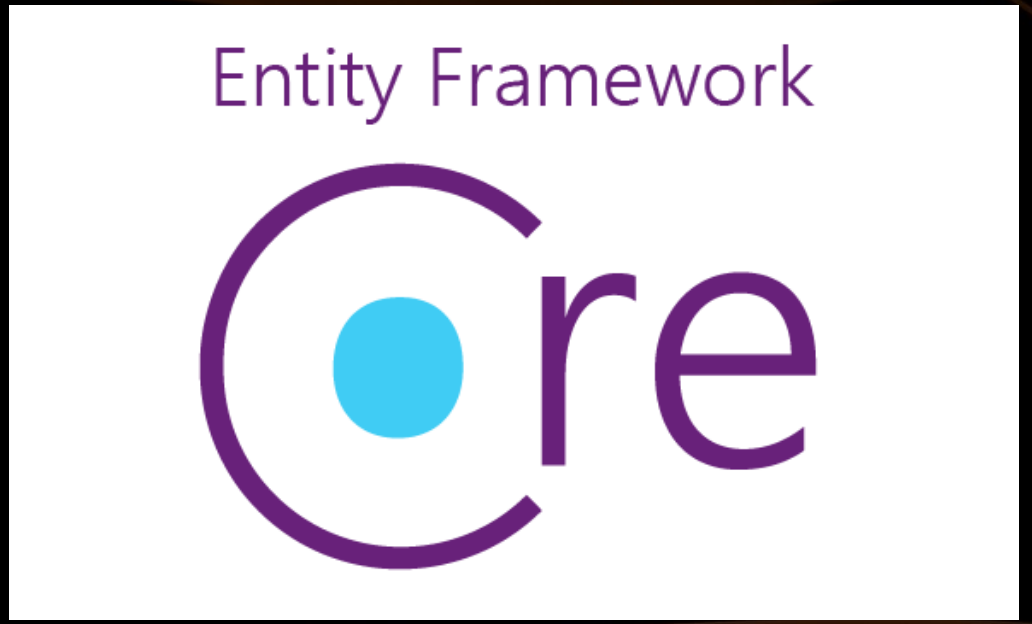
Комуникация с база от данни

Entity Framework Core

Учителски екип

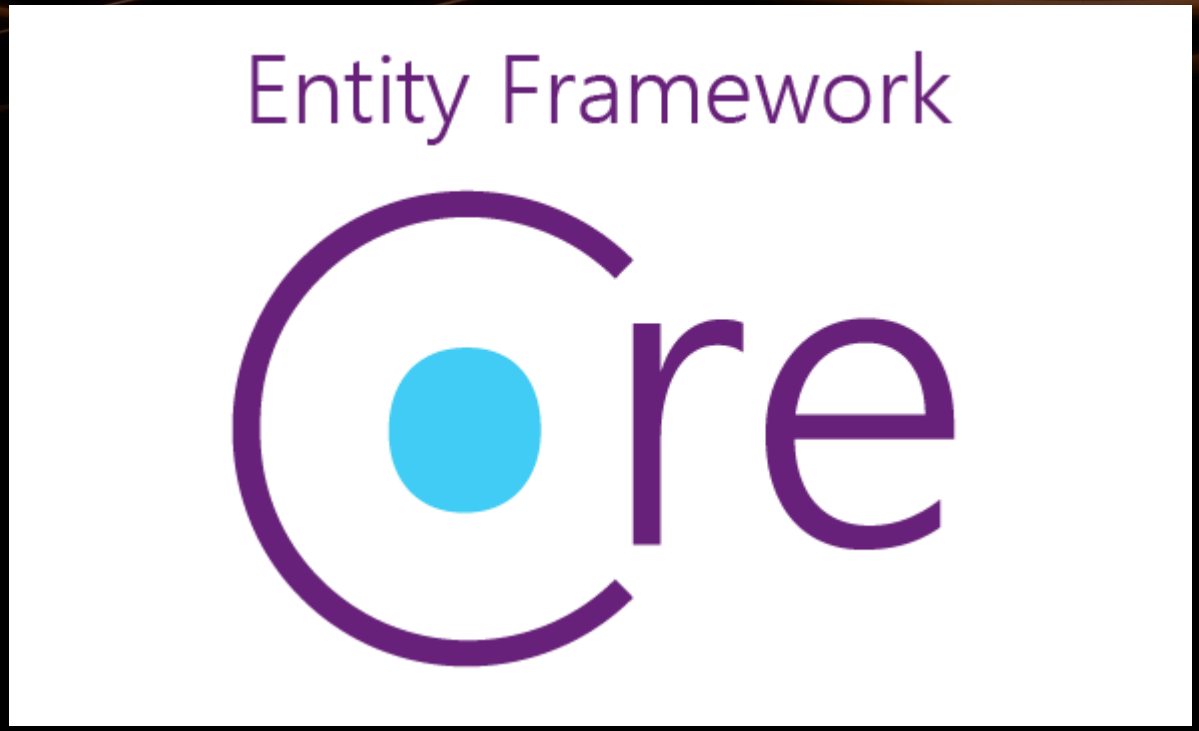
Обучение за ИТ кариера

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



Съдържание

1. Преглед на Entity Framework Core
2. Моделът "Database First"
3. CRUD Операции използвайки Entity Framework Core
4. Работа с LINQ



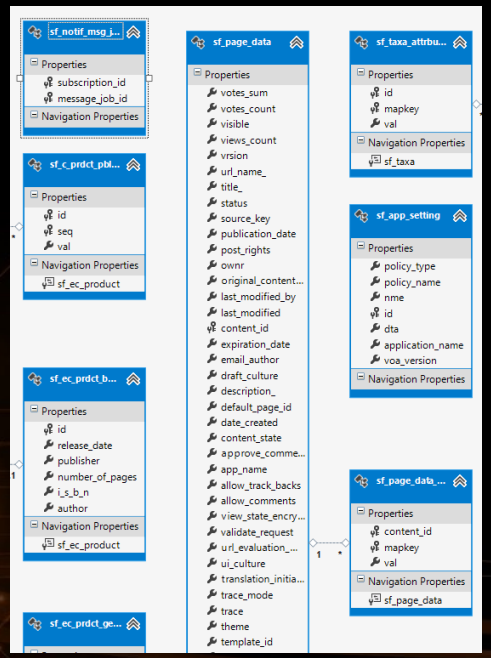
Entity Framework Core

Entity Framework Core

- Стандартната рамка за ORM за .NET и .NET Core
- Предоставя LINQ-базирани заявки за данни и CRUD операции
- Автоматично проследяване на промяната на обекти в паметта
- Работи с много релационни бази данни (с различни доставчици)
- Отворен код с независими цикъл на пускане на версии

Основен Работен Процес

1. Определете модела на данни (Code First / Scaffold from DB)
2. Писане и изпълняване на заявки върх IQueryable
3. EF генерира и изпълнява SQL заявка в БД

