**BÀI TẬP TUẦN 09-10 MÔN LẬP TRÌNH WEB NÂNG CAO**

**Chương 7:**

*Mục tiêu:*

* *Hiểu được thế nào là RESTful services.*
* *Xây dựng được RESTful sử dụng Web API.*
* *Triển khai Web API trên IIS.*
* *Gọi sử dụng services từ các client như HTML (dùng Jquery & Ajax), Console Application, ASP.NET MVC.*

*Yêu cầu:*

* *Tất cả các bài tập lưu trong thư mục: T:\MaSV\_HoTen\Tuan09-10*
* *Tạo Solution Tuan09-.10 trong thư mục T:\MaSV\_HoTen\Tuan09-10 trong Visual Studio 2015. Mỗi bài tập là một Project riêng biệt.*
* *Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.*
* *Bài tập bắt buộc của Tuần 05-10 GV sẽ công bố cuối giờ. Bài tập bắt buộc sẽ gửi vào email* [*nguyenthihongminh@iuh.edu.vn*](mailto:nguyenthihongminh@iuh.edu.vn) *với Subject và tập tin nén kèm theo:* ***DHKTPM10A\_LTWebNC\_TUANYY\_HOTENSINHVIEN*** *(trong đó YY sẽ là 09-10)*

**Lý thuyết ban đầu:**

1. **Web API:**

Web API Controller Characteristics:

* Thừa kế từ lớp System.Web.Http.ApiController.
* Có thể tạo trong bất kỳ thư mục nào của Web API Project, tuy nhiên nên theo quy ước để trong thư mục **Controllers**.
* Tên các phương thức hoạt động của Web API Controller (Action method) có thể cùng tên với các HTTP verb, bắt đầu bởi HTTP verb (case in-sensitive) hoặc có thể áp dụng thuộc tính của HTTP verb.
* Kiểu dữ liệu trả về của Action method có thể là kiểu primitive hoặc kiểu complex type.

Trường hợp Client muốn dữ liệu trả về là dữ liệu định dạng JSON cần gửi GET HTTP với Accept Header tới Web API

*HTTP GET Request:*

GET http://localhost:60464/api/student HTTP/1.1

User-Agent: Fiddler

Host: localhost:1234

**Accept: application/json**

Hoặc Client gửi đi dữ liệu là JSON kèm theo các thông tin:

*HTTP GET Request:*

POST http://localhost:60464/api/student?age=15 HTTP/1.1

User-Agent: Fiddler

Host: localhost:60464

**Content-Type: application/json**

Content-Length: 13

{

id:1,

name:'Steve'

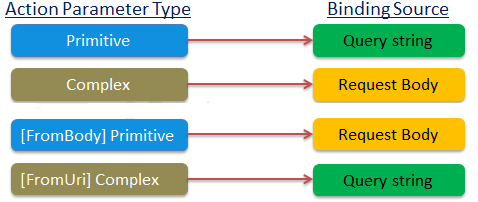
}

Dùng 1 số chương trình để kiểm tra Web API

* Postman: <https://www.getpostman.com/>
* Fiddler: <http://www.telerik.com/fiddler>
* HttpRequester: plug-in FireFox
* …

Tham số ràng buộc trong ASP.NET Web API

|  |  |  |
| --- | --- | --- |
| HTTP Method | Query String | Request Body |
| GET | Primitive Type,  Complex Type | NA |
| POST | Primitive Type | Complex Type |
| PUT | Primitive Type | Complex Type |
| PATCH | Primitive Type | Complex Type |
| DELETE | Primitive Type, Complex Type | NA |

**

* *Primitive Parameter Binding*

public class StudentController : ApiController

{

public Student Get(int id)

{

}

}

HTTP GET Requests

<http://localhost/api/student?id=1>

<http://localhost/api/student?ID=1>

* *Multiple Primitive Parameters*

public class StudentController : ApiController

{

public Student Get(int id, string name)

{

}

}

HTTP GET Requests

<http://localhost/api/student?id=1&name=steve>

<http://localhost/api/student?ID=1&NAME=steve>

<http://localhost/api/student?name=steve&id=1>

* *POST Action Method with Primitive Parameter*

public class StudentController : ApiController

{

public Student Post(id id, string name)

{

}

}

HTTP POST request: <http://localhost/api/student?id=1&name=steve>

* *Post Method with Complex Type Parameter*

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

public class StudentController : ApiController

{

public Student Post(Student stud)

{

}

}

Web API sẽ lấy các giá trị của tham số stud từ HTTP request body. Web API lấy đối tượng JSON từ Request body chuyển thành đối tượng Student (tên thuộc tính của đối tượng JSON tương ứng thuộc tính của lớp Student (case-insensitive).

* *Post Method with Primitive and Complex Type Parameters*

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

public class StudentController : ApiController

{

public Student Post(int age, Student student)

{

}

}

Biến primitive age sẽ được đưa vào query string, student được gán vào HTTP request body

* *[FromUri] and [FromBody]*

public class StudentController : ApiController

{

public Student Get([FromUri] Student stud)

{

}

}

Web API lấy giá trị của tham số loại Student từ query string. HTTP GET request: <http://localhost:xxxx/api/student?id=1&name=steve>

public class StudentController : ApiController

{

public Student Post([FromBody]string name)

{

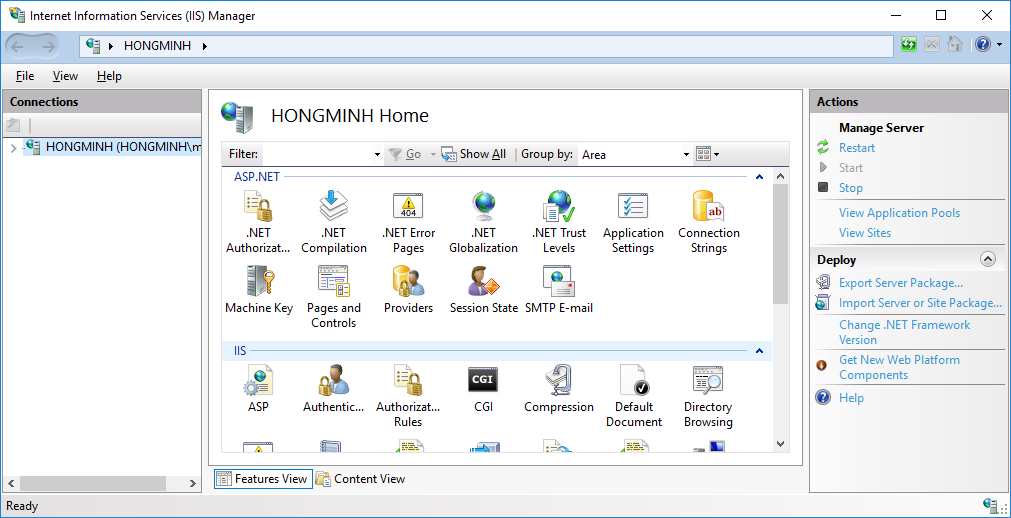
}

}

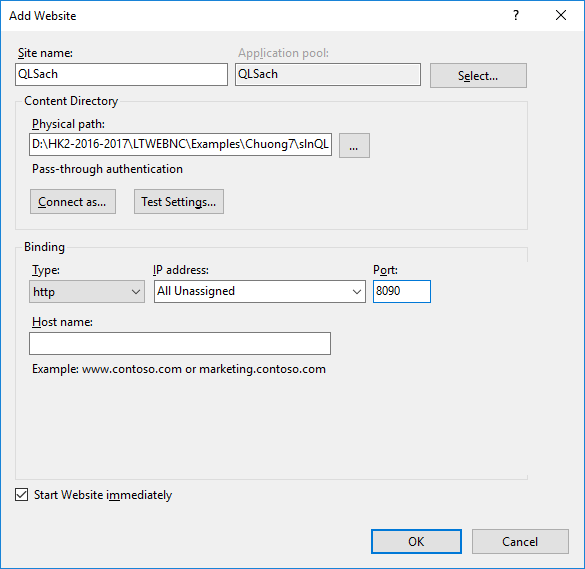
Web API lấy giá trị của tham số loại Student từ HTTP request body.

1. **Hướng dẫn dùng Internet Information Services IIS publish website.**

B1. Cài đặt IIS/Khởi động IIS



B2. Tạo mới website, khai báo Site name, Physical path và Port.



