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.

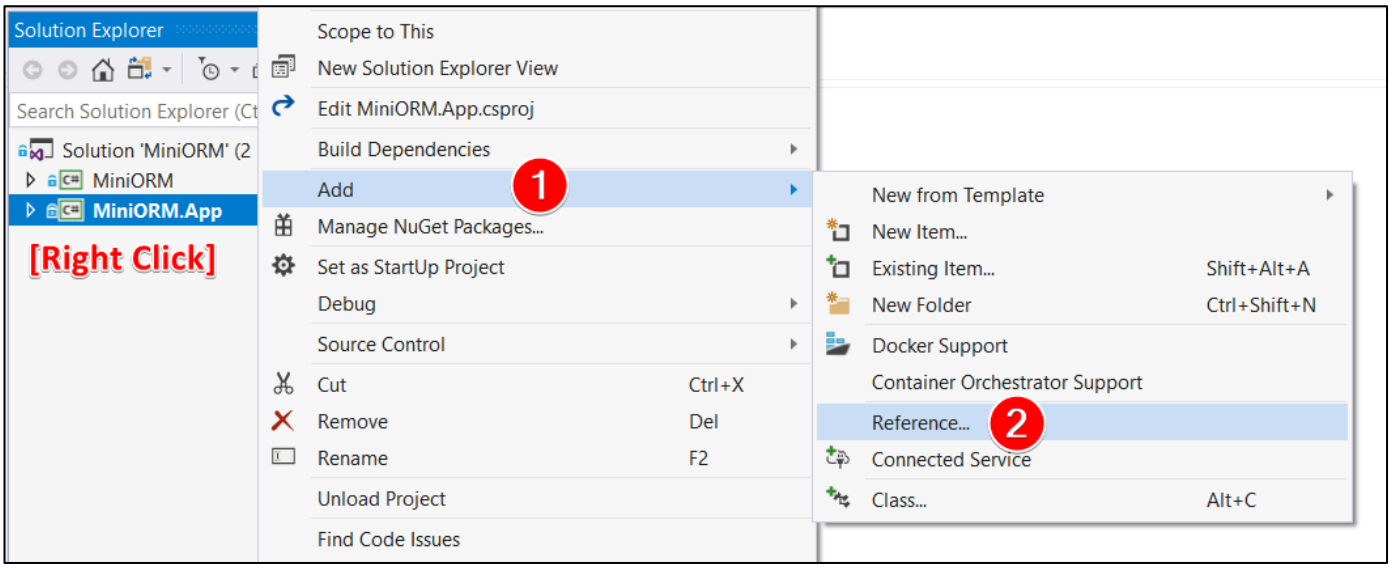
Problem 1. Създаване на базата данни

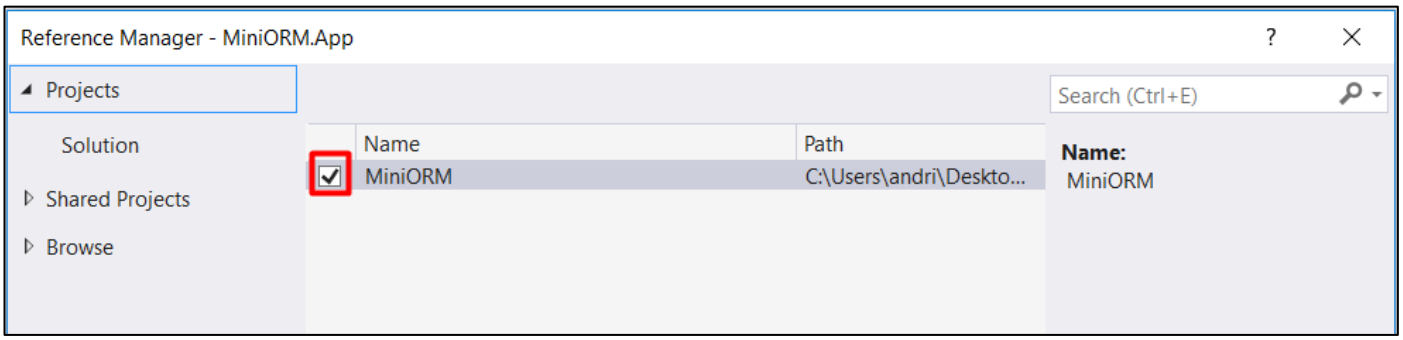
Импортирайте този SQL скрипт в SSMS:

```
CREATE DATABASE MiniORM
GO
USE MiniORM
GO
CREATE TABLE Projects
(
        Id INT IDENTITY PRIMARY KEY,
        Name VARCHAR(50) NOT NULL
)

CREATE TABLE Departments
(
        Id INT IDENTITY PRIMARY KEY,
        Name VARCHAR(50) NOT NULL
)

CREATE TABLE Employees
(
        Id INT IDENTITY PRIMARY KEY,
        FirstName VARCHAR(50) NOT NULL,
        MiddleName VARCHAR(50),
        LastName VARCHAR(50) NOT NULL,
        IsEmployed BIT NOT NULL,
        DepartmentId INT
        CONSTRAINT FK_Employees_Departments FOREIGN KEY
        REFERENCES Departments(Id)
)
CREATE TABLE EmployeesProjects
(
        ProjectId INT NOT NULL
        CONSTRAINT FK_Employees_Projects REFERENCES Projects(Id),
        EmployeeId INT NOT NULL
        CONSTRAINT FK_Employees_Employee REFERENCES Employees(Id),
        CONSTRAINT PK_Projects_Employees
        PRIMARY KEY (ProjectId, EmployeeId)
)
GO
INSERT INTO MiniORM.dbo.Departments (Name) VALUES ('Research');
INSERT INTO MiniORM.dbo.Employees (FirstName, MiddleName, LastName, IsEmployed, DepartmentId) VALUES
('Stamat', NULL, 'Ivanov', 1, 1),
('Petar', 'Ivanov', 'Petrov', 0, 1),
('Ivan', 'Petrov', 'Georgiev', 1, 1),
('Gosho', NULL, 'Ivanov', 1, 1);
INSERT INTO MiniORM.dbo.Projects (Name)
VALUES ('C# Project'), ('Java Project');
INSERT INTO MiniORM.dbo.EmployeesProjects (ProjectId, EmployeeId) VALUES
(1, 1),
(1, 3),
(2, 2),
(2, 3)
```

