

Упражнения: ASP.NET MVC като потребителски интерфейс

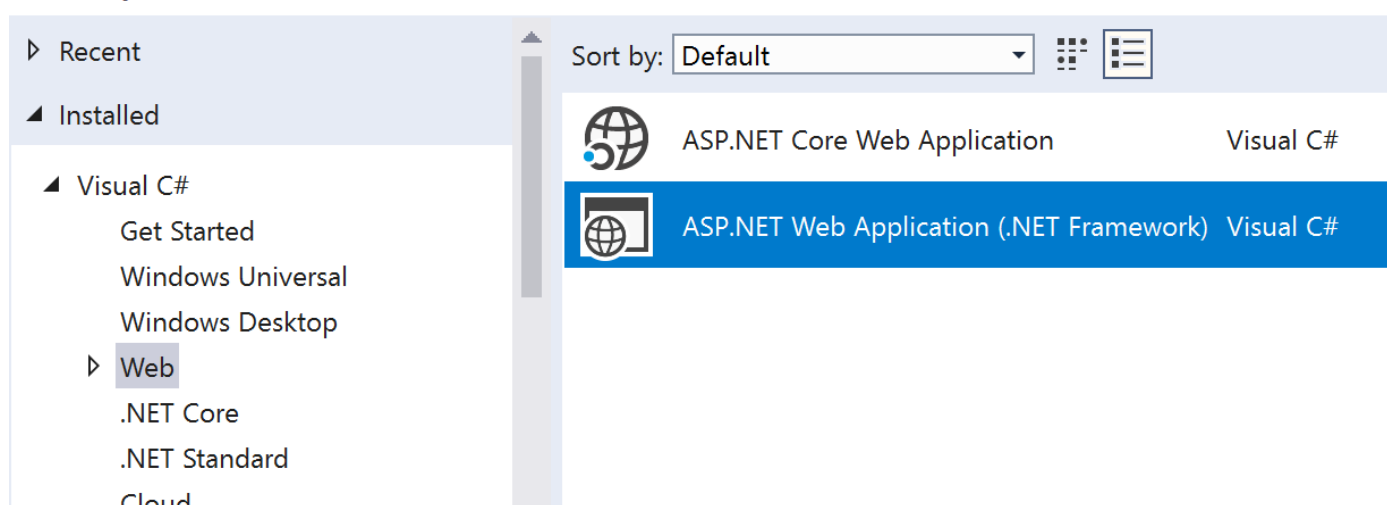
Създаване на просто приложение

В рамките на това упражнение ще направим трислойно приложение, като слоевете за данни и услуги ще са аналогични с досега направените, но за презентационен слой ще използваме ASP.NET MVC.