B3.Trong SQL Server, tạo login CSDL cho website mới tạo [IIS APPPOOL\Services], sau đó tạo User và gán quyền cần thiết cho user này. *(Trường hợp HOST IIS không dùng CSDL bỏ qua bước này)*

IF NOT EXISTS (SELECT name FROM sys.server\_principals WHERE name = 'IIS APPPOOL\Test1')

BEGIN

CREATE LOGIN [IIS APPPOOL\Test1]

FROM WINDOWS WITH DEFAULT\_DATABASE=[master],

DEFAULT\_LANGUAGE=[us\_english]

END

GO

CREATE USER [Test1User]

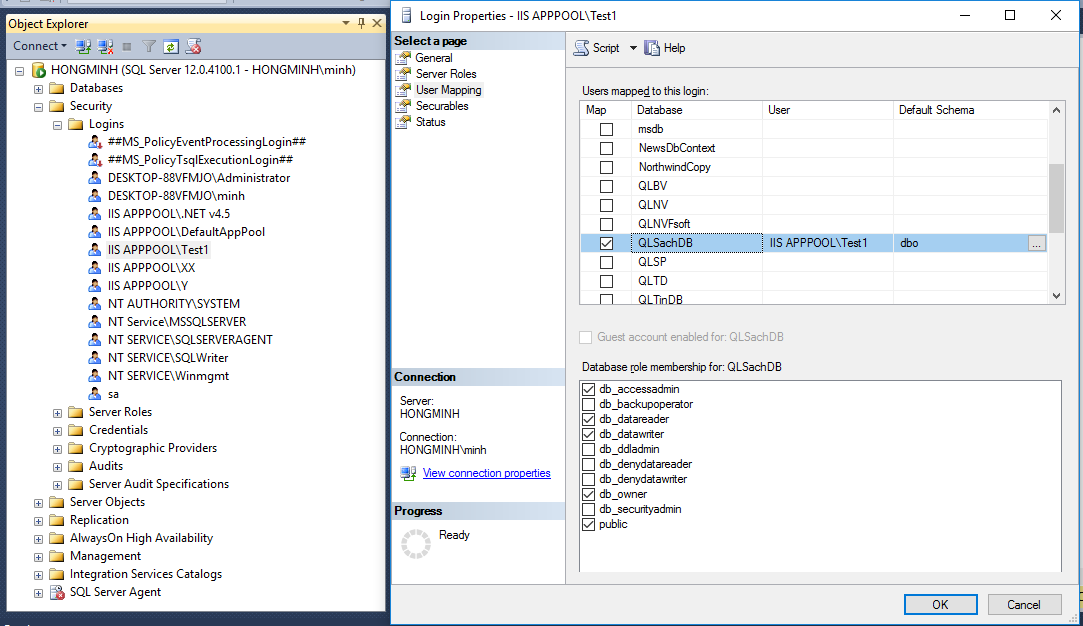
FOR LOGIN [IIS APPPOOL\Test1]

GO

EXEC sp\_addrolemember 'db\_owner', 'Test1User'

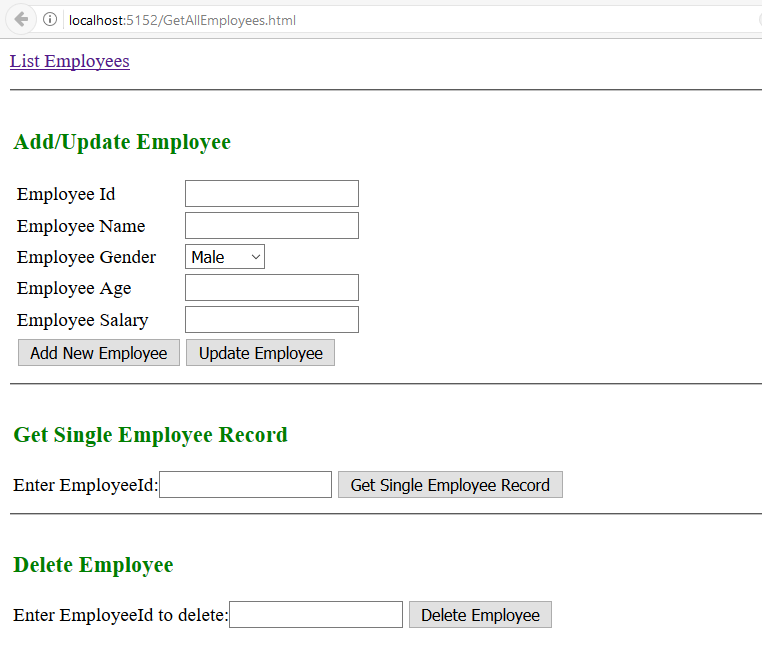
GO

B4. Gán các quyền truy cập (User Mapping) cho CSDL



**Bài 1:** Quản lý thông tin nhân viên (ID, Name, Gender, Age, Salary)

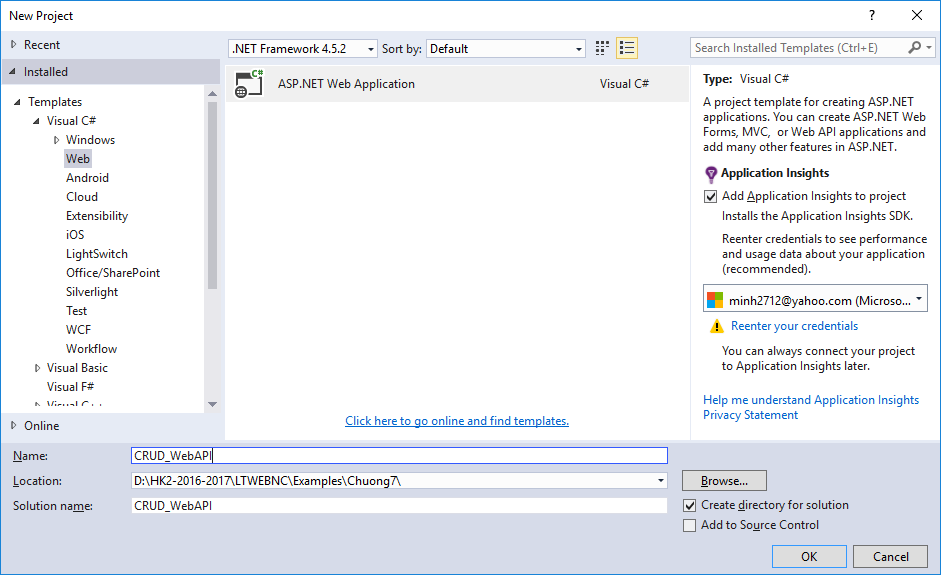
1. Tạo Web API Controller gồm các chức năng thêm mới, xóa, sửa và cập nhật.
2. Publish website với IIS Port 8888
3. Dùng jQuery (ASP.NET MVC hoặc HTML) thực hiện thao tác phía Client.
4. Ràng buộc kiểm tra dữ liệu nhập phía Client và Server.



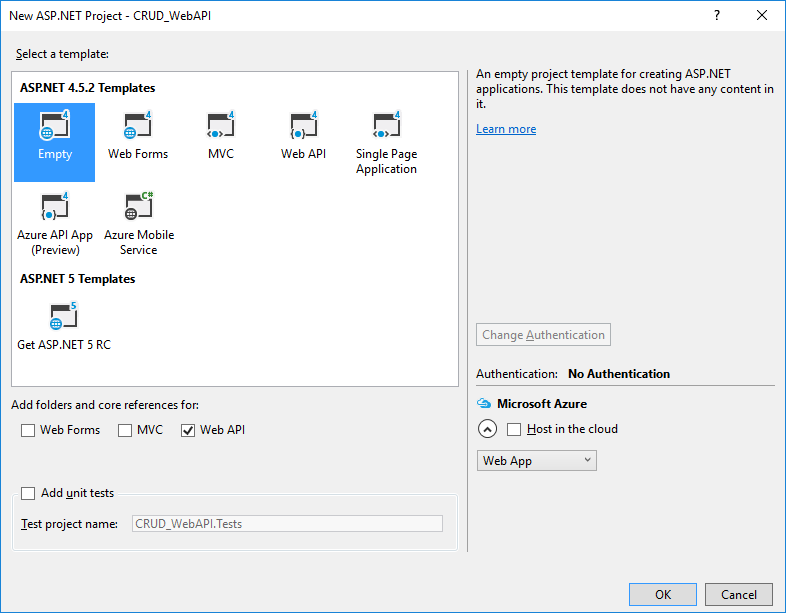
Hướng dẫn:

1. Tạo project Web API như sau:

File 🡪 New 🡪 Project:

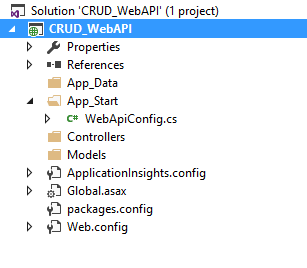


Nhập thông tin tên project, nhấn OK:



Chọn Empty project, check Web API 🡪 OK

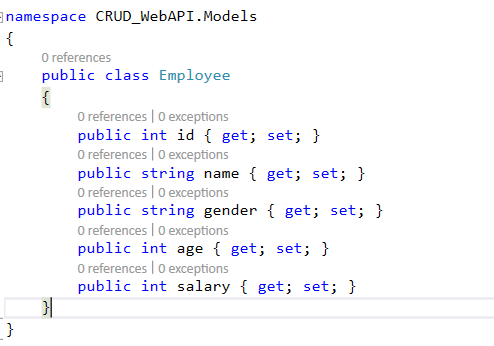
Cửa sổ Solution:



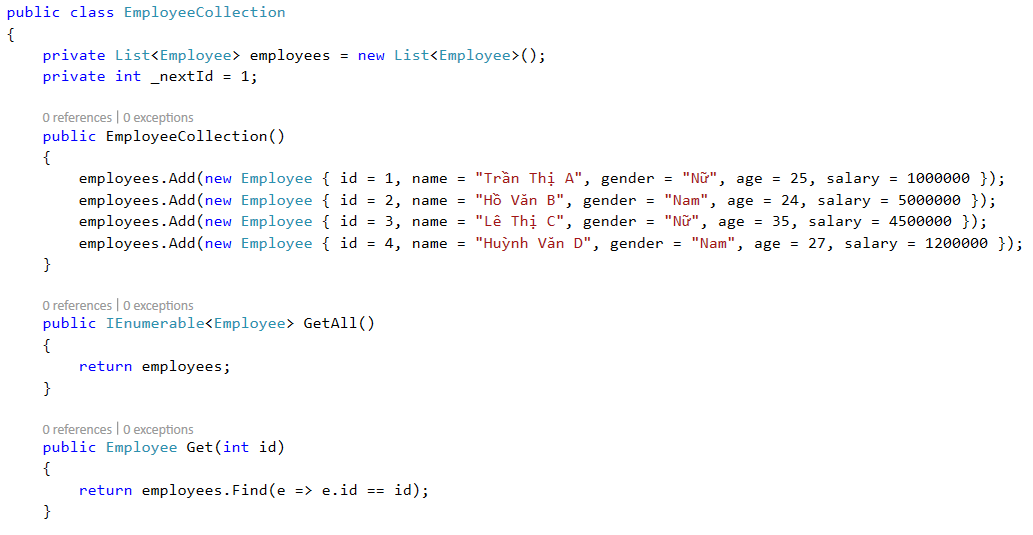
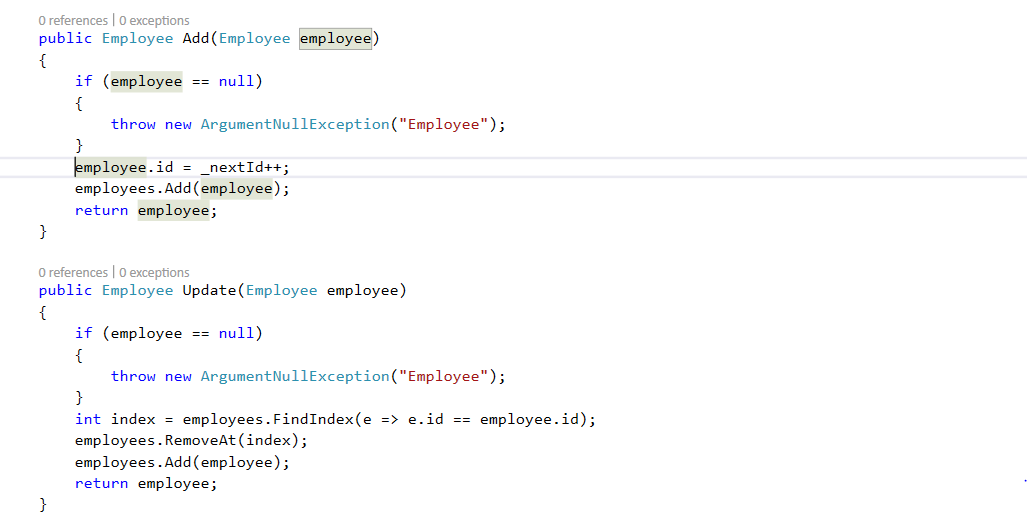
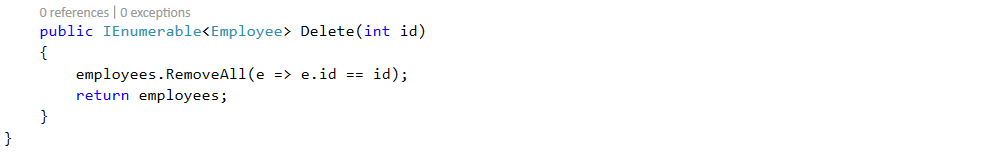
1. Xây dựng Model:

Model Employee:

Nhấp chuột phải vào thư mục Models 🡪 Add 🡪 Class: đặt tên Employee:

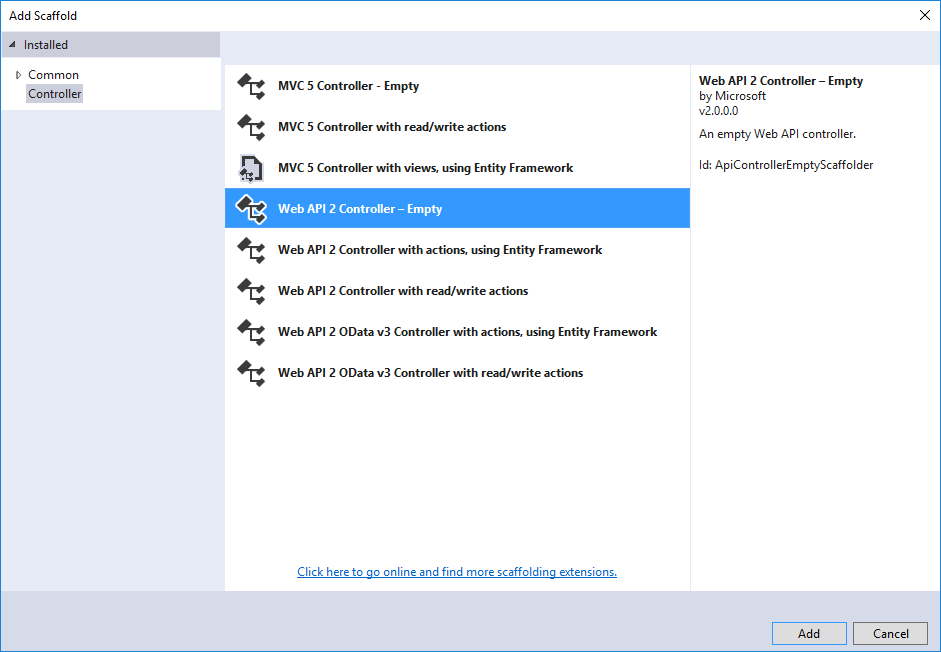


Tạo thêm class EmployeeCollection: để thực hiện các thao tác trên danh sách Employee như thêm, xóa, sửa, xem danh sách, xem Employee theo mã:

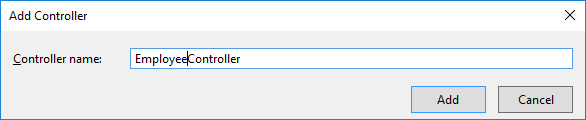
  

1. Tạo API Controller cung cấp CRUD:

Nhập phải chuột trên thư mục Controller 🡪 Add 🡪 Controller:



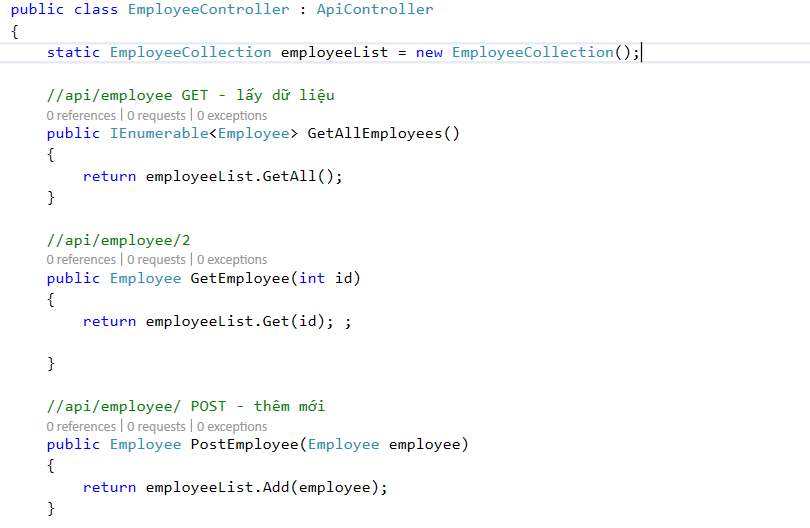
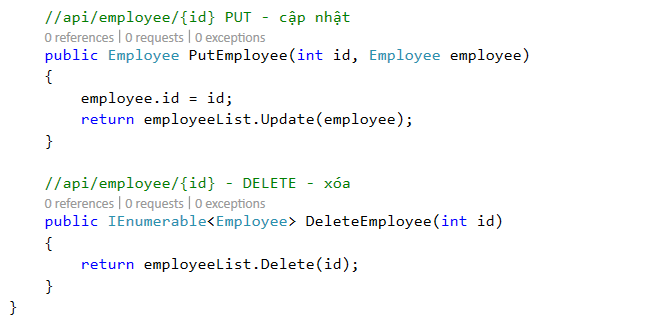
Chọn Web API 2 Controller – Empty 🡪 Add



Nhập tên Controller 🡪 Add

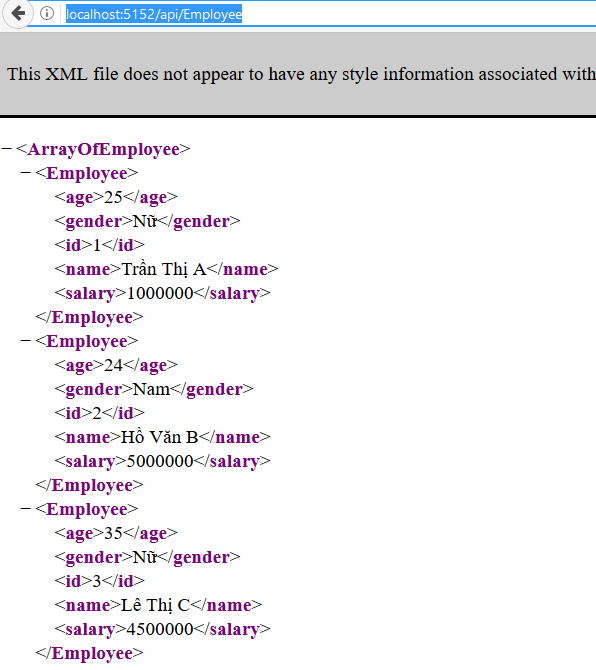
Viết các phương thức cung cấp CRUD services API:

Chú ý: lớp này thừa kế tứ ApiController

Nhấn F5 kiểm tra xem Services có hoạt động không?

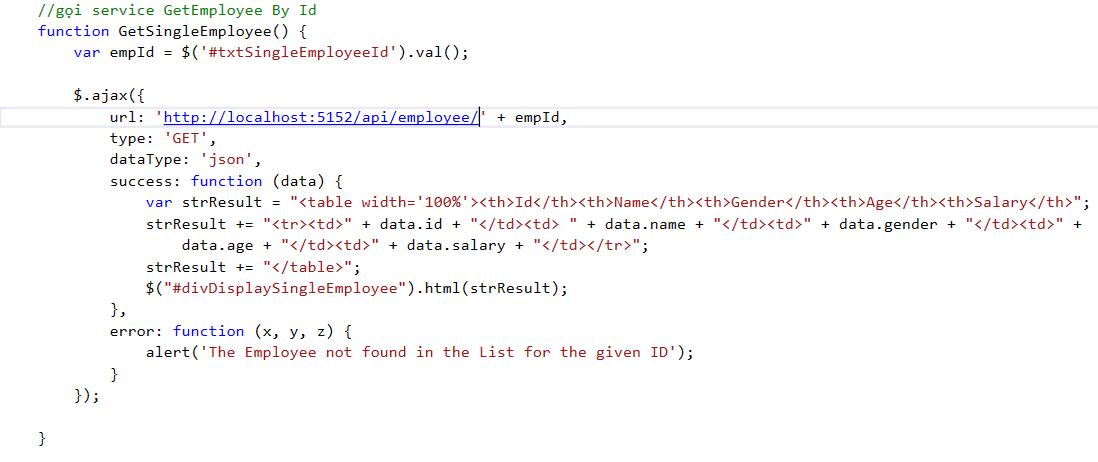
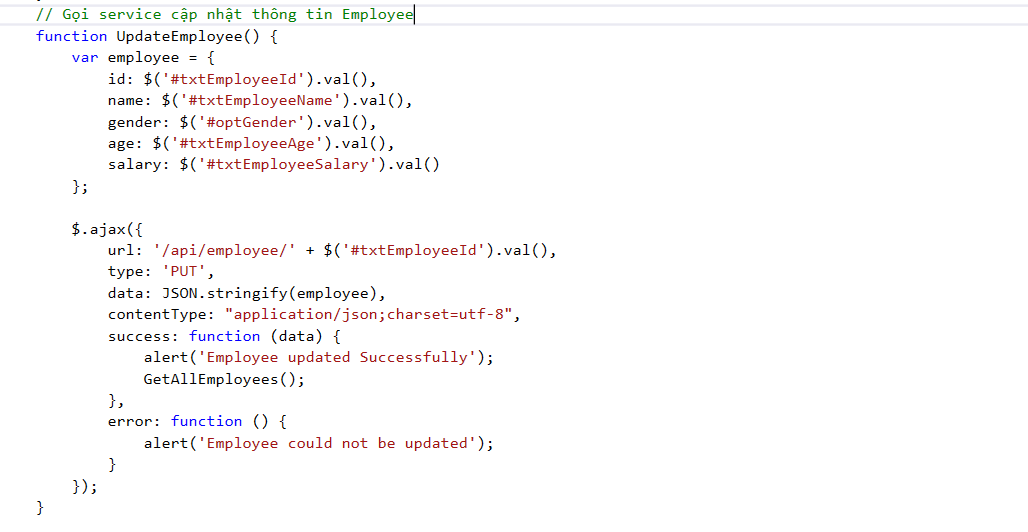
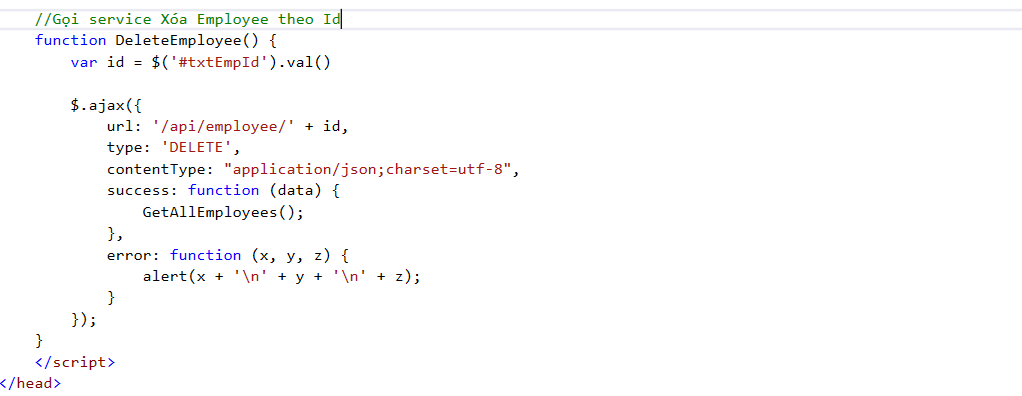
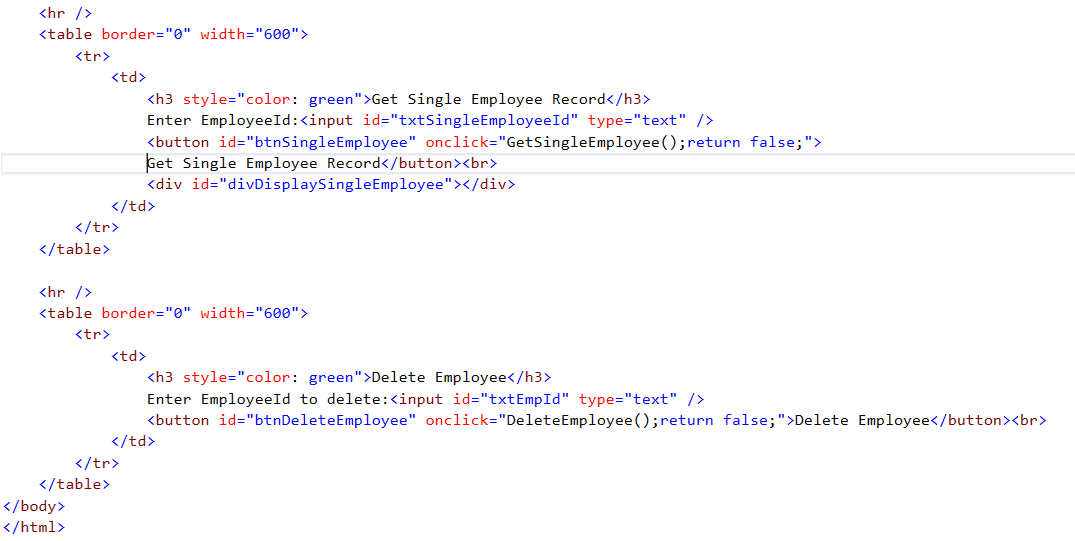
Nhập vào thanh địa chỉ như sau để gọi phương thức GetAll, nếu danh sách Employee hiển thị dạng XML thì service đã hoạt động:



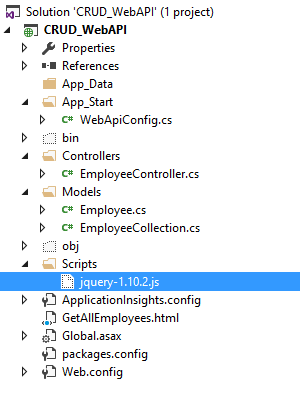
1. Tạo Web client truy xuất CRUD API:

Nhấp phải chuột vào project CRUD\_WebAPI 🡪 Add 🡪HTML Page 🡪 Nhập tên: GetAllEmployees.html 🡪OK:

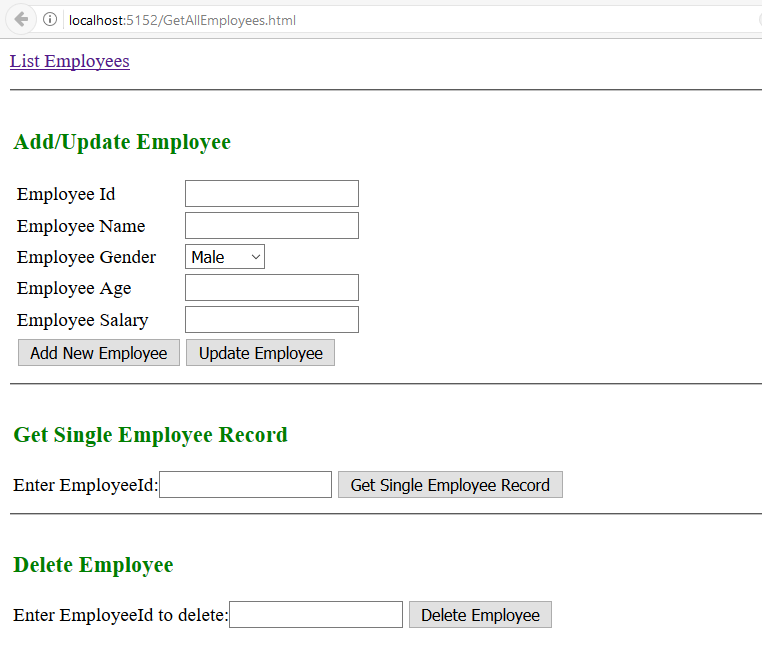
Code cho GetAllEmployees.html như sau:

Tạo thư mục Scripts trong project, copy thư viện jquery: để gọi trong trang html:



Chạy trang html để kiểm tra services:



**Bài 2:**

Thực hiện truy vấn dữ liệu dùng Odata và jQuery. Thông tin về sản phẩm bao gồm: ProductId, Name, Quantity***.***

B1. Tạo Model Product

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public int Quantity { get; set; }

}

B2. Tạo ProductCollection lưu thông tin của Product dùng List (thao tác này thay đổi với CSDL)

public class ProductCollection

{

private static IList<Product> listProducts;

static ProductCollection()

{

listProducts = new List<Product>();

var random = new Random();

for (var i = 0; i < 100; i++)

{

var product = new Product()

{

Id = i,

Name = "Product " + i,

Quantity = random.Next(1000)

};

listProducts.Add(product);

}

}

public IQueryable<Product> GetAll()

{

return listProducts.AsQueryable();

}

}

B3. Thực hiện Web API Controller với phương thức lấy dữ liệu từ Collection. Lưu ý gán thêm attribute [EnableQuery] cho phương thức này.

public class ProductsController : ApiController

{

private readonly ProductCollection pCollection = new ProductCollection();

[EnableQuery]

[Route("api/products")]

public IQueryable<Product> GetAll()

{

return pCollection.GetAll();

}

}

B4. Host ASP.NET Web API với IIS Port 7788

B5. Thao tác với OData và jQuery

Hàm jQuery tìm kiếm theo tên sản phẩm

function Search(criteria) {

$('#filter-list').html('ID - NAME - QUANTITY');

$.getJSON("http://localhost:7788/api/products?$filter=substringof('" + criteria + "', Name) eq true",

function (data) {

$.each(data, function (key, val) {

var str = val.Id + ' - ' + val.Name + ' - ' + val.Quantity;

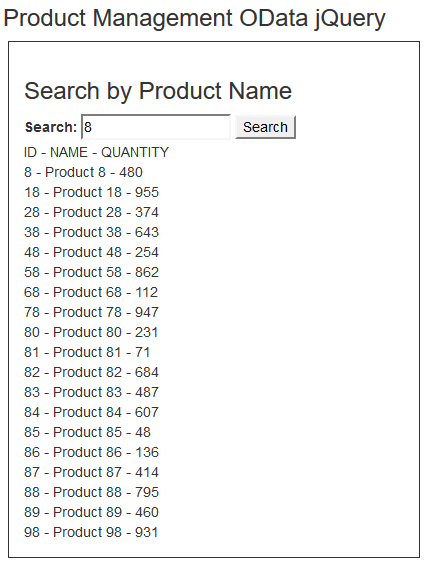
$('<div/>', { html: str })

.appendTo($('#filter-list'));

});

});

}



Hàm jQuery lấy 5 sản phẩm đầu tiên, kết quả sắp xếp tăng dần theo Quantity

function GetTopFive() {

$('#top-quantity-list').html('ID - NAME - QUANTITY');

$.getJSON("http://localhost:7788/api/products?$top=5&$orderby=Quantity desc",

function (data) {

$.each(data, function (key, val) {

var str = val.Id + ' - ' + val.Name + ' - ' + val.Quantity;

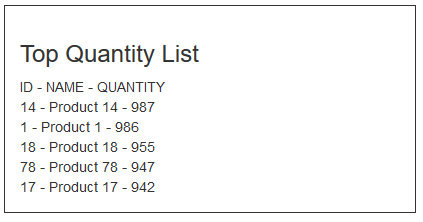
$('<div/>', { html: str })

.appendTo($('#top-quantity-list'));

});

});

}



Hàm jQuery phân trang dữ liệu

function GetProducts(pageId, pageSize) {

$('#product-list').html('ID - NAME - QUANTITY');

var skipCount = (pageId - 1) \* pageSize;

$.getJSON("http://localhost:7788/api/products?$skip=" + skipCount + "&$top=" + pageSize,

function (data) {

$.each(data, function (key, val) {

var str = val.Id + ' - ' + val.Name + ' - ' + val.Quantity;

$('<div/>', { html: str })

.appendTo($('#product-list'));

});

var prev = pageId > 1 ? pageId - 1 : pageId;

var next = data.length > 0 ? pageId + 1 : pageId;

var str = '<a href="#product-list" onclick="GetProducts(' + prev

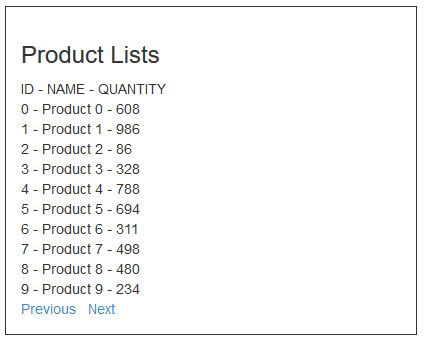
+ ',10)">Previous</a> &nbsp; <a href="#product-list" onclick="GetProducts('

+ next + ',10)">Next</a>';

$('#product-paging').html(str);

});

}



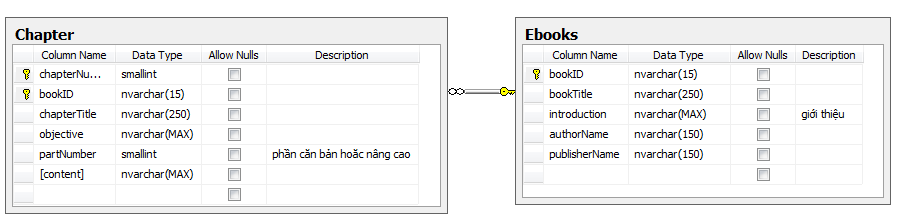
**Bài 3: D**ùng code first (3tier) + web API để xây dựng website quản lý nội dung các quyển sách với các yêu cầu như sau:

* Phía người quản trị website:
  + Cho phép nhập sách, thêm sách, xóa sách
  + Cho phép thêm chương mới vào sách, xóa chương và sửa thông tin chương
* Phía người dùng:
  + Cho phép xem danh sách các quyển sách
  + Cho phép xem thông tin chi tiết các chương của 1 quyển sách tương ứng.