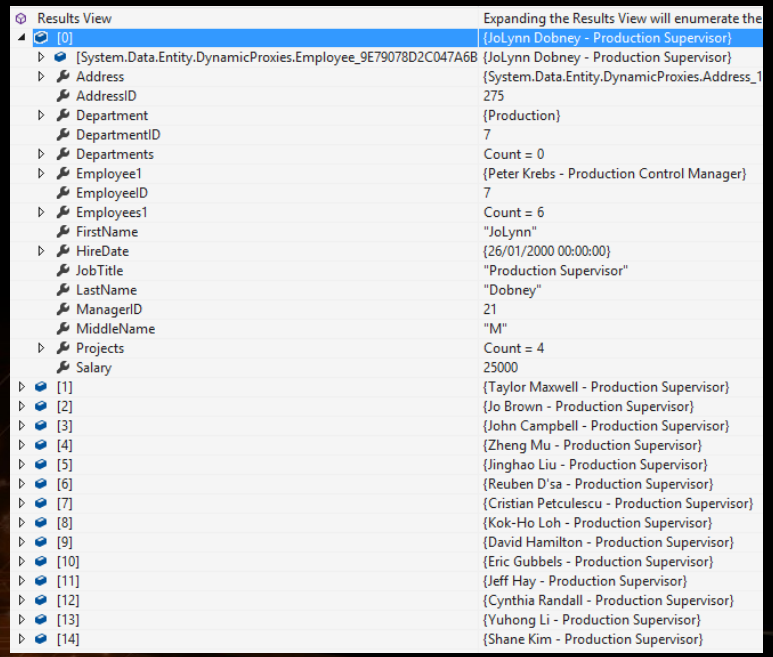



```
var toolName = "";

var snippetOptions = DefaultToolGroup
    .Tools
    .OfType<EditorListTool>()
    .Where(t =>
        t.Name == toolName &&
        t.Items != null &&
        t.Items.Any())
    .SelectMany(
        (t, index) =>
            t.Items
            .Select(item =>
                new {
                    text = item.Text,
                    value = item.Value
                }));

if (snippetOptions.Any())
{
    options[toolName] = snippetOptions;
}
```

```
exec sp_executesql N'SELECT
[Filter2].[UserInCourseId] AS [UserInCourse
[Filter2].[UserId] AS [UserId],
[Filter2].[CourseInstanceId1] AS [CourseIns
[Filter2].[FirstCourseGroupId] AS [FirstCou
[Filter2].[SecondCourseGroupId] AS [SecondC
[Filter2].[ThirdCourseGroupId] AS [ThirdCou
[Filter2].[FourthCourseGroupId] AS [FourthC
[Filter2].[FifthCourseGroupId] AS [FifthCou
[Filter2].[IsLiveParticipant] AS [IsLivePar
[Filter2].[Accommodation] AS [Accommodatior
[Filter2].[ExcellentResults] AS [ExcellentR
[Filter2].[Result] AS [Result],
[Filter2].[CanDoTestExam] AS [CanDoTestExam
[Filter2].[CourseTestExamId] AS [CourseTest
[Filter2].[TestExamPoints] AS [TestExamPoir
[Filter2].[CanDoPracticalExam] AS [CanDoPra
[Filter2].[CoursePracticalExamId1] AS [Cour
[Filter2].[PracticalExamPoints] AS [Practic
[Filter2].[AttendancesCount] AS [Attendance
[Filter2].[HomeworkEvaluationPoints] AS [Ho
FROM   (SELECT [Extent1].[UserInCourseId] A
AS [SecondCourseGroupId], [Extent1].[ThirdC
[IsLiveParticipant], [Extent1].[Accommodati
[CourseTestExamId], [Extent1].[TestExamPoir
[PracticalExamPoints], [Extent1].[Attendanc
        FROM  [courses].[UsersInCourses] AS
        INNER JOIN [courses].[CoursePractic
        WHERE ( EXISTS (SELECT
                1 AS [C1]
                FROM [courses].[CoursePract
                WHERE [Extent1].[UserInCour
        )) AND ([Extent2].[AllowExamFilesEv
INNER JOIN [courses].[CoursePracticalExams]
WHERE ([Filter2].[UserId] = @p__linq__0) AN
```

Основен Работен Процес (2)

4. EF преобразува резултатите от заявката в .NET обекти