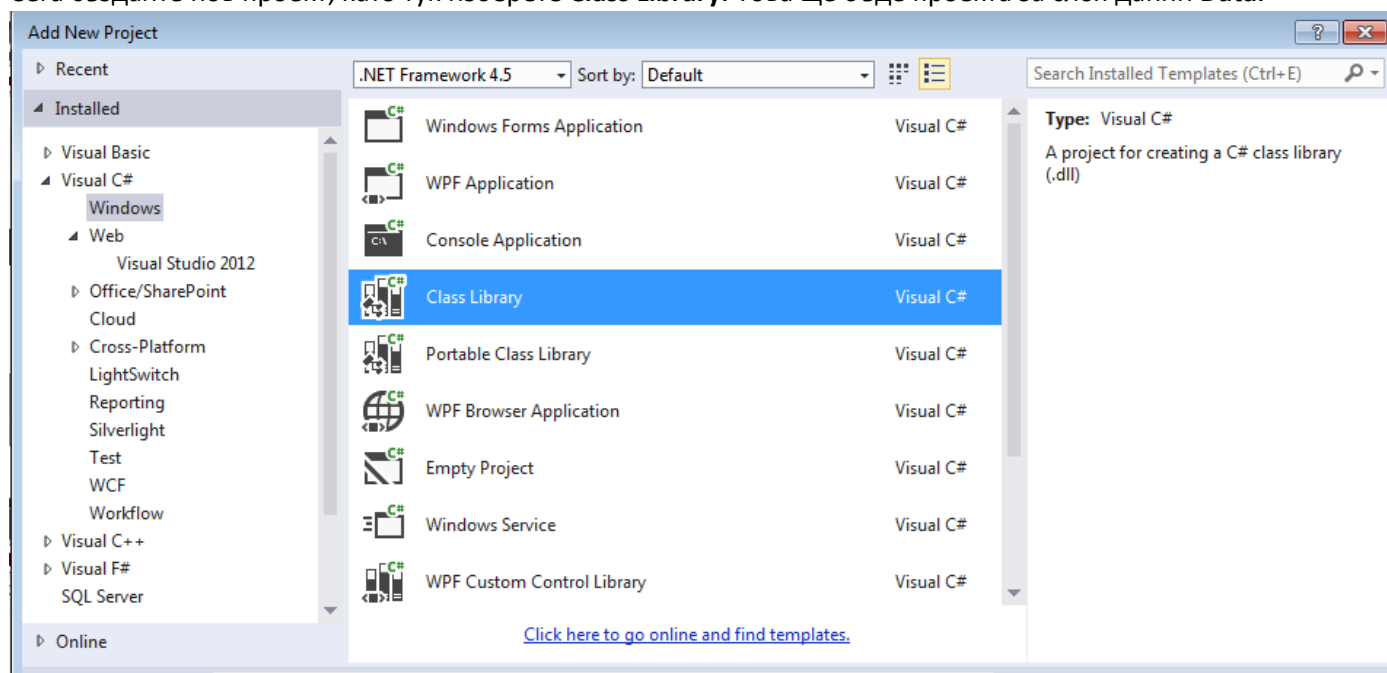
1. Структура на проекта

Започнете със създаване на **ASP.NET Web Application** проект **ProductApp**. Използвайте **Visual C# -> Web -> ASP.NET Web Application**

New Project



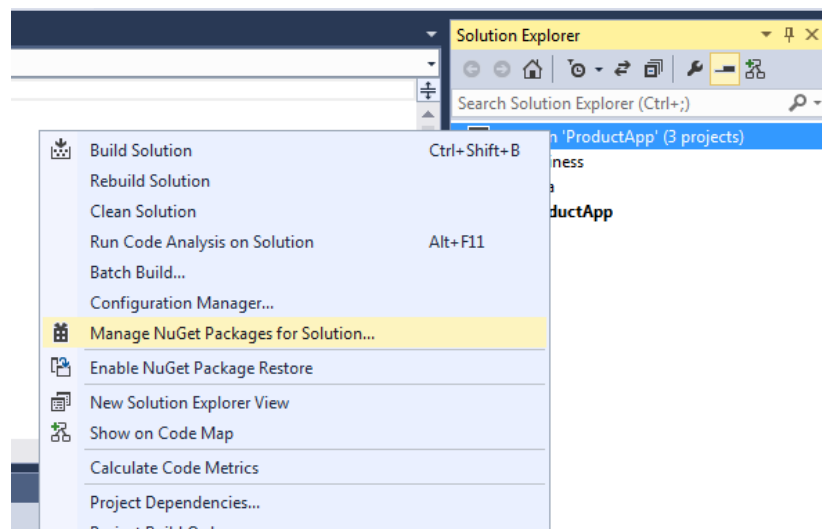
Сега създайте нов проект, като тук изберете **Class Library**. Това ще бъде проекта за слоя данни **Data**.



По сходен начин ще създадем и проект **Business**. Използваме **Class Library**, тъй като във всеки от тези проекти създаваме класове, които да могат да бъдат достъпвани от други проекти.

2. Добавяне на EntityFramework и референции

След като сме създали всички проекти е време да добавим **EntityFramework**. Цъкнете с десен бутон върху solution-а и изберете **Manage NuGet Packages for Solution** и инсталирайте **EntityFramework**. Изберете инсталация за **всичките** проекти в solution-а.

