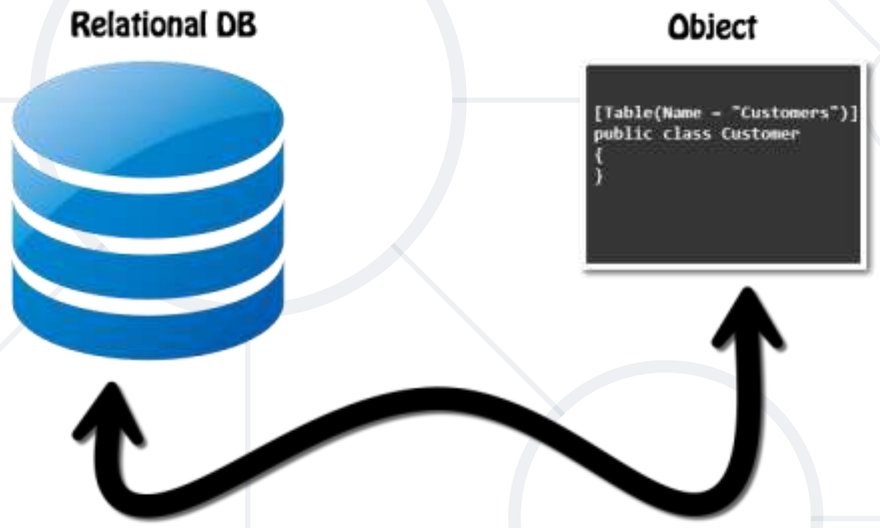
ORM Fundamentals

The ORM Concept, Config, CRUD Operations



```
[Table(Name = "Customers")]
public class Customer
{
}
```

