# Упражнения: Да направим приложение с ORM

## Създаване на просто клиентско приложение

След като рамката е готова, нека да видим как намира типовете, таблиците връзките и всичко останало в нашата БД, използвайки силата на отражението.

## Създаване на базата данни

Импортирайте този SQL скрипт в SSMS:

|  |
| --- |
|  |

## Създайте проекта

Създайте ново C# **конзолно приложение**, наречено “MiniORM.App” и **добавете референция** към MiniORM проекта:





## Дефинирайте модел на данните /Data Model/

Създайте директория **Data**, а в нея - директория **Entities**. Резултатът трябва да изглежда така: 

Сега нека да създадем модела. Първо създайте Department клас в Entities папката. Класовете за данни имат по **едно свойство за всяка колона** от таблицата.

Създайте две свойства – Id и Name. За Id, използвайте [Key] анотацията (не забравяйте да добавите и using System.ComponentModel.DataAnnotations), за да покажете на нашият фреймуорк, че това е **първичния ключ** на записа. Можем да забраним на Name **свойството** да има **null стойност** при извикване на SaveChanges(), добавяйки [Required] анотация. Нашата **рамка** се грижи за **валидирането на вскеи обект** преди записването на каквито и да е промени в БД. Накрая, добавете ICollection от служители като **навигационно свойство** за всички **служители**, които са част от даден **отдел**. Когато сте готови, класът трябва да изглежда така:



Сега създайте клас **Project** с Id и Name. Id е **първичен ключ,** а Name трябва да е **задължително** (стойност различна от null). Освен това, създайте и **навигационно свойство,** наречено EmployeesProjects, което е съпоставящо между Employee и Project данните. Ще създадем този клас по-късно.

Като цяло е добра идея да ползваме свойства, които са само с **get**, когато типът е ICollection<T> за навигационните свойства, с цел да не позволяваме да бъдат редеклраиране извън рамката.

Полученият код трябва да изглежда по сходен начин:



След това, създайте **Employee** клас и използвайте същата логика. Единствената разлика между другите два модела е, че в **Employee** класа, трябва да ползваме **външен ключ** към **Department** модела. Това се прави посредством анотация [ForeignKey(nameof(Department)] над DepartmentId свойството.



Последният клас за създаване е EmployeesProjects, където ще имаме **cъставен** ключ за свойствата Projects и EmployeesId. А след това ще направим двата съставящи ключа и **външни**.



Сега е ред на DbContext класа. Създайте ItKarieraDbContext в Data папката, който **наследява** от базовия DbContext и има DbSet-ове за всички **модели**, които сме създали. Уверете се, че наследявате и конструктора.



## Тестване на рамката

Време е да тестваме MiniORM рамката, вкарвайки малко данни в БД. В главния метод трябва да декларирате **низа за връзка.** След това създайте инстанция на ItKarieraDbContext със съответния низ за връзка и вмъкнете нов Employee обект. След това, намерете **първият служител** и променете неговото **име**. Накрая, извикайте SaveChanges() метода. 

Ако всичко сработи без изключения, то би трябвало всичко да е готово!

## 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".



* 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).

