

# Да напишем ORM!



Учителски екип

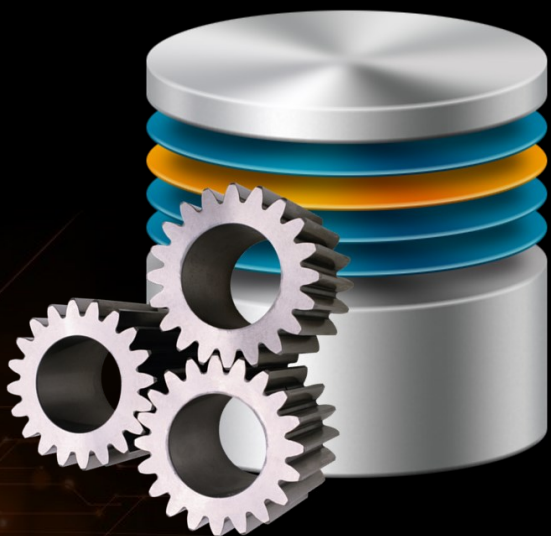
Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



# Съдържание

1. ORM анатомия
2. Класове от данни



MiniORM Cre

**Да си направим ORM рамка**  
Какво и как?



# MiniORM Core: Преглед

- Проектиран по подобие на **Entity Framework Core**
- Дава ни възможност да използваме LINQ-базирани заявки и **CRUD** операции
- **Change tracking** of in-memory objects
- Съпоставя навигационни свойства
- Съпоставя колекции
  - Един-към-много, Много-към-един, ...

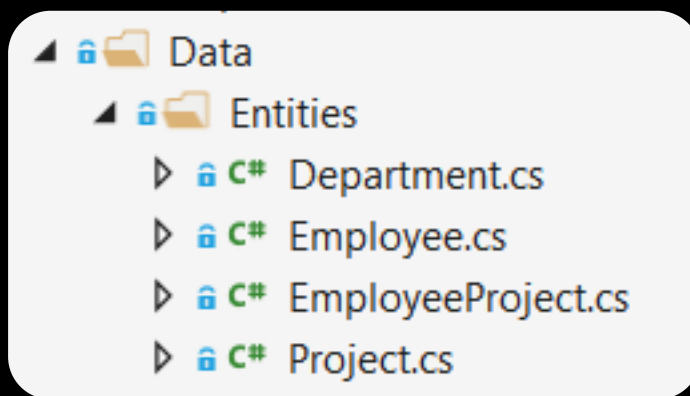
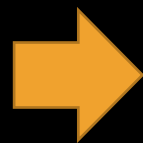
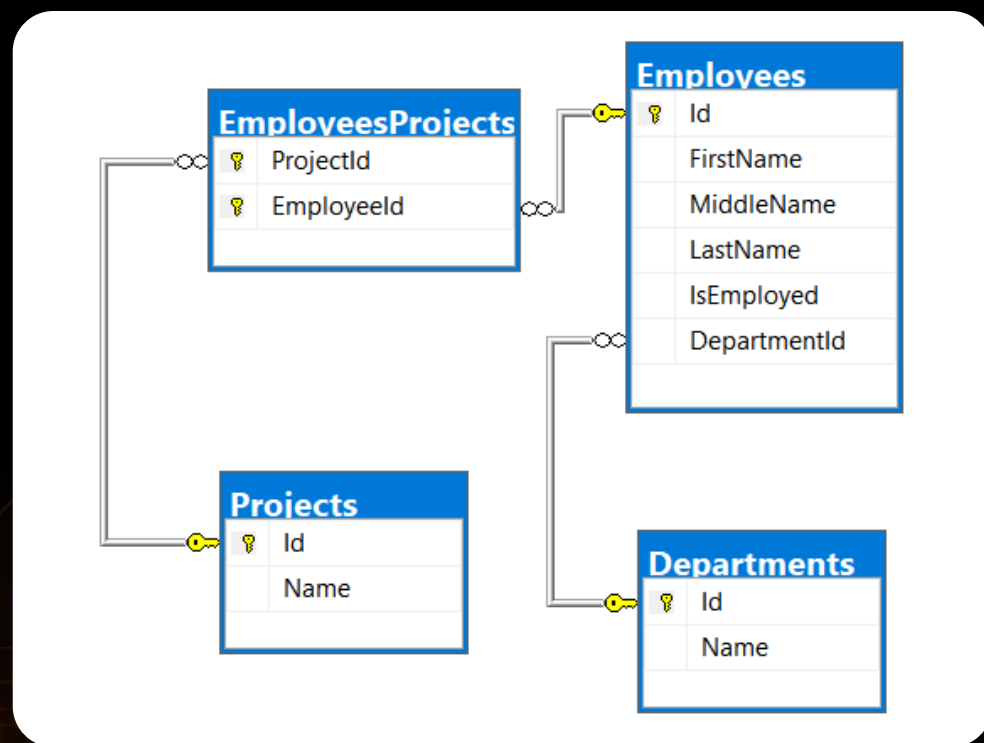
MiniORM Core

# MiniORM Core работен поток

- Дефинираме модел на информацията (БД с предимство; database-first)
  - Класове на данните /Entity Classes/
  - **DbContext** (с **DbSets**)
- Инициализиране на **DbContext**
  - Използвайки низ за връзка /connection string/
- Изпълняване на заявки върху информация, чрез контекст
- Манипулиране на данни (добавяне/премахване/промяна на данни)
- Контекстът се запазва в базата данни

# Модел „База данни с предимство“

- „База данни с предимство“ моделът моделира класовете на данните /entity classes/ според базата данни



# MiniORM компоненти

## ■ DbContext класът

- Отговаря за връзката с БД и DB Sets
- Предоставя ни LINQ-базиран достъп до данните
- Предоставя ни change tracking, и API за CRUD операции

## ■ DB Sets

- Съдържат данни /entities/ (обекти с техните атрибути и връзки)
- Всяка таблица в БД обикновено съответства на единствен C# клас

# MiniORM компоненти (2)

- **Асоциации** (съответствия на връзките)
  - Асоциацията е връзка базирана на първичен или външен ключ между два класа на данни
  - Позволява навигиране от един клас данни към друг

```
var courses = student.Courses.Where(...);
```

- MiniORM поддържа **един-към-един**, **един-към-много** и **много-към-много** връзки




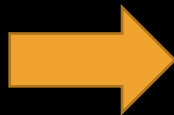
```
public class Employee
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string MiddleName { get; set; }
    public string LastName { get; set; }
    public bool IsEmployed { get; set; }
    public Department Department { get; set; }
}
```

## Класове от данни (Entity Classes)

# Класове от данни

- Класовете от данни са обикновени C# класове
- Използват се, за да съхраняват данните от БД в паметта

Employees	
	Id
	FirstName
	MiddleName
	LastName
	IsEmployed
	DepartmentId



```
public class Employee
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string MiddleName { get; set; }
    public string LastName { get; set; }
    public bool IsEmployed { get; set; }
    public Department Department { get; set; }
}
```

# Класове от данни: Навигационни свойства

- Свойства от референтен тип
- Сочат към съответни обекти, свързани чрез външен ключ
- Задават се от рамката
- Пример: Отдел служители

```
public class Employee
{
    public int Id { get; set; }
    ...
    [ForeignKey(nameof(Department))]
    public int DepartmentId { get; set; }
    public Department Department { get; set; }
}
```

## Класове от данни: Навигационни свойства (2)

- Навигационните свойства могат и да са колекции
- Обикновено се използва **ICollection<T>**
- Съдържат всички обекти, чиито **външни ключове** са същите като **основния ключ** на класа от данни
- Задава се от ORM рамката

```
public class Department
{
    public int Id { get; set; }
    ...
    public ICollection<Employee> Employees { get; set; }
}
```



Да напишем ORM



Въпроси?



# 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**



SoftUni  
Foundation