5. Промяна на данните със C# и се извиква "
   "
6. EF генерира и изпълнява SQL команда за промяна на БД



Results View	Expanding the Results View will enumerate the
[0]	{JoLynn Dobney - Production Supervisor}
[System.Data.Entity.DynamicProxies.Employee_9E79078D2C047A6B	{JoLynn Dobney - Production Supervisor}
Address	{System.Data.Entity.DynamicProxies.Address_1
AddressID	275
Department	{Production}
DepartmentID	7
Departments	Count = 0
Employee1	{Peter Krebs - Production Control Manager}
EmployeeID	7
Employees1	Count = 6
FirstName	"JoLynn"
HireDate	{26/01/2000 00:00:00}
JobTitle	"Production Supervisor"
LastName	"Dobney"
ManagerID	21
MiddleName	"M"
Projects	Count = 4
Salary	25000
[1]	{Taylor Maxwell - Production Supervisor}
[2]	{Jo Brown - Production Supervisor}
[3]	{John Campbell - Production Supervisor}
[4]	{Zheng Mu - Production Supervisor}
[5]	{Jinghao Liu - Production Supervisor}
[6]	{Reuben D'sa - Production Supervisor}
[7]	{Cristian Petculescu - Production Supervisor}
[8]	{Kok-Ho Loh - Production Supervisor}
[9]	{David Hamilton - Production Supervisor}
[10]	{Eric Gubbels - Production Supervisor}
[11]	{Jeff Hay - Production Supervisor}
[12]	{Cynthia Randall - Production Supervisor}
[13]	{Yuhong Li - Production Supervisor}
[14]	{Shane Kim - Production Supervisor}

```
private void ChangeBlogPostName(int id,
    string newName)
{
    var db = new Context();

    var post = db.Posts
        .FirstOrDefault(x => x.Id == id);

    if (post == null)
    {
        throw new ArgumentException(
            "Item with that id was not fo
            "id");
    }

    post.Name = newName;

    db.SaveChanges();
}
```

```
SELECT
[Extent1].[EmployeeID] AS [EmployeeID],
[Extent1].[FirstName] AS [FirstName],
[Extent1].[LastName] AS [LastName],
[Extent1].[MiddleName] AS [MiddleName],
[Extent1].[JobTitle] AS [JobTitle],
[Extent1].[DepartmentID] AS [DepartmentID],
[Extent1].[ManagerID] AS [ManagerID],
[Extent1].[HireDate] AS [HireDate],
[Extent1].[Salary] AS [Salary],
[Extent1].[AddressID] AS [AddressID]
FROM [dbo].[Employees] AS [Extent1]
WHERE N'Production Supervisor' = [Extent1].[JobTitle]
```