Problem 2. Създайте проекта

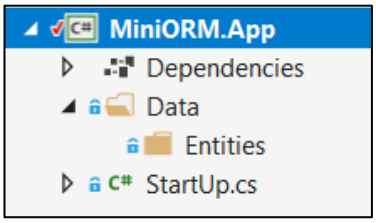
Създайте ново C# **конзолно приложение**, наречено "**`MiniORM.App`**" и **добавете референция** към **`MiniORM`** проекта:





Problem 3. 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 всеки 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:

```
namespace MiniORM.App.Data.Entities
{
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public class Department
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public string Name { get; set; }

        public ICollection<Employee> Employees { get; }
    }
}
```

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:

```
namespace MiniORM.App.Data.Entities
{
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public class Project
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public string Name { get; set; }

        public ICollection<EmployeeProject> EmployeeProjects { get; }
    }
}
```

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.

```
namespace MiniORM.App.Data.Entities
{
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;

    public class Employee
    {
        [Key]
        public int Id { get; set; }

        [Required]
        public string FirstName { get; set; }

        public string MiddleName { get; set; }

        [Required]
        public string LastName { get; set; }

        public bool IsEmployed { get; set; }

        [ForeignKey(nameof(Department))]
        public int DepartmentId { get; set; }

        public Department Department { get; set; }

        public ICollection<EmployeeProject> EmployeeProjects { get; }
    }
}
```

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

```
namespace MiniORM.App.Data.Entities
{
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Schema;

    public class EmployeeProject
    {
        [Key]
        [ForeignKey(nameof(Employee))]
        public int EmployeeId { get; set; }

        [Key]
        [ForeignKey(nameof(Project))]
        public int ProjectId { get; set; }

        public Employee Employee { get; set; }

        public Project Project { get; set; }
    }
}
```

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.

```
namespace MiniORM.App.Data
{
    using Entities;

    public class SoftUniDbContext : DbContext
    {
        public SoftUniDbContext(string connectionString)
            : base(connectionString)
        {
        }

        public DbSet<Employee> Employees { get; }

        public DbSet<Department> Departments { get; }

        public DbSet<Project> Projects { get; }

        public DbSet<EmployeeProject> EmployeesProjects { get; }
    }
}
```

Problem 4. 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.

```
namespace MiniORM.App
{
    using System.Linq;
    using Data;
    using Data.Entities;

    public class StartUp
    {
        public static void Main(string[] args)
        {
            var connectionString = "Server=.;Database=MiniORM;Integrated Security=True";

            var context = new SoftUniDbContext(connectionString);

            context.Employees.Add(new Employee
            {
                FirstName = "Gosho",
                LastName = "Inserted",
                DepartmentId = context.Departments.First().Id,
                IsEmployed = true,
            });

            var employee = context.Employees.Last();
            employee.FirstName = "Modified";

            context.SaveChanges();
        }
    }
}
```

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).