SoftUni Team

Technical Trainers

Software University

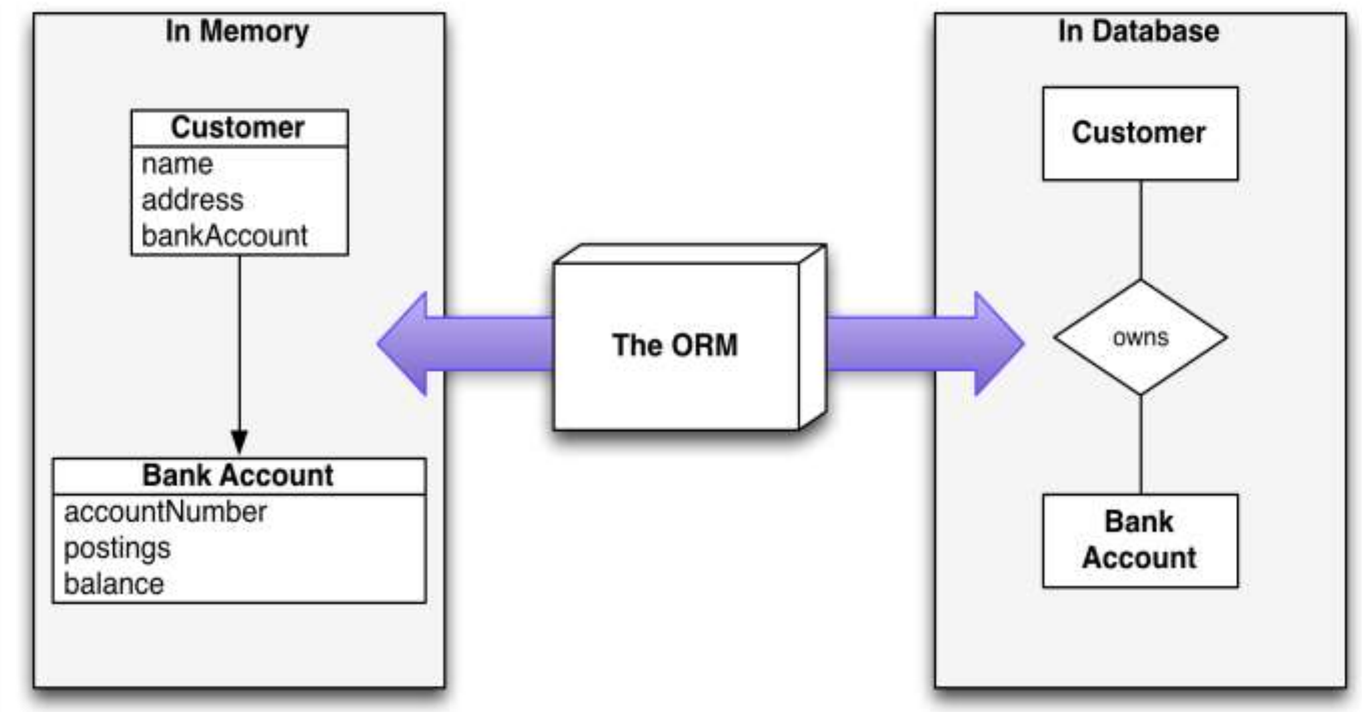
https://about.softuni.bg/

Table of Contents

Questions

sli.do

#csharp-db

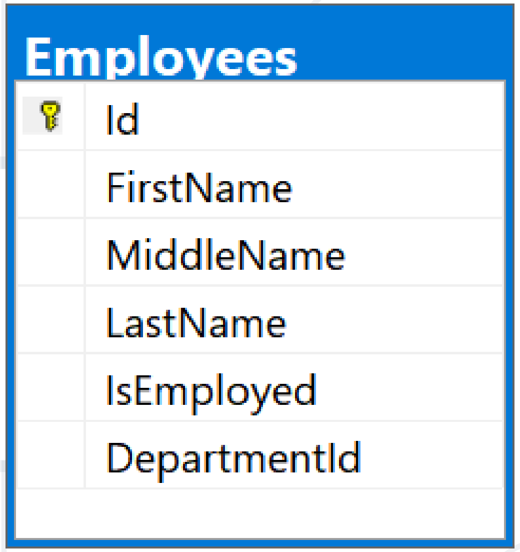


Introduction to ORM

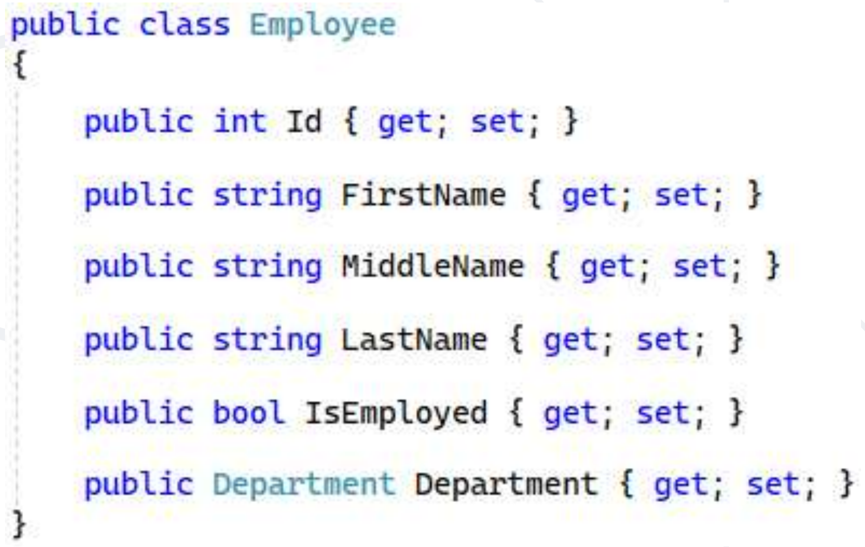
Object-Relational Mapping

What is ORM?

- **Object-Relational Mapping** (ORM) allows manipulating databases **using common classes and objects**
  - **Database Tables → C#/Java/etc. classes**



