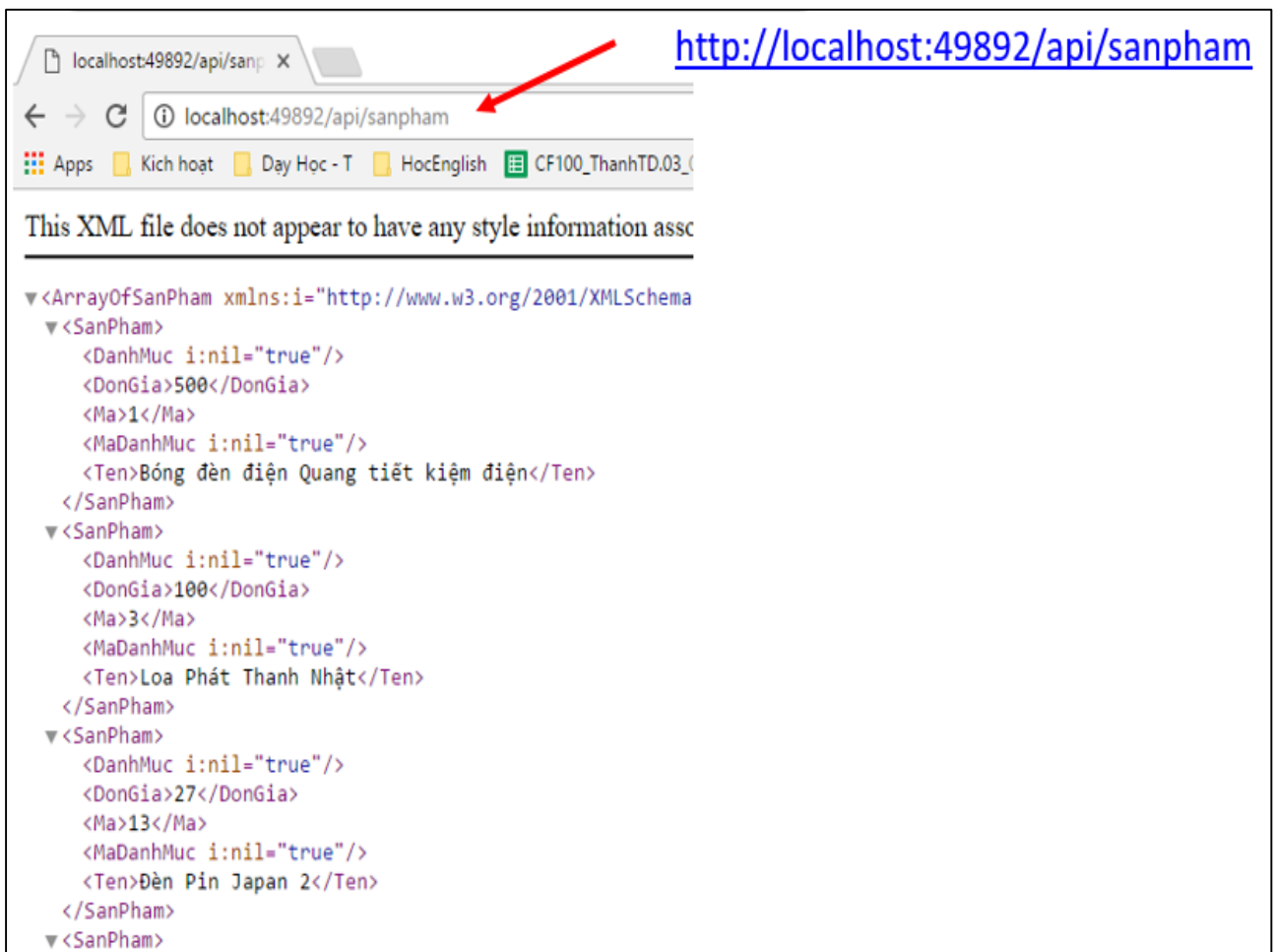
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

Tương tự như tạo Controller cho bảng DanhMuc hay bất kỳ bảng nào khác

## 5. Tạo Web API với HttpGet

+ Tạo Web API lấy toàn bộ danh sách Sản phẩm

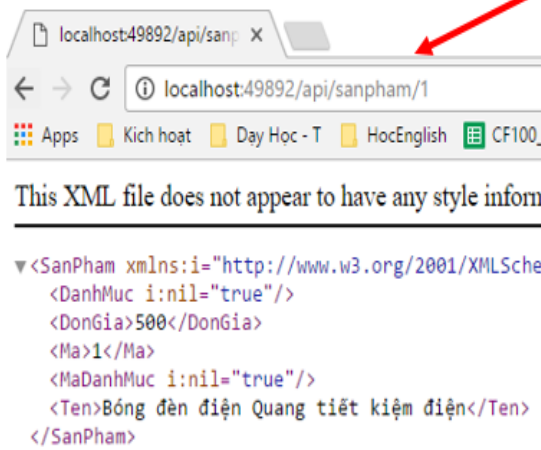
```
[HttpGet]
public List<SanPham>LayToanBoSanPham()
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    List<SanPham> dsSP = context.SanPhams.ToList();
    foreach (SanPham sp in dsSP)
        sp.DanhMuc = null;
    return dsSP;
}
```