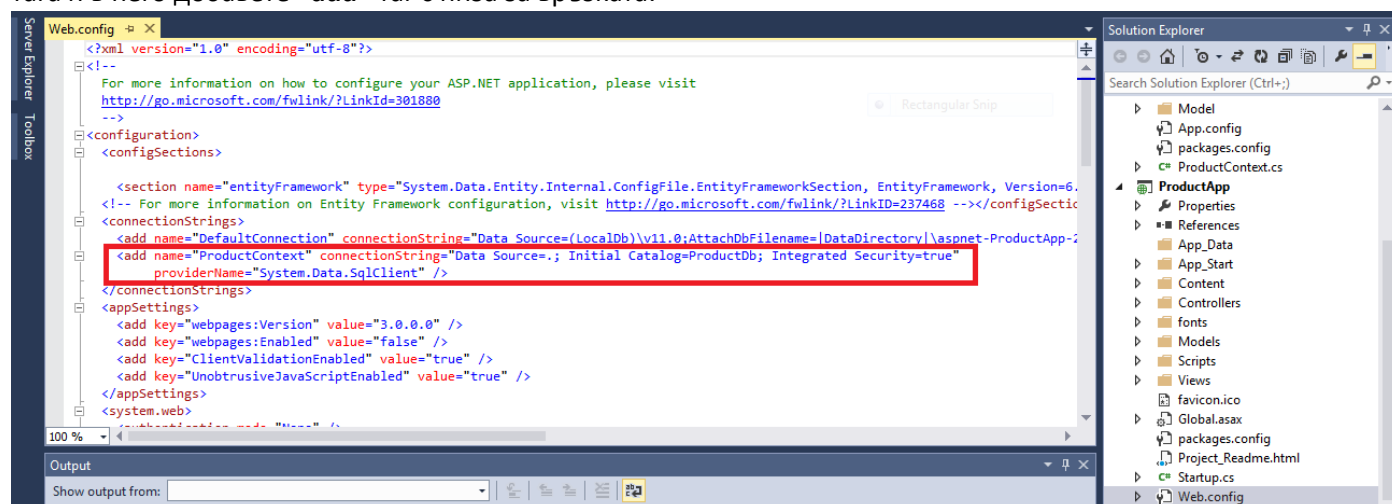


Освен това добавете следните референции (десен бутон върху проекта -> Add -> Reference) за всеки от проектите:

- На проект **Business** добавете референцията към **Data**
- На проект **ProductApp** /ASP.NET MVC проекта/ добавете референцията към **Business** и **Data**

3. База данни

Тук ще използваме вече съществуващата от предните проекти **ProductDb**. Отново единственият ни ангажимент е да добавим низа за връзка. В **Web.config** на проект **ProductApp** отворите **<connectionStrings>** тага и в него добавете **<add>** таг с низа за връзката:



4. Слой за данни

Слоят за данни тук е напълно аналогичен с този в предното упражнение.

Тук той се състои от папка с **модел** и **контекст**.

Моделът тук е клас **Product.cs**, находящ се в **Model** папката, като в него описваме единствено свойствата му, които всъщност съответстват на колоните от таблицата.

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Stock { get; set; }
}
```

В самата папка **Data** се намира и **ProductContext** класа. Той трябва да наследява **DbContext**. Тук ще се наложи да добавите и **using** директива, понеже класа **DbContext** е част от **EntityFramework**. Самият клас ще съдържа конструктор, в който ще има обръщение към конструктора на базовия клас, където за параметър се подава името (**name** атрибутът) от низа за връзка, който добавихме в **App.config** по-рано.

Освен това в конструктора ще имаме и свойство, което ще е от **DbSet<Product>**.

Кодът е както следва:

```
public class ProductContext : DbContext
{
    public ProductContext()
        : base("name=ProductContext")
    {
    }
    public DbSet<Product> Products { get; set; }
}
```

С това сме готови с нашия слой за данни.

5. Бизнес слой

Слоят за бизнес логиката тук е напълно аналогичен с този в предното упражнение.

Тук ще имаме поле от тип **ProductContext**, което ще използваме в методите.

Методите като логика работят по абсолютно сходен начин с предното упражнение. Класът изглежда по следния начин:

```
class ProductBusiness
{
    private ProductContext productContext;

    public List<Product> GetAll()...
    public Product Get(int id)...
    public void Add(Product product)...
    public void Update(Product product)...
    public void Delete(int id)...
```

А методите са съответно:

```
public List<Product> GetAll()
{
    using (productContext = new ProductContext())
    {
        return productContext.Products.ToList();
    }
}

public Product Get(int id)
{
    using (productContext = new ProductContext())
    {
        return productContext.Products.Find(id);
    }
}

public void Add(Product product)
{
    using (productContext = new ProductContext())
    {
        productContext.Products.Add(product);
        productContext.SaveChanges();
    }
}

public void Update(Product product)
{
    using (productContext = new ProductContext())
    {
        var item = productContext.Products.Find(product.Id);
        if (item != null)
        {
            productContext.Entry(item).CurrentValues.SetValues(product);
            productContext.SaveChanges();
        }
    }
}