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; }
}
```

ORM Frameworks: Features

- **ORM frameworks** typically **provide** the following functionality
  - **Automatically generate SQL** to perform data operations

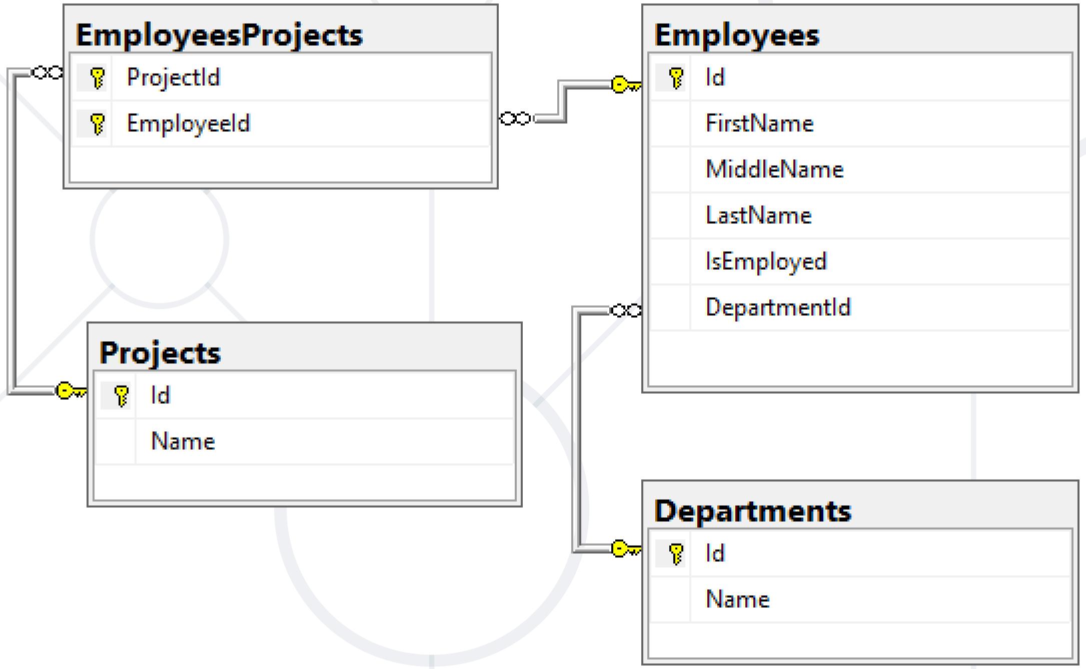
```
database.Employees.Add(new Employee
{
   FirstName = "George",
   LastName = "Peterson",
   IsEmployed = true    });
```

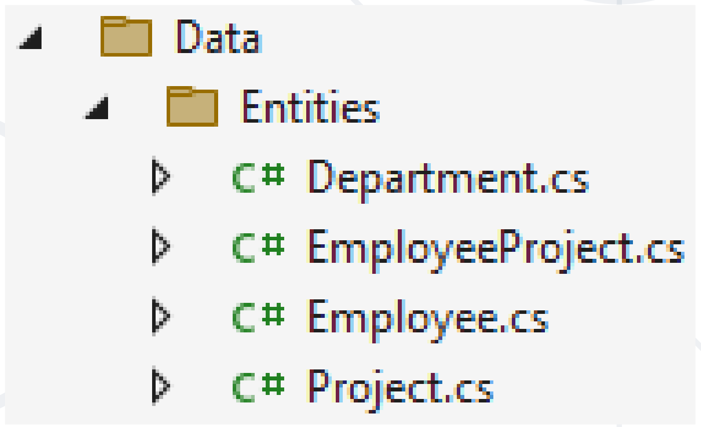
```
INSERT INTO Employees
(FirstName, LastName, IsEmployed)
VALUES
('George ', 'Peterson', 1)
```

  - **Create object model from database schema** (DB First model)
  - **Create database schema from object model** (Code First model)
  - **Query data by object-oriented API** (e.g., LINQ queries)

Database First Model

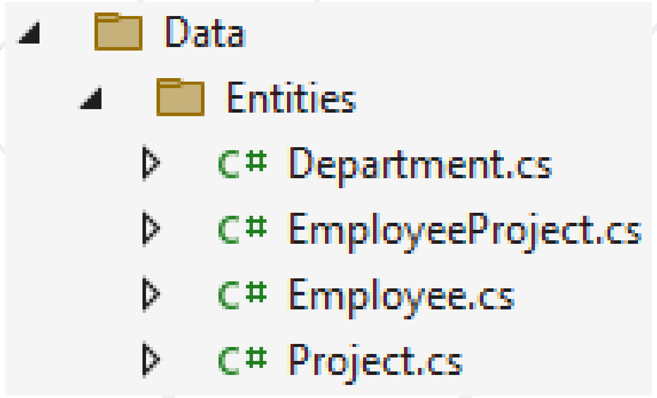
- **Database First model** - models the entity classes after the database

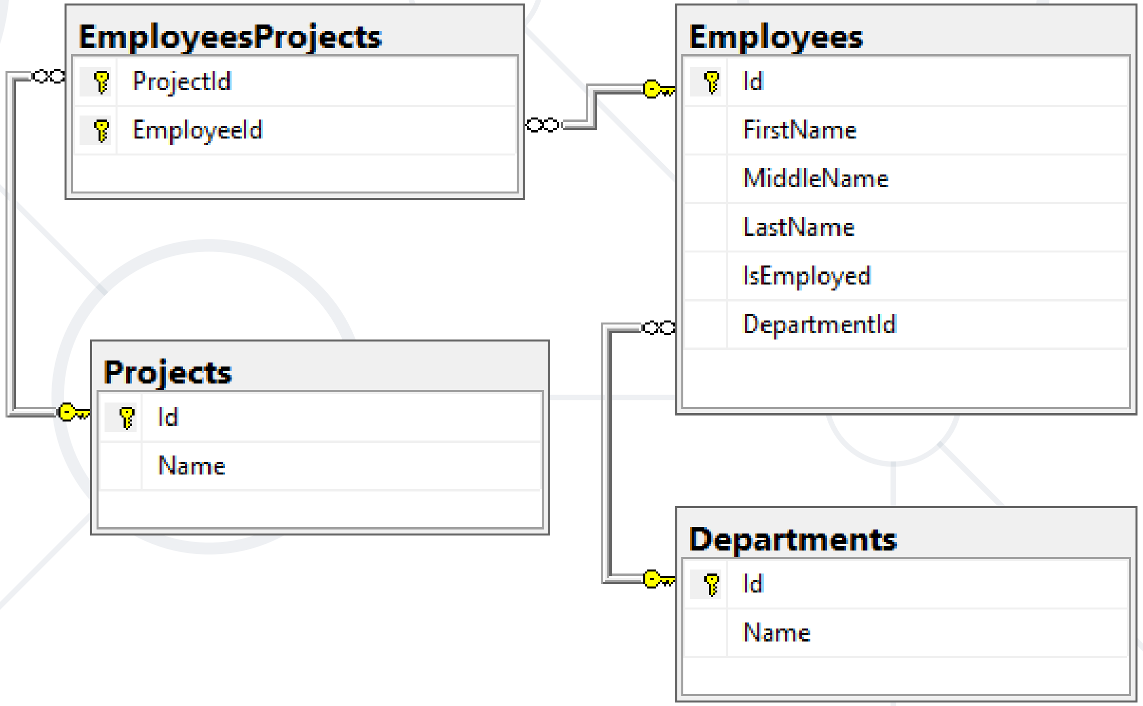




Code-First Model

- **Code-first model** - begins with classes that describe the model and then the ORM generates a database





ORM Advantages and Disadvantages

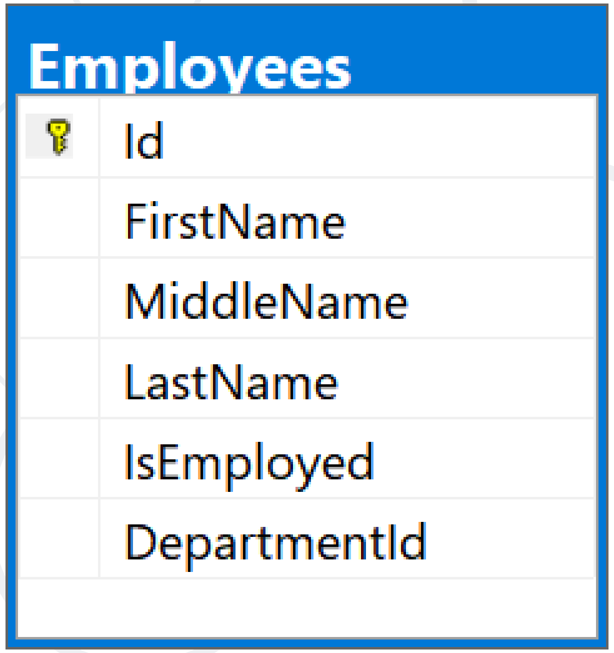
- **Advantages**
  - Developer productivity – **writing less code**
  - Abstract from differences between object and relational world
  - **Manageability of the CRUD operations** for complex relationships
  - **Easier maintainability**
- **Disadvantages**
  - **Reduced performance** (due to overhead or autogenerated SQL)
  - **Reduces flexibility** (some operations are hard to implement)

Entity Classes

Data Holders

Entity Classes

- **Entity classes** are regular **C# classes**
- Used for **storing** the **data** from the DB **in-memory**



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; }
}
```