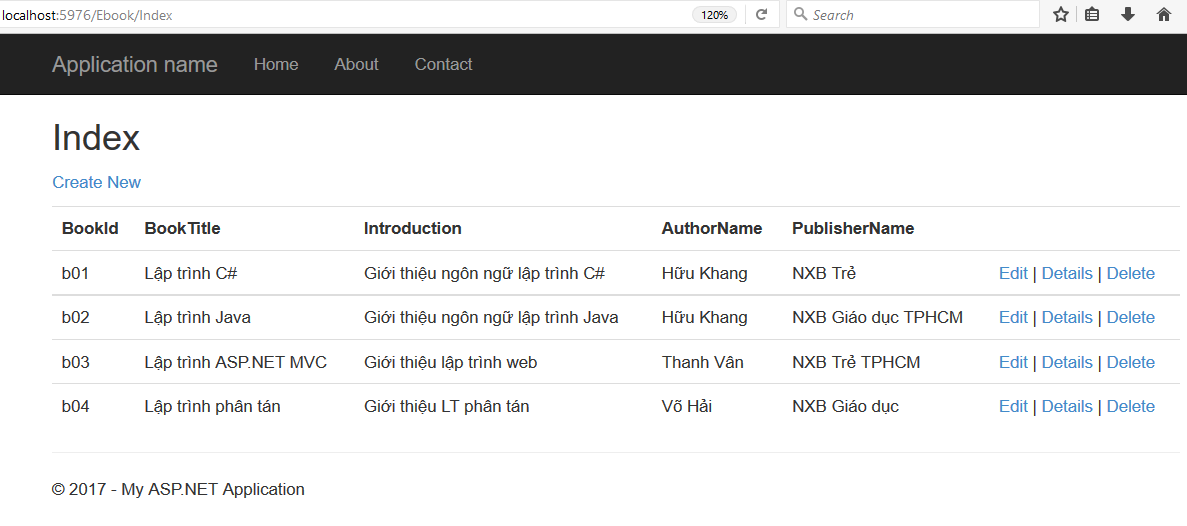
**Yêu cầu code:**

* Xây dựng services cung cấp các phương thức (dùng codefirst + 3 tier):
  + EbookApiController: Thêm sách, sửa thông tin sách, xóa sách, lấy danh sách các quyển sách
  + ChapterApiController: Thêm chương, sửa thông tin chương, xóa chương, xem danh sách các chương.
* Xây dựng project web application thực hiện theo yêu cầu như trên, sử dụng các services đã cung cấp.

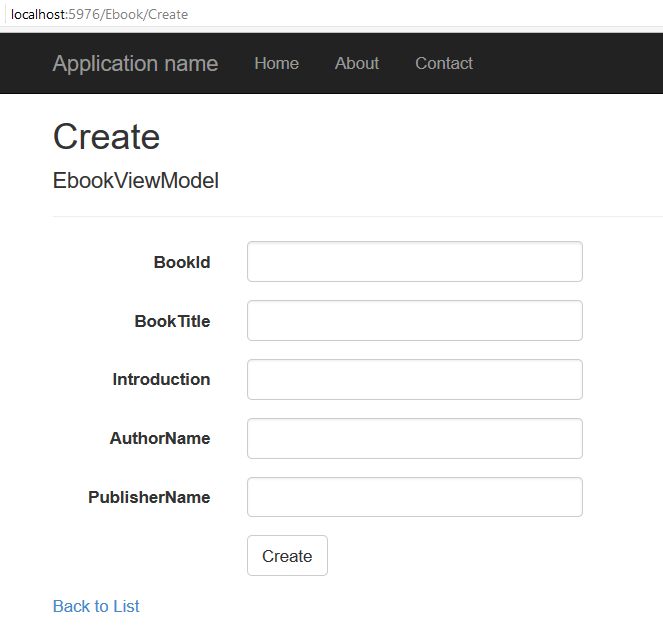
**CSDL gợi ý:**



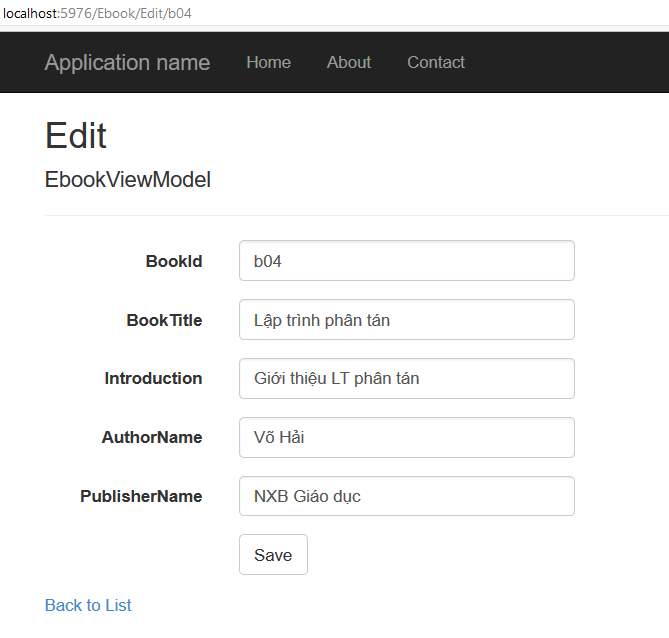
Giao diện website trang Index:



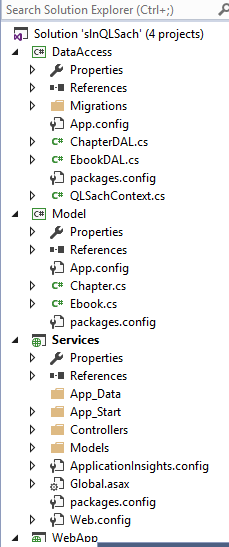
Trang Create:



Trang Edit:

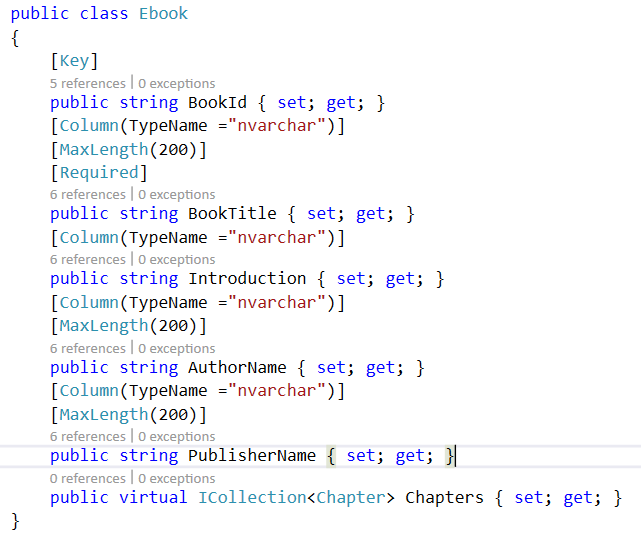
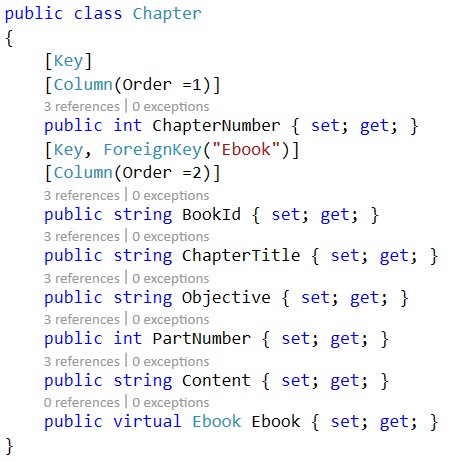


Hướng dẫn:



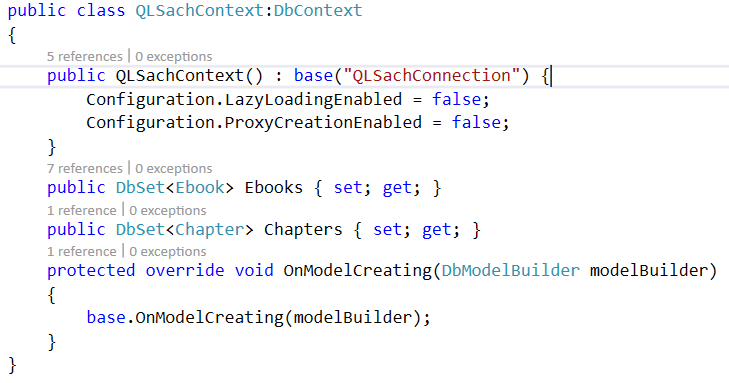
1. Tạo blank solution **slnQLSach**.
2. Tạo 2 project Class Library: **Model** (Entities) + **DataAccess**. Install Package *EntityFramework*

Code gợi ý cho Model:

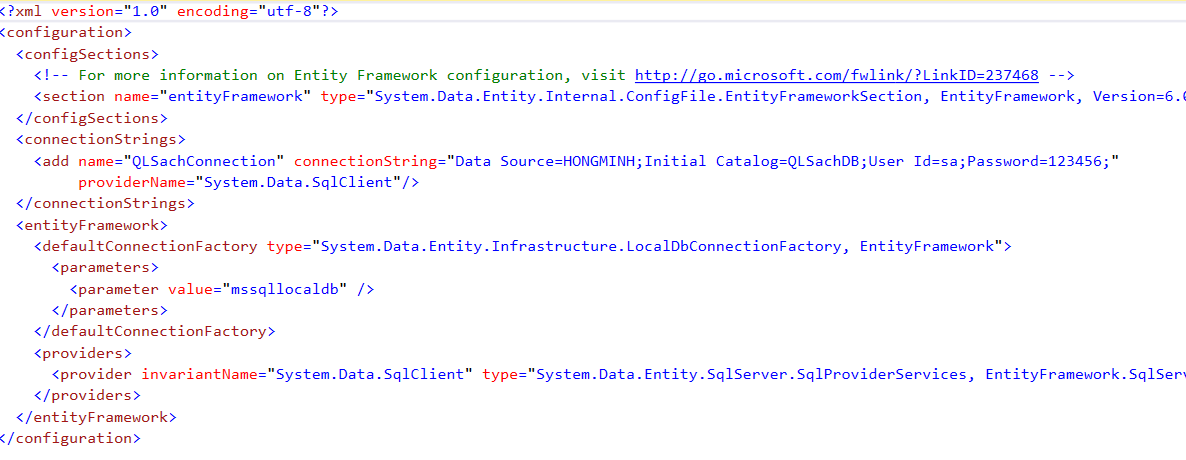
 

Code gợi ý cho DataAccess:

**Class QLSachContext:**



**File App.Config: (nhớ copy connectionString qua tập tin Web.Config bên Services project)**



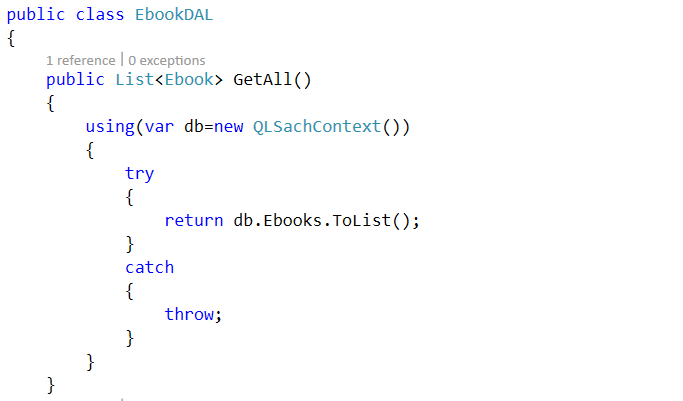
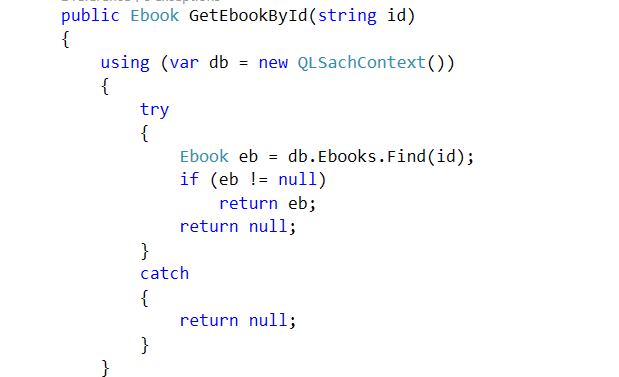
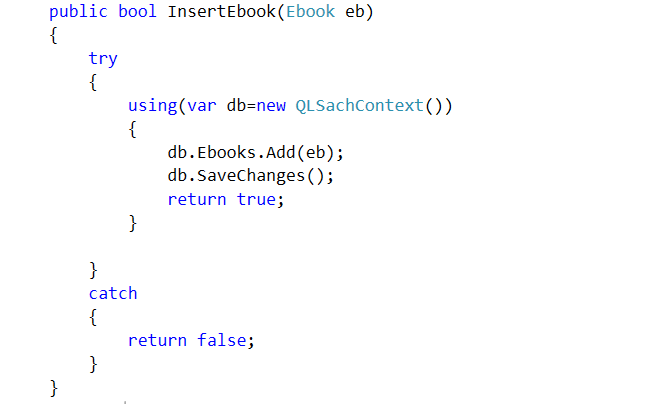
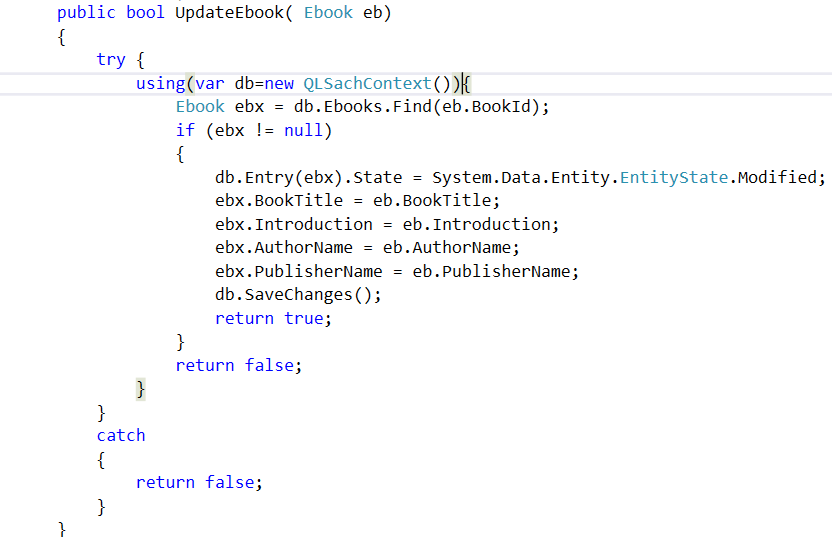
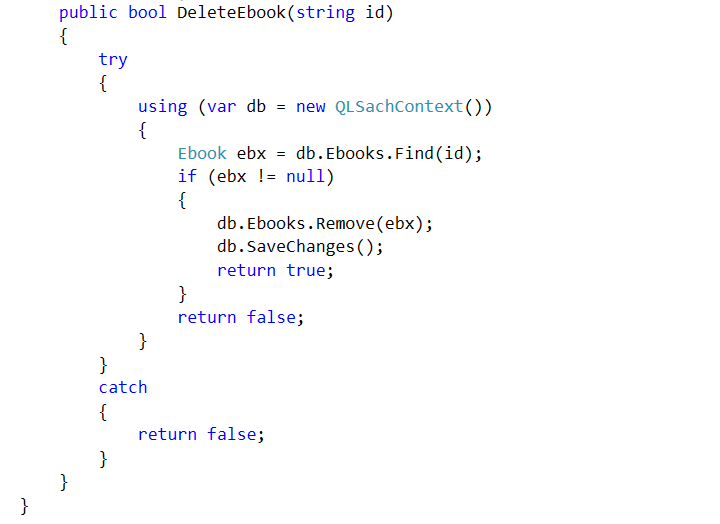
Tạo database: mở package console chạy các lệnh:

1/enable-migrations

2/add-migration InitDB

3/update-database

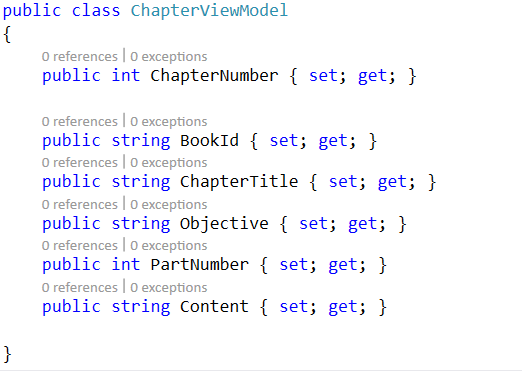
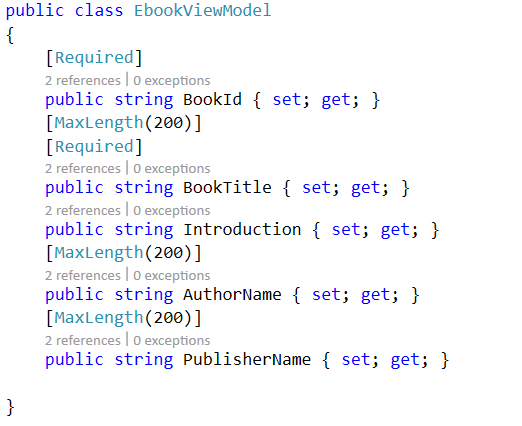
**Class EbookDAL:**

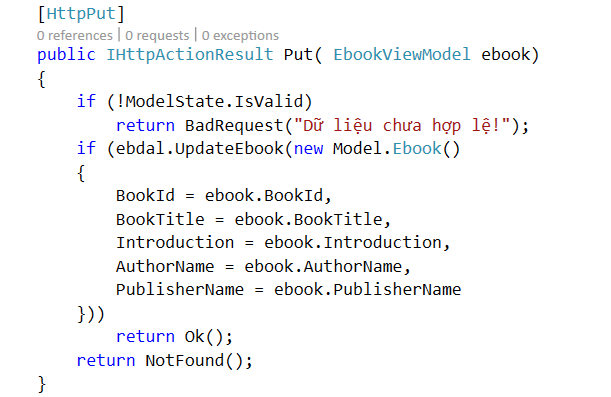
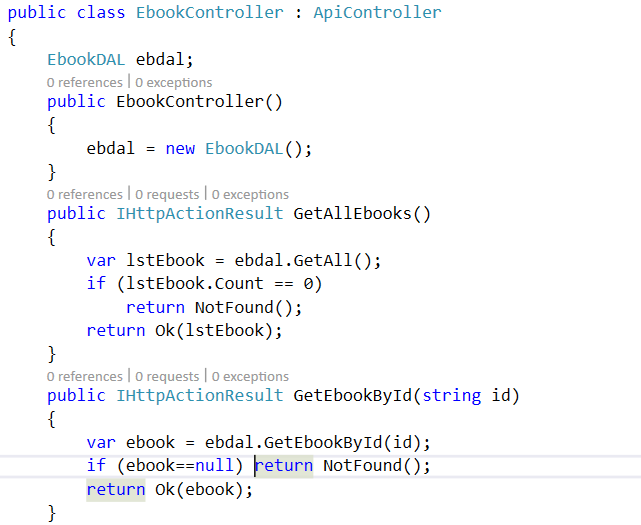
1. Tạo Empty ASP.NET Web Application Project **Services** với template của Web API thực hiện các services thực hiện thêm, xóa, sửa, liệt kê dữ liệu.

Code gợi ý cho ApiController:

* Code các tập tin EbookViewModel, ChapterViewModel trong thư mục Model như sau:



* Code EbookController:



Mở tập tin Web.Config thêm ConnectionString. (copy từ App.Config của DataAccess)

1. **(Có thể tách ra 1 project độc lập với Services)** Tạo ASP.NET Web Application Project **WebApp** với template của MVC.
   * Install Package *Microsoft.AspNet.WebApi.Client* (mở Package console)
   * Tạo **EbookViewModel** và **ChapterViewModel** vào thư mục Model giống tầng **Services**.
   * Tạo EbookController:Controller gọi các phương thức được cung cấp từ Services như: GetAll, GetById, PostEbook, PutEbook, DeleteEbook như sau:

public class EbookController : Controller

{

// GET: Ebook

public ActionResult Index()

{

IEnumerable<EbookViewModel> ebooks = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:34377/api/");

//HTTP GET

var responseTask = client.GetAsync("Ebook");

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode) {

var readTask = result.Content.ReadAsAsync<IList<EbookViewModel>>();

readTask.Wait();

ebooks = readTask.Result;

}

else //web api sent error response

{ //log response status here..

ebooks = Enumerable.Empty<EbookViewModel>();

ModelState.AddModelError(string.Empty, "Server error.");

}

}

return View(ebooks);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(EbookViewModel ebook)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:34377/api/Ebook");

//HTTP POST

var postTask = client.PostAsJsonAsync<EbookViewModel>("Ebook", ebook);

postTask.Wait();

var result = postTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

ModelState.AddModelError(string.Empty, "Server Error. Please contact administrator.");

return View(ebook);

}

//HTTP GET

public ActionResult Edit(string id)

{

EbookViewModel ebook = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:34377/api/");

//HTTP GET

var responseTask = client.GetAsync("Ebook?id=" + id);

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<EbookViewModel>();

readTask.Wait();

ebook = readTask.Result;

}

}

return View(ebook);

}

[HttpPost]

public ActionResult Edit(EbookViewModel ebook)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:34377/api/");

//HTTP POST

var putTask = client.PutAsJsonAsync<EbookViewModel>("Ebook", ebook);

putTask.Wait();

var result = putTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

return View(ebook);

}

public ActionResult Delete(string id)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:34377/api/");

//HTTP DELETE

var deleteTask = client.DeleteAsync("Ebook/" + id);

deleteTask.Wait();

var result = deleteTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

return RedirectToAction("Index");

}

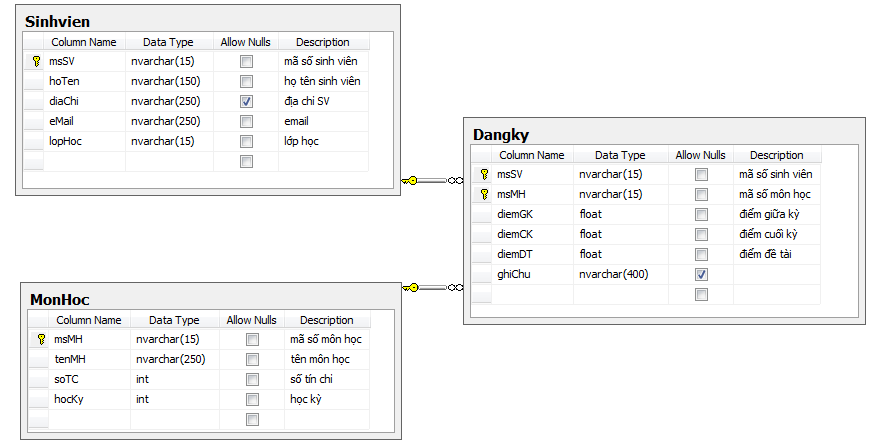
}

* + **Phần View: sinh viên tự code ☺**

**Bài 4: (Dùng codefirst + Web API)** xây dựng ứng dụng web quản lý sinh viên đơn giản với các chức năng như sau:

* Phía người dùng giáo vụ:
  + Cho phép nhập, xóa, sửa thông tin sinh viên khi có yêu cầu
  + Cho phép mở lớp học phần (môn học), xóa lớp học phần, sửa thông tin lớp học phần
* Phía người dùng sinh viên:
  + Cho phép xem danh sách lớp học phần
  + Cho phép đăng ký môn học theo lớp học
* Phía người dùng giảng viên:
  + Cho phép xem danh sách sinh viên theo lớp học
  + Cho phép nhập điểm cho tất cả sinh viên đối với 1 môn học

CSDL gợi ý:



**Bài 5**: (dùng code first + web API) xây dựng ứng dụng web thi trắc nghiệm với các yêu cầu sau:

* Phía người dùng giảng viên:
  + Đăng nhập theo tài khoản được cấp.
  + Cho phép thêm, xóa, sửa thông tin môn học.
  + Cho phép xem danh sách câu hỏi theo môn học.
  + Nhập câu hỏi trắc nghiệm, đáp án theo môn học
  + Sửa, xóa câu hỏi khi không cần thiết
* Phía người dùng sinh viên:
  + Cho phép thi thử theo môn học
  + Cho phép xem kết quả sau khi thi xong

CSDL gợi ý:

