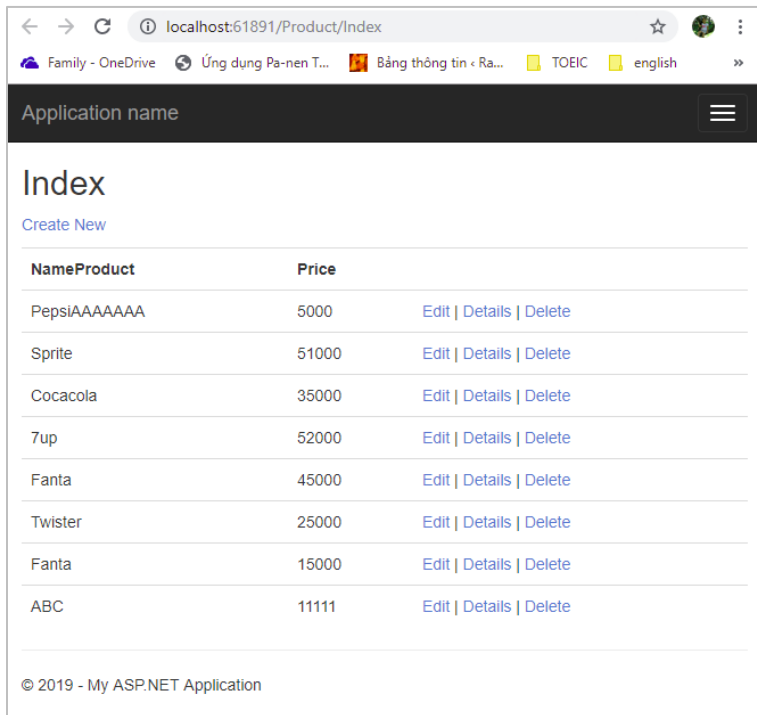


CodeFirst WebAPI CRUD Operations:

Thực hiện Website quản lý sản phẩm

Tạo solution chứa 3 project riêng nhau (SlnCodeFristAPIProduct, PrjProductMVC, DTOs)

Thực hiện website như sau

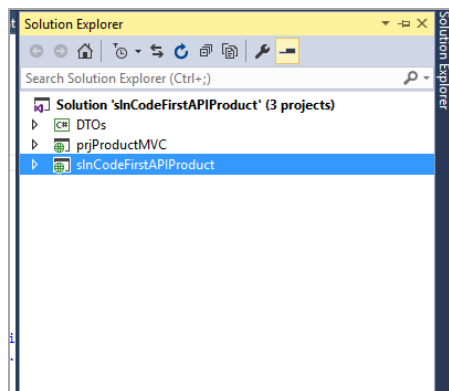


NameProduct	Price	
PepsiAAAAAAA	5000	Edit Details Delete
Sprite	51000	Edit Details Delete
Cocacola	35000	Edit Details Delete
7up	52000	Edit Details Delete
Fanta	45000	Edit Details Delete
Twister	25000	Edit Details Delete
Fanta	15000	Edit Details Delete
ABC	11111	Edit Details Delete

© 2019 - My ASP.NET Application

1. Bước 1 tạo solution `slnCodeFirstAPIProduct`

Tạo solution chứa 3 project riêng nhau (`SlnCodeFristAPIProduct`, `PrjProductMVC`, `DTOs`)



2. Bước 2:

2.1. Tạo Project `SlnCodeFristAPIProduct` empty

2.1.1. Trong thư mục Model;

- Tạo class `Product.cs`

```
public class Product
{
    public int ID { get; set; }
    public string NameProduct { get; set; }
    public double Price { get; set; }
}
```

- Cài đặt *EntityFramework* cho project (*Nudget*)
- Cài đặt *Web API Client* (*Nudget*)

- Tạo ProductDB.cs: DbContext

```
using System.Data.Entity;
using SlnCodeFirstAPIProduct.Models;
```

(Chú ý: Khai báo namespace

)

```
public class ProductDB:DbContext
{
    public ProductDB() : base("name=ProductDB")
    {
    }
    public System.Data.Entity.DbSet<SlnCodeFirstAPIProduct.Models.Product> Products { get; set; }
}
```

- Config chuỗi kết nối: (dùng 1 databaseEF để phát sinh connectionstring sau đó sửa lại theo database của project)

```
<connectionStrings>
  <add name="ProductDB" connectionString="data source=THUHA\SQLEXPRESS;initial catalog=ProductDB;integrated security=True;" providerName="System.Data.SqlClient" />
</connectionStrings>
```

- Tạo 1 class DataInitiaier.cs khởi tạo dữ liệu mẫu theo Model Product.cs

```
public class DataInitiaier:
    System.Data.Entity.CreateDatabaseIfNotExists<ProductDB>
{
    protected override void Seed(ProductDB context)
    {
        context.SaveChanges();

        context.Products.Add(new Product { NameProduct = "Pepsi", Price = 5000 });
        context.Products.Add(new Product { NameProduct = "Sprite", Price = 51000 });
        context.Products.Add(new Product { NameProduct = "Cocacola", Price = 35000 });
        context.Products.Add(new Product { NameProduct = "7up", Price = 52000 });
        context.Products.Add(new Product { NameProduct = "Fanta", Price = 45000 });
        context.Products.Add(new Product { NameProduct = "Twister", Price = 25000 });
        context.Products.Add(new Product { NameProduct = "Fanta", Price = 15000 });

        base.Seed(context);
    }
}
```

- Thực thi các lệnh để migration database
 - + enable-migrations
 - +add-migration DataInitiaier
 - +update-database
- Run project SlnCodeFristAPIProduct để tạo Database

2.1.2. (Bước trung gian tạo classlibrary DTOs)

- Right Click lên solution chọn Add → New Project → Class library: DTOs
- Class trung gian giữa 2 project để truyền dữ liệu từ service về MVC (client)
- Class ProductDTO (giống class Produc.cs trong SlnCodeFristAPIProduct đã tạo): ...
- Trong project SlnCodeFristAPIProduct add Reference DTOs

2.1.3. Trong thư mục Controller

- Tạo controller empty là ProductController (APIcontroller)

- Trong ProductController thực hiện các IHttpActionResultResult là các services tương ứng với các chức năng trên View của project PrjProductMVC là CRUD Action.

```
public class ProductController : ApiController
{
    // GET: api/Product
    public IHttpActionResult GetAllProducts()
    {
        IList<ProductDTO> products = null;
        using (var ctx = new ProductDB())
        {
            products = ctx.Products.Select(p => new ProductDTO()
            {
                ID = p.ID,
                NameProduct = p.NameProduct,
                Price = p.Price
            }).ToList<ProductDTO>();
        }
        if(products.Count==0)
        {
            return NotFound();
        }
        return Ok(products);
    }

    // GET: api/Product/5
    public IHttpActionResult GetProductById(int id)
    {
        ProductDTO productDTO = null;
        using (var ctx = new ProductDB())
        {
            productDTO = ctx.Products.Where(p => p.ID == id)
                .Select(p => new ProductDTO()
                {
                    ID=p.ID,
                    NameProduct=p.NameProduct,
                    Price=p.Price
                }).FirstOrDefault<ProductDTO>();
        }
        if (productDTO == null)
        {
            return NotFound();
        }
        return Ok(productDTO);
    }

    //POST: api/Product
    private Product GetNewProduct(ProductDTO p)
    {
        Product product = new Product()
        {
            NameProduct = p.NameProduct,
            Price = p.Price,
        };
        return product;
    }
}
```

```

public IHttpActionResult PostNewProduct(ProductDTO productDTO)
{
    if (!ModelState.IsValid)
        return BadRequest("Invalid data");
    using (var ctx = new ProductDB())
    {
        Product product = GetNewProduct(productDTO);
        if (product != null)
        {
            ctx.Products.Add(product);
            ctx.SaveChanges();
            return Ok();
        }
        else
            return BadRequest("Invalid data");
    }
}

private bool UpdateProduct(Product p, ProductDTO pDTO)
{
    p.NameProduct = pDTO.NameProduct;
    p.Price = pDTO.Price;

    return true;
}

}

public IHttpActionResult PutProduct(ProductDTO product)
{
    if (!ModelState.IsValid)
        return BadRequest("Invalid data");
    using (var ctx = new ProductDB())
    {
        var existingPro = ctx.Products.Where(p => p.ID == product.ID).FirstOrDefault<Product>();
        if (existingPro != null)
        {
            if (UpdateProduct(existingPro, product))
            {
                ctx.SaveChanges();
                return Ok();
            }
            return BadRequest("Not a valid model");
        }
        return NotFound();
    }
}

public IHttpActionResult Delete(int id)
{
    if (id < 0)
        return BadRequest("Not a valid id");
    using (var ctx = new ProductDB())
    {
        var pro = ctx.Products.Where(p => p.ID == id).FirstOrDefault();
        ctx.Entry(pro).State = System.Data.Entity.EntityState.Deleted;
        ctx.SaveChanges();
    }
    return Ok();
}
}

```

- 2.1.4. Chạy slnCodeFristAPIProduct trên trình duyệt test thử API có thực thi được hay ko?
- 2.1.5. Nếu chạy được (WebAPI với các service tương ứng đã hoàn thành) chuyển sang bước 3,

3. Bước 3:

2.1. Trên project PrjProductMVC

- Trong project PrjProductMVC add Reference DTOs và slnCodeFristAPIProduct

3.1.1. Trong thư mục Controller tạo controller empty ProductController

- Trong ProductController thực hiện các ActionResult CRUD bằng cách gọi các Service trên API vừa tạo ở bước 2

```
public ActionResult Index()
{
    IEnumerable<ProductDTO> products = null;
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("http://localhost:55317/api/");
        var responseTask = client.GetAsync("product");
        responseTask.Wait();
        var result = responseTask.Result;
        if (result.IsSuccessStatusCode)
        {
            var readTask = result.Content.ReadAsAsync<IList<ProductDTO>>();
            readTask.Wait();
            products = readTask.Result;
        }
        else
        {
            products = Enumerable.Empty<ProductDTO>();
            ModelState.AddModelError(string.Empty, "Server error");
        }
    }

    return View(products);
}
```

```

public ActionResult Create()
{
    return View();
}
[HttpPost]
public ActionResult Create(ProductDTO product)
{
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("http://localhost:55317/api/");
        var postTask = client.PostAsJsonAsync<ProductDTO>("product", product);
        postTask.Wait();
        var result = postTask.Result;
        if (result.IsSuccessStatusCode)
        {
            return RedirectToAction("Index");
        }
        else
            ModelState.AddModelError(string.Empty, "Server error");
    }

    return View(product);
}

public ActionResult Edit(int id)
{
    ProductDTO pro = null;

    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("http://localhost:55317/api/");
        var responseTask = client.GetAsync("product?id="+id.ToString());
        responseTask.Wait();
        var proResult = responseTask.Result;
        if (proResult.IsSuccessStatusCode)
        {
            var readTask = proResult.Content.ReadAsAsync<ProductDTO>();
            readTask.Wait();
            pro = readTask.Result;
        }
        else
            ModelState.AddModelError(string.Empty, "Server error");
    }

    return View(pro);
}

[HttpPost]
public ActionResult Edit(ProductDTO pro)
{
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("http://localhost:55317/api/");
        //http post
        var putTask = client.PutAsJsonAsync<ProductDTO>("product", pro);
        putTask.Wait();
        var result = putTask.Result;
        if (result.IsSuccessStatusCode)
        {
            return RedirectToAction("Index");
        }
    }

    return View(pro);
}

```

- SV tự tạo các View tương ứng

4. Bước 4: Chạy Project

Thiết lập multistartup cho cả solution

Right_click on the solution → chọn Properties →