+ Tạo Web API lấy chi tiết 1 Sản phẩm

```
[HttpGet]
public SanPham ChiTietSanPham(int id)
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    SanPham sp = context.SanPhams
        .FirstOrDefault(x => x.Ma == id);

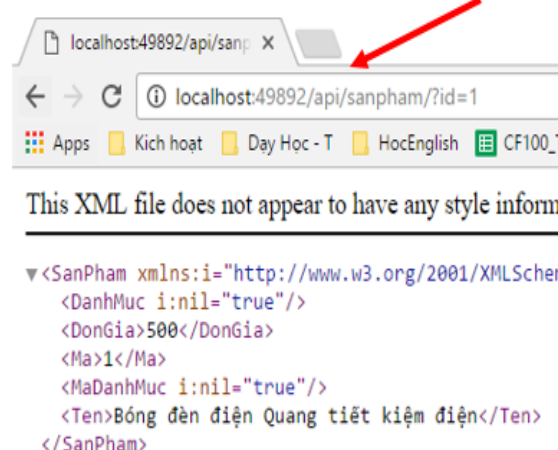
    if(sp!=null)
        sp.DanhMuc = null;
    return sp;
}
```



<http://localhost:49892/api/sanpham/1>

<http://localhost:49892/api/sanpham/1>

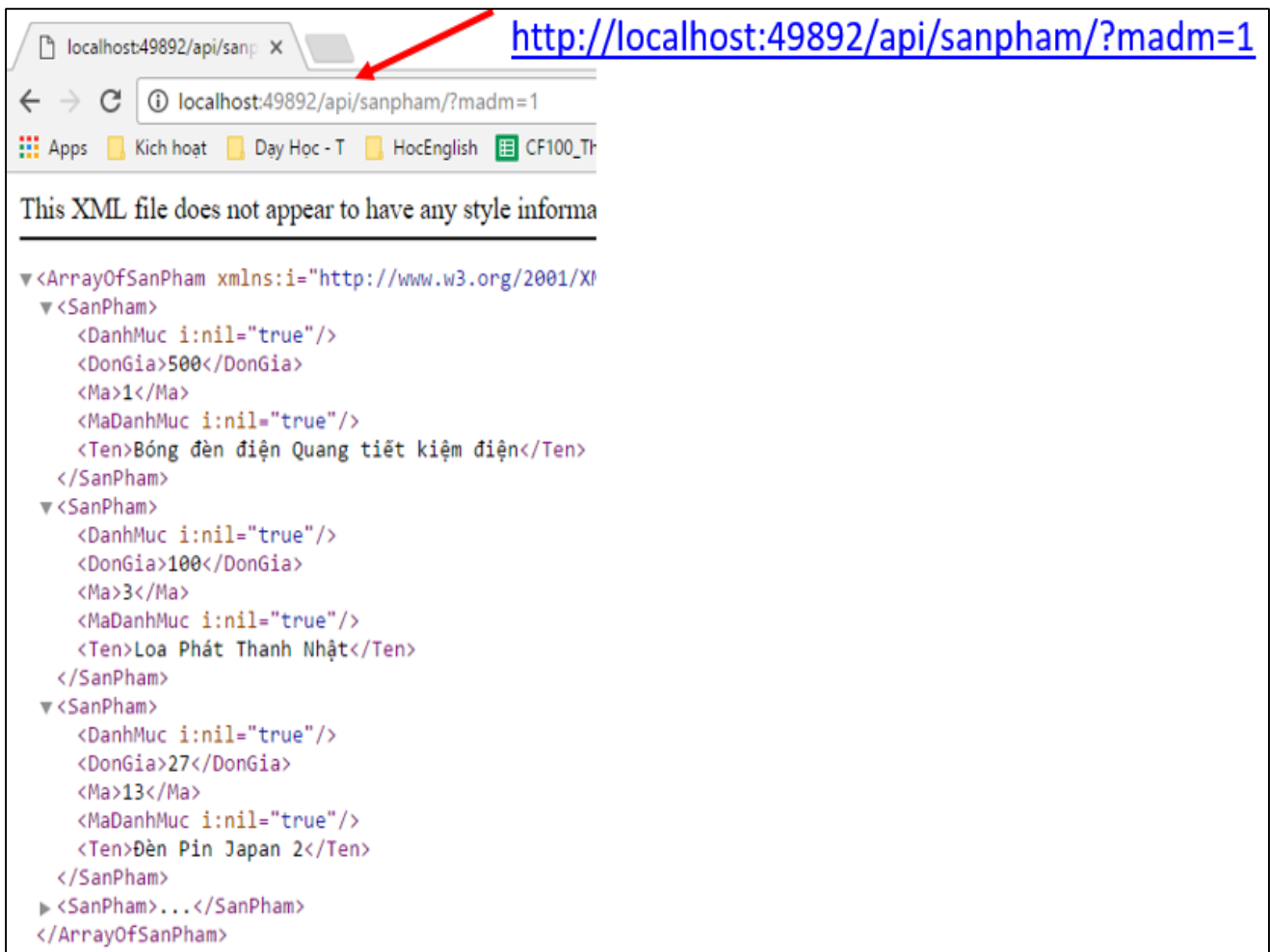
<http://localhost:49892/api/sanpham/?id=1>



+ Lấy danh sách sản phẩm theo danh mục

```
[HttpGet]
public List<SanPham>DanhSachSanPhamTheoDanhMuc(int madm)
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    List<SanPham> dsSP = context.SanPhams
                                .Where(x=>x.MaDanhMuc==madm)
                                .ToList();

    foreach (SanPham sp in dsSP)
        sp.DanhMuc = null;
    return dsSP;
}
```

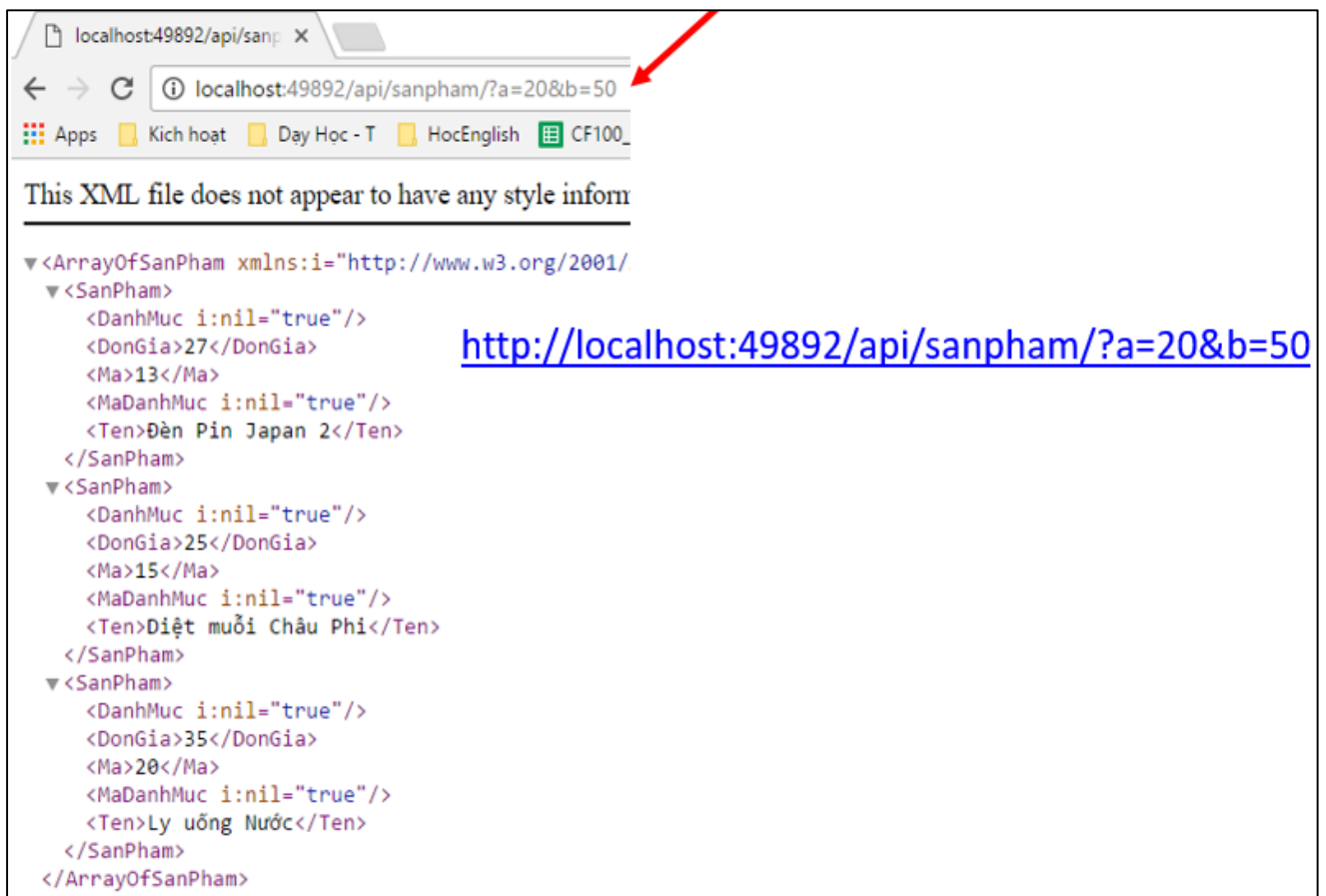




+ Tìm danh sách Sản phẩm có đơn giá [a ... b]

```
[HttpGet]
public List<SanPham> TimDanhSachSanPhamCoGiaTrongDoanAB(int a,int b)
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    List<SanPham> dsSP = context.SanPhams
        .Where(x => x.DonGia>=a && x.DonGia<=b)
        .ToList();

    foreach (SanPham sp in dsSP)
        sp.DanhMuc = null;
    return dsSP;
}
```



## Bài tập

1. Lấy toàn bộ danh sách Danh mục
2. Lấy chi tiết 1 Danh mục



## 6. Tạo Web API với HttpPost

+ Tạo Web API lưu một sản phẩm: SanPhamController.cs

```
[HttpPost]
public bool LuuSanPham(int masp, string tensp, int dongia, int madm)
{
    try
    {
        CSDLTestDataContext context = new CSDLTestDataContext();
        SanPham sp = new SanPham();
        sp.Ma = masp;
        sp.Ten = tensp;
        sp.DonGia = dongia;
        sp.MaDanhMuc = madm;
        context.SanPhams.InsertOnSubmit(sp);
        context.SubmitChanges();
        return true;
    }
    catch { }
    return false;
}
```

+ Lưu một Danh mục: DanhMucController.cs

```
[HttpPost]
public bool LuuDanhMuc(int madm, string tendm)
{
    try
    {
        CSDLTestDataContext context = new CSDLTestDataContext();
        DanhMuc dm = new DanhMuc();
        dm.MaDanhMuc = madm;
        dm.TenDanhMuc = tendm;
        context.DanhMucs.InsertOnSubmit(dm);
        context.SubmitChanges();
        return true;
    }
    catch { }
    return false;
}
```

## 7. Tạo Web API với HttpPut

+ Sửa một sản phẩm: SanPhamController.cs

```
[HttpPut]
