**EntityFrameworkSample\_Completed**

---------------------------------------------------------

Тъй като при първо стартиране нямаме картинки за ресторанти трява да си свалим няколко.

----------------------------------------------------------------------------------------------------------------------------

Ако нямаме база данни можем да си направим такава с проекта Restaurants.DB

Отваряме го и избираме Publish.

От там си избираме нашия сървър : Target database connection и пишем име на новата база данни. Напрмер RestaurantsEFC и даваме Publish.

---------------------------------------------------------------------------------------------------

Ако има такава база:

Отваряме проекта DataAccess и отваряме файла App.Config. Там проверяваме дали в <connectionStrings> тага атрибута connectionString има правилно зададени име на сървъра и име на базата

data source=localhost\SQLEXPRESS

initial catalog=RestaurantsEFC

Можем да тестваме това като като дадем update на .edmx

-------------------------------------------------------------------------------------------------------

За да работи проекта EntityFrameworkSample, трябва да направим същата проверка в неговия файл Web.config.

-------------------------------------------------------------

Първо разглеждаме проекта DataAccess

Там ни интересува модела RestaurantsModel.edmx

Също и контекста който се намира във файла RestaurantsModel.Context.cs

Там се намира класа RestaurantsEntities : DbContext. Този клас ще го ползваме когато инстанцираме обект чрез който ще ползваме базата данни. Това се прави в проекта Repositories.

В този проект се намират и автоматично генерираните класове за ентититата:

Category.cs, City.cs, Restaurants.cs – това са базовите класове.

-------------------------------------------------------------------------

Разглеждаме проекта Repositories. Той съдържа класове за работа с обектите (таблиците).

Тъй като и трите вида обекти (Category, City, Restaurants) ще имат методи с еднакви имена и подобна функционалност може да дефинираме един интерфейс IbaseRepository. За да бъде универсален този интерфейс е generic.

Например да разгледаме един метод от интерфейса:

List<T> GetAll();

Когато имплементираме този интерфейс за класа City ще имаме готов метод:

List< City > GetAll();

Имаме и един базов клас BaseRepository. Той също за удобстово е generic. В този клас са дефинирани всички методи които имат еднотипна функционалност за трите вида обекти (Category, City, Restaurants). И тъй като е generic само задаваме с кой тип да се работи.

В баовия клас е дефиниран и контекста с който ще работят всички останали репозиторита. Както споменахме по-горе той е от DataAccess проекта и е от тип RestaurantsEntities.

Тъй като метода Save() ще има различни дефиниции в наследниците, той трябва да се дефинира като abstract. Заради това базовия клас става абстрктен.

============================================================================

Разглеждаме уеб проекта EntityFrameworkSample.

//

Можем да стартираме проекта и да отидем на Restaurants и с бутона Edit да редактираме някои от тях като им наслагаме картинки.

//

--------------------------

Нека разгледаме някои от функционалностите.

Обикновенно за да се ориентираме отваряме \Views\Shared\\_Layout.cshtml за да разгледаме какви менюта имаме. Например може да разгледаме линка:

@Html.ActionLink("Categories", "Index", "Category")

Ако изберем него ще ни отведе до action-а Index в конторлера Category.

Затова отваряме CategoryController и разглеждаме action-а Index.

Тъй като в index-а на категориите искаме да изведем всички, то ще си създадем списък от категории.

Използваме обект CategoryViewModel който ще съдържа информация за една категория. Ние попълваме списък със всички категории и този списък ще го подадем към строго типизирано View (т.е. има за модел този списък от обекти CategoryViewModel) за да бъде визуализиран там.

!! Много е важно да обърнем внимание, че към View-то се подават обекти от типа “CategoryViewModel”, а не от базовия тип Category въпреки че ще върши подобна работа. Това се прави за разграничаване на View модела!!!

Във View-то \Views\Category\Index.cshtml виждаме че с един цикъл foreach се извеждат всички категории като за всяка има по два линка които са всъщност бутони (bootstrap: имат @class = "btn.....“)

Нека сме си избрали бутона:

@Html.ActionLink("Edit", "Edit", "Category", new { id= category.ID }, new { @class = "btn btn-primary btn-sm" })

Той ще ни отведе към контролера Category в action-а Edit и ще има подаден параметър id. Http метода е GET. Затова разглеждаме в контролера метода:

public ActionResult Edit(int id = 0)

там се търси категория с това ID чрез метода от репозиторито categoryRepository.GetByID(id)

След като намерим такава категория използваме обект от CategoryViewModel и го подаваме на View-то. View-то носи името като Action-а, т.е. Edit (това става автоматично при създаване на View).

Затова разглеждаме \Views\Category\Edit.cshtml

Това View e отново строго типизирано и работи с подадения му обект (т.е. model) CategoryViewModel.

Тъй като ще може да се редактира дадена категория ние трябва да знаем нейното ID. Него ще го пазим в едно скрито поле : @Html.HiddenFor(model => model.ID)

В това View има форма Html.BeginForm("Edit", "Category") която съдържа

както една текстова кутия за редактиране на името на категорията

@Html.TextBoxFor(model => model.Name)

така и бутон Submit за самата форма. Когато натиснем Submit бутона, модела model (който е с променено Name, но същото ID ) ще се изпрати към контролера Category и намиращия се там метод Edit. Тъй като формата работи с http метод POST разглеждаме:

[HttpPost]

public ActionResult Edit(CategoryViewModel categoryEdit)

Там намираме категорията с това ID, променяме и името и я запазваме. След това препращаме към View-то "Index". Когато искаме да предадем едно съобщение от едно view към друго може да използваме глобалното пропърти TempData което ще е видимо и от друго View. В случая искаме да изпишем че сме добавили нова категория.

-----------------------------

Интересно е да се разгледа как е реализиран DropDownList във View-то \Views\Restaurants\Edit.cshtml

От контролера RestaurantController (метода [HttpGet] ... Edit(...)) може да видим че списъка с градовете и категориите се подават през ViewBag