Entity Framework Core: Конфигурация

- За да добавите поддръжка на EF Core към проект във Visual Studio:
  - Инсталирайте го от

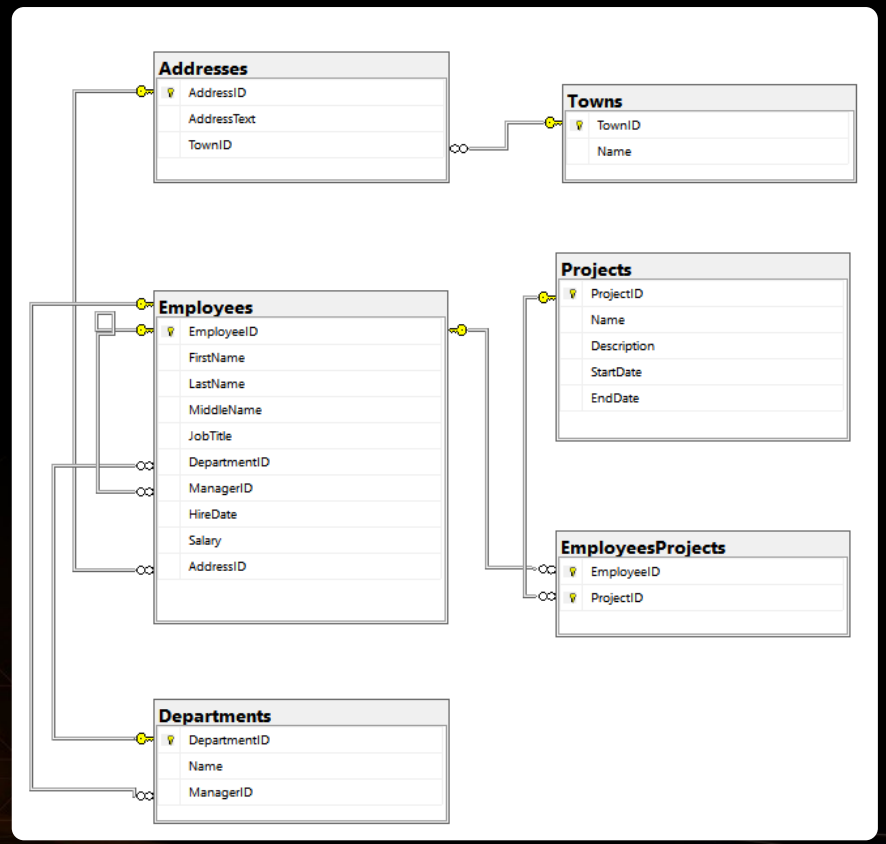
```
Install-Package Microsoft.EntityFrameworkCore
```

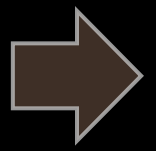
  - EF Core е модулен - различни допълнителни пакети могат да бъдат инсталирани:

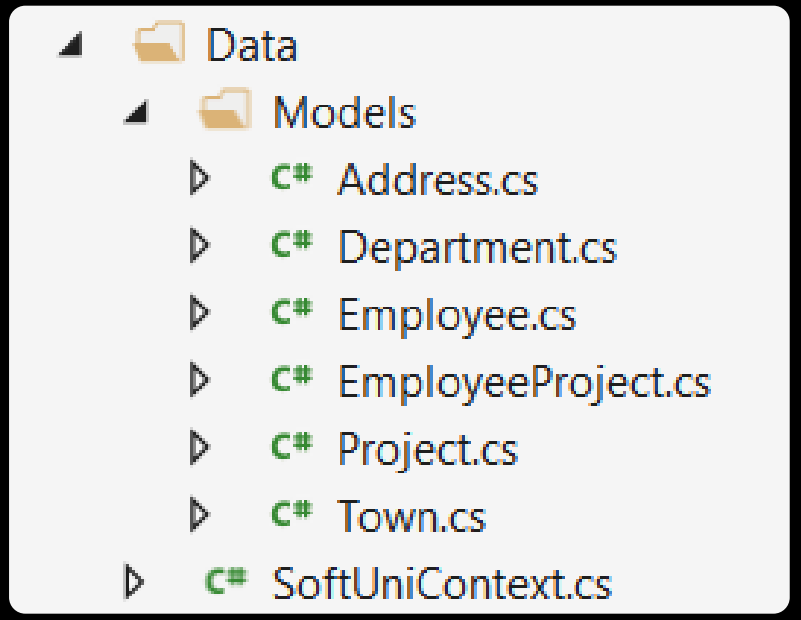
```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

Моделът "Database First"

- Моделът "Database First" моделира класовете субекта след като базата данни е създадена:







Използване на Модела "Database First"

- Scaffolding DbContext от DB с командата Scaffold-DbContext в конзолата за управление на пакети:

```
Scaffold-DbContext
  -Connection "Server=.;Database=…;Integrated Security=True"
  -Provider Microsoft.EntityFrameworkCore.SqlServer
  -OutputDir Data
```

- Scaffolding предварително изисква следните пакети:

```
Install-Package Microsoft.EntityFrameworkCore.Tools
Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design
```

EF Компоненти

- Класът DbContext:
  - Съдържа връзката към базата данни и преобразуваните класове
  - Осигурява достъп до данни, базиран на LINQ
  - Осигурява проследяване на идентичността, проследяване на промените и API за CRUD операции
- Entity classes
  - Всяка таблица от база данни се свежда до C# клас

EF Компоненти (2)

- Асоциации (връзки между таблиците)
  - Асоциацията е базирана на първичен ключ / чужд ключ между два класа
  - Позволява навигация от един обкет към друг

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

- Concurrency контрол
  - Entity Framework използва optimistic concurrency контрол
    - Няма заключване по подразбиране
    - Автоматично открива concurrency конфликти

Четене на Данни

Заявки към БД с помощта на EF Core

Класът DbContext

- DbContext предоставя:
  - CRUD Операции
    - Начин за достъпване на записите
    - Метод за добавяне на нови записи (методът Add())
    - Възможност за манипулиране на данни от база данни чрез промяна на обекти
- Изпълнение на LINQ заявки като SQL заявки
- Управление на база данни създаване/изтриване/миграция

Използване на Класът DbContext

- Първо създайте инстанци на класа DbContext:

```
var context = new SoftUniDbContext();
```

- В конструктора може да бъде подаден низ за свързване към БД
- Свойствата на класа DbContext:
  - Database – EnsureCreated/Deleted методи, DB връзка
  - ChangeTracker – Съдържа информация за вградения тракера за промени
  - Всички таблици са изредени като свойства в следния формат :
    - DbSet<Employee> Employees { get; set; }

Четене на Данни с LINQ Заявки

- Изпълненние на LINQ заявка:

```
using (var context = new SoftUniEntities())
{
  var employees = context.Employees
    .Where(e => e.JobTitle == "Design Engineer")
    .ToArray();
}
```

EF превежда това до SQL заявка

- Employees е свойство към класа DbContext:

```
public partial class SoftUniEntities : DbContext
{
  public DbSet<Employee> Employees { get; set; }
  public DbSet<Project> Projects { get; set; }
  public DbSet<Department> Departments { get; set; }
}
```

Четене на Данни с LINQ Заявки (2)

- Може да се използват и extension методи в заявката