public void Delete(int id)
{
    using (productContext = new ProductContext())
    {
        var product = productContext.Products.Find(id);
        if (product != null)
        {
            productContext.Products.Remove(product);
            productContext.SaveChanges();
        }
    }
}
```

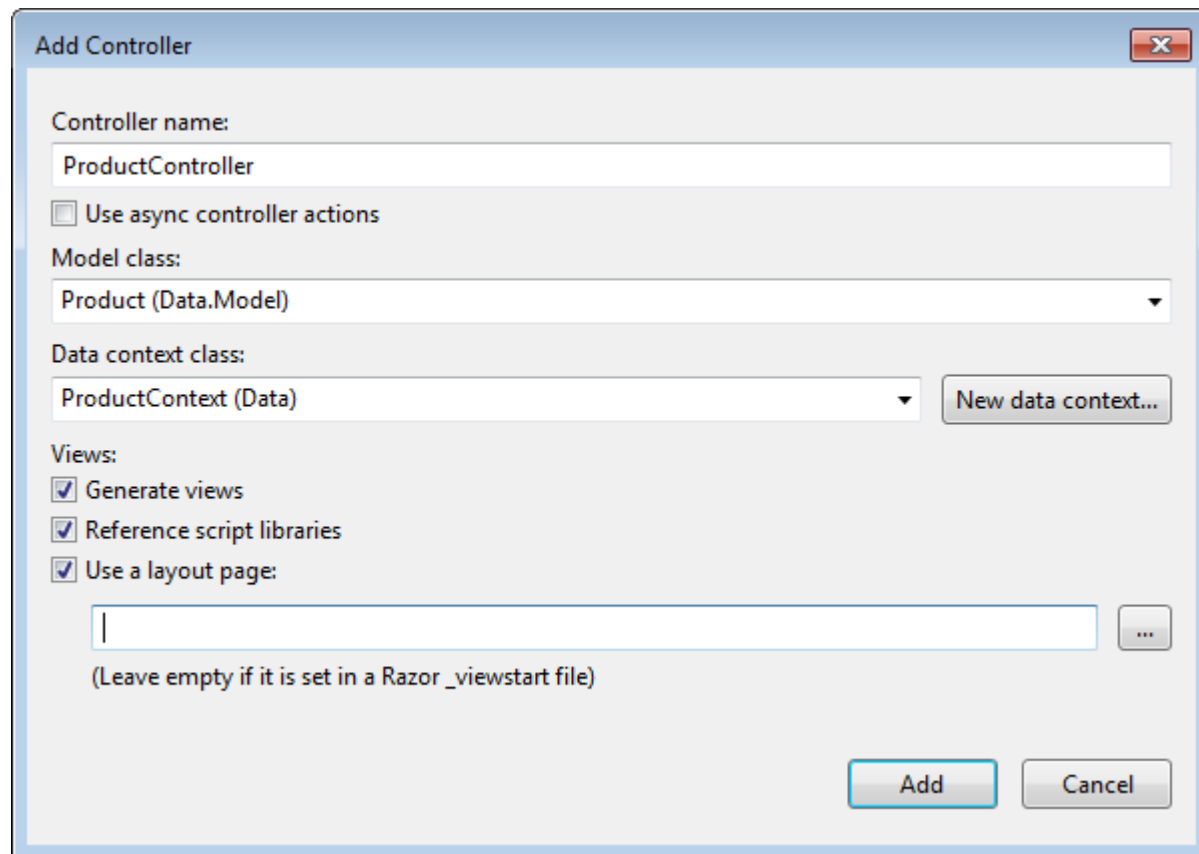
6. Презентационен слой

Уверете се, че сте създали успешно слоя за данни и бизнес слоя. Изпълнете команда **Build** за целия **solution**.

Ако всичко е наред, можем да пристъпим към реализация на нашия презентационен слой. За тази цел ще ползваме уеб интерфейс. Тук няма да се впускаме в подробности как да изменяме външния му вид или да направите допълнително модификации по него, а само ще покажем как се създава.

В **ProductApp** открийте папката **Controllers**. Чрез десен бутон изберете **Add -> Controller**. От опциите изберете **MVC Controller with views, using Entity Framework**.

След това трябва да зададете име на вашия контролер, а освен това трябва да зададете и клас на модела, както и клас на контекста:



Натиснете **Add** и оставете на Visual Studio само да свърши останалото за вас 😊

7. Употреба на приложението

След всичко това е време да видим нашето приложение в действие – за разлика от досегашните ни проекти тук на мястото за стартиране е изписано името на уеб браузър:



Натиснете бутона и стартирайте смело. Ще се отвори браузъра, в който ще фигурира адрес от вида <http://localhost:2960/>. Имайте предвид, че вместо 2960 е възможно при вас номерът на порта да е друг!

Към този адрес добавете думичката **Product**. Ще ви излезе страничка, в която са показани всички записи от базата данни с възможност за **редакция /Edit/**, **подробности /Details/** и **изтриване /Delete/**. Това са CRUD операциите и те сега са налични чрез добре изглеждащ уеб интерфейс.

С това приключваме реализацията на презентационния ни слой и съответно това упражнение.

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "Обучение за ИТ кариера" на МОН за подготовка по професия "Приложен програмист".



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от фондация "Софтуерен университет" и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).



SoftUni
Foundation