public bool SuaSanPham(int masp, string tensp, int dongia)
{
    try
    {
        CSDLTestDataContext context = new CSDLTestDataContext();
        SanPham sp = context.SanPhams
                               .FirstOrDefault(x => x.Ma == masp);
        if (sp != null)
        {
            sp.Ten = tensp;
            sp.DonGia = dongia;
            context.SubmitChanges();
            return true;
        }
    }
    catch { }
    return false;
}
```

+ Sửa một Danh mục

```
[HttpPut]
public bool SuaDanhMuc(int madm, string tendm)
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    DanhMuc dm = context.DanhMucs
                        .FirstOrDefault(x => x.MaDanhMuc == madm);
    if (dm != null)
    {
        dm.TenDanhMuc = tendm;
        context.SubmitChanges();
        return true;
    }
    return false;
}
```

## 1.1 8. Tạo Web API với HttpDelete

+ Xóa một sản phẩm: SanPhamController.cs

```
[HttpDelete]
public bool XoaSanPham(int masp)
{
    CSDLTestDataContext context = new CSDLTestDataContext();
    SanPham sp = context.SanPhams
                        .FirstOrDefault(x => x.Ma == masp);
    if (sp != null)
    {
        context.SanPhams.DeleteOnSubmit(sp);
        context.SubmitChanges();
        return true;
    }
    return false;
}
```

+ Xóa một Danh mục:

```
[HttpDelete]
public bool XoaDanhMuc(int madm)
{
    try
    {
        CSDLTestDataContext context = new CSDLTestDataContext();
        DanhMuc dm = context.DanhMucs
                        .FirstOrDefault(x => x.MaDanhMuc == madm);
        if (dm != null)
        {
            context.DanhMucs.DeleteOnSubmit(dm);
            context.SubmitChanges();
            return true;
        }
    }
    catch { }
    return false;
}
```

### **Tài liệu tham khảo:**

- [1]. Anura Guruge, Web service: Theory and Practice, Elsevier press 2017, 2017
- [2]. Joydip Kanjilal, *ASP.NET Web API: Build RESTful web applications and services on the .NET framework*, Packt Publishing, 2013.