```
using (var context = new SoftUniEntities())
  var employees = context.Employees
      .Where(c => c.JobTitle == "Design Engineering")
      .Select(c => c.FirstName)
      .ToList();
```

- Намиране на запис по ID

```
using (var context = new SoftUniEntities())
{
  var project = context.Projects.Find(2);
  Console.WriteLine(project.Name);
}
```

Прости Операции с LINQ

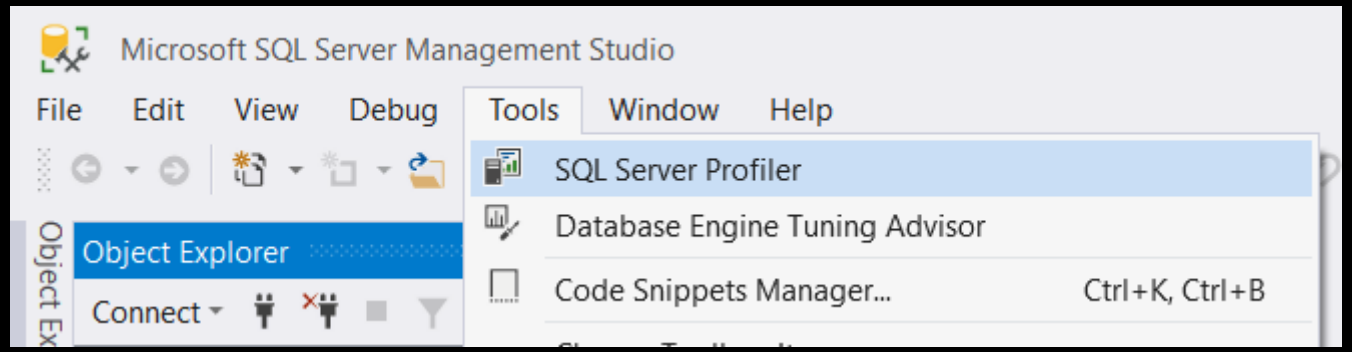
- Where()
  - Търси по дадено условие
- First/Last() / FirstOrDefault/LastOrDefault()
  - Получава първия / последния елемент, който съответства на условието
  - Хвърля InvalidOperationException грешка без OrDefault
- Select()
  - Преобразува колекция до друг тип
- OrderBy() / ThenBy() / OrderByDescending()
  - Сортира колекция по дадено условие

Прости Операции с LINQ (2)

- Any()
  - Проверява дали някой елемент съответства на условие
- All()
  - Проверява дали всички елементи съответстват на условие
- Distinct()
  - Връща само уникалните елементи от колекция
- Skip() / Take()
  - Пропуска / взима X на брой елементи

Проследяване на SQL Заявките

- Заявки, изпратени до SQL Server, могат да бъдат наблюдавани с SQL Server Profiler
  - Включено е в SQL Server Management Studio:



- Заявките също могат да бъдат наблюдавани с Express Profiler

https://expressprofiler.codeplex.com/

CRUD Операции с EF Core

Създаване на Нови Данни

- За да създадете нов ред в БД, използвайте метода Add(...) на съответния DbSet:

```
var project = new Project()
{
  Name = "Judge System",
  StartDate = new DateTime(2015, 4, 15),
};

context.Projects.Add(project);
context.SaveChanges();
```

Създаване на нво Project обект

Добавяне на обекта към DbSet-а

Изпълнява SQL заявка

Каскадни Добавяния

- Можем да добавяме и каскадно:

```
Employee employee = new Employee();
employee.FirstName = "Petya";
employee.LastName = "Grozdarska";
employee.Projects.Add(new Project { Name = "SoftUni Conf"} );
softUniEntities.Employees.Add(employee);
softUniEntities.SaveChanges();
```

- Проектът ще бъде добавен, когато служителя бъде добавен в базата данни

Промяна на Съществуващи Данни

- DbContext позволява промяна на свойствата на обект и запазване на промяната в базата данни
  - Просто заредете записа, променете го и извикайте SaveChanges()
- DbContext автоматично проследява всички промени

```
Employees employee =
  softUniEntities.Employees.First();
employees.FirstName = "Alex";
context.SaveChanges();
```

Изтриване на Съществуващи Данни

- Изтриването се извършва чрез Remove() върху зададена колекция
- Методът SaveChanges() извършва изтриване в базата данни

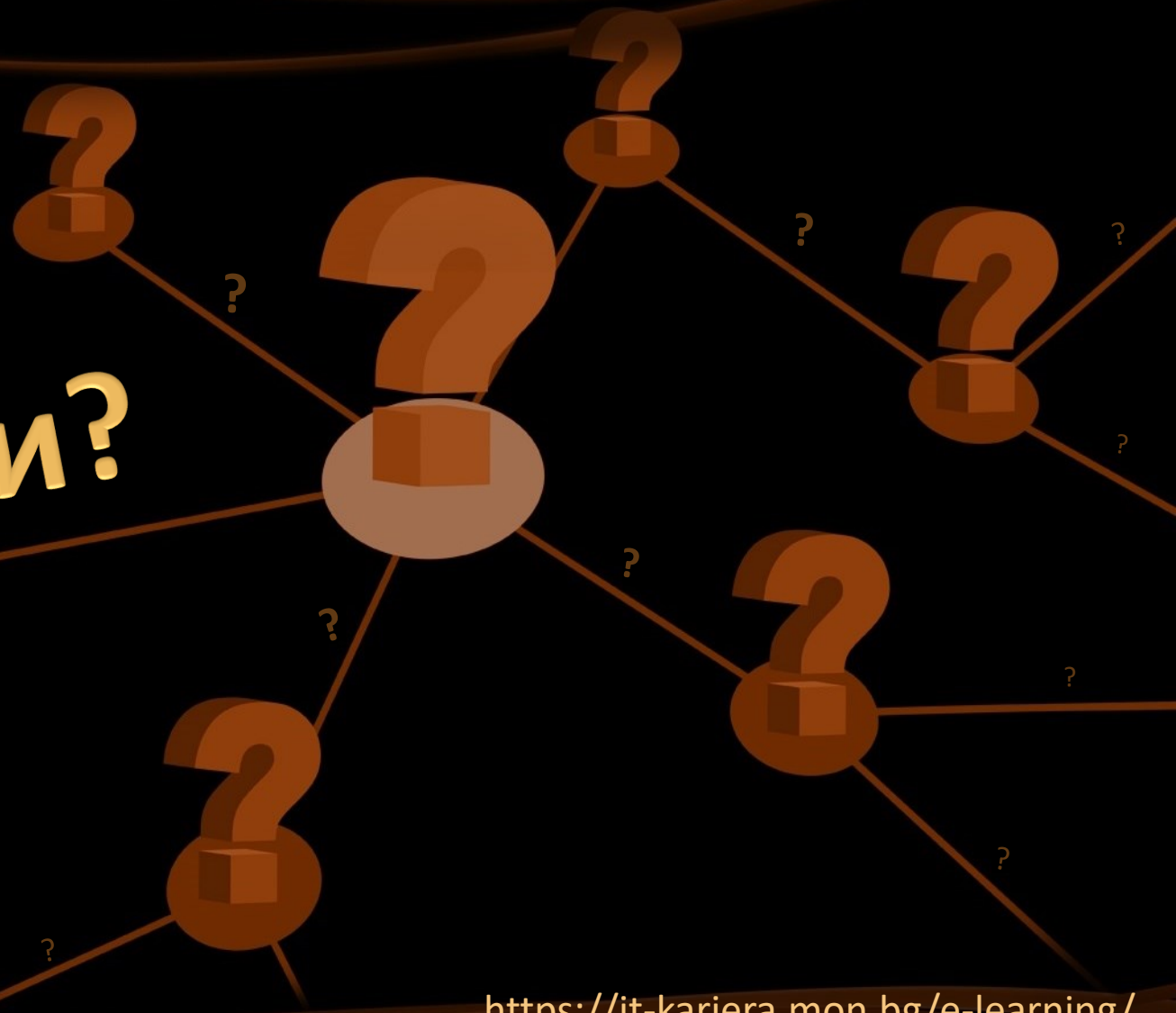
```
Employees employee =
  softUniEntities.Employees.First();
softUniEntities.Employees.Remove(employee);
softUniEntities.SaveChanges();
```

Маркира обекта за изтриване при следващото записване

Изпълнете командата за изтриване в SQL

Комуникация с база от данни

Въпроси?



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

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**