Entity Classes: Navigation Properties (1)

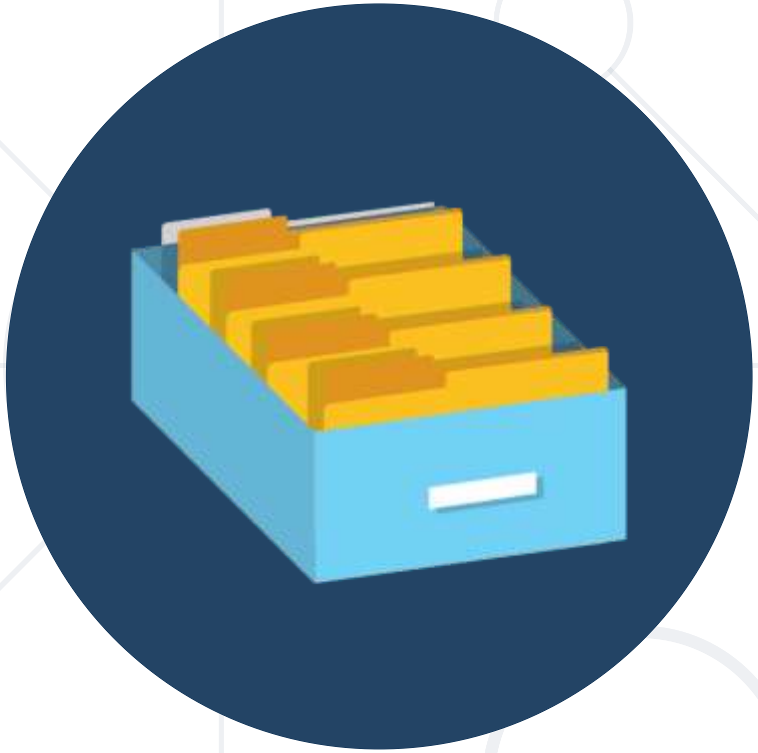
- Reference type properties
- Point to relevant object, connected by foreign key
- Set by the framework
- Example: Employee's Department

```
public class Employee
{
  public int Id { get; set; }
  ...
  public int DepartmentId { get; set; }
  public Department Department { get; set; }
}
```

Entity Classes: Navigation Properties (2)

- Navigation Properties can also be collections
- Usually of type **`ICollection<T>`**
- Hold all of the objects whose **foreign keys** are the same as the entity's **primary key**
- Set by the ORM framework

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



DbSet<T>

Specialized Collections

DbSet<T> Class

- Generic collection with additional features
- Each **`DbSet<T>`** corresponds to a single database table
- Inherits from **`ICollection<T>`**
  - **`foreach`**`-able`
  - Supports **LINQ** operations
- Usually several **`DbSets`** are a part of a **`DbContext`**

DbSet<T> Features

- Each **DbSet** tracks its own entities through a change tracker
- Has every other feature of an `ICollection<T>`
  - **Accessing** the elements (LINQ)
  - **Adding**/**Updating** elements
  - **Removing** an entity/a range of entities
  - **Checking** for element **existence**
  - Accessing the **count** of elements

DbContext

DbContext Class

- Holds several **DbSet<T>**
- Responsible for **populating** the **DbSets**
- Users create a **DbContext**, which **inherits** from **DbContext**
  - Using one **DbSet** per database table

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

Reading Data

Querying the DB Using ORM

Using DbContext Class

- First create instance of the **DbContext**

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

- In the constructor, you can pass a database connection string
- **DbContext properties**
  - All **entity classes** (tables) are listed as **properties**
    - e.g., `DbSet<Employee> Employees { get; }`

Reading Data with LINQ Query (1)

- Executing **LINQ-to-Entities** query over entity

```
var context = new SoftUniDbContext()

var employees = context.Employees
   .Where(e => e.JobTitle == "Design Engineer")
   .ToList();
```

- **Employees** property in the **DbContext**

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

Reading Data with LINQ Query (2)

- We can also use **extension methods** for constructing the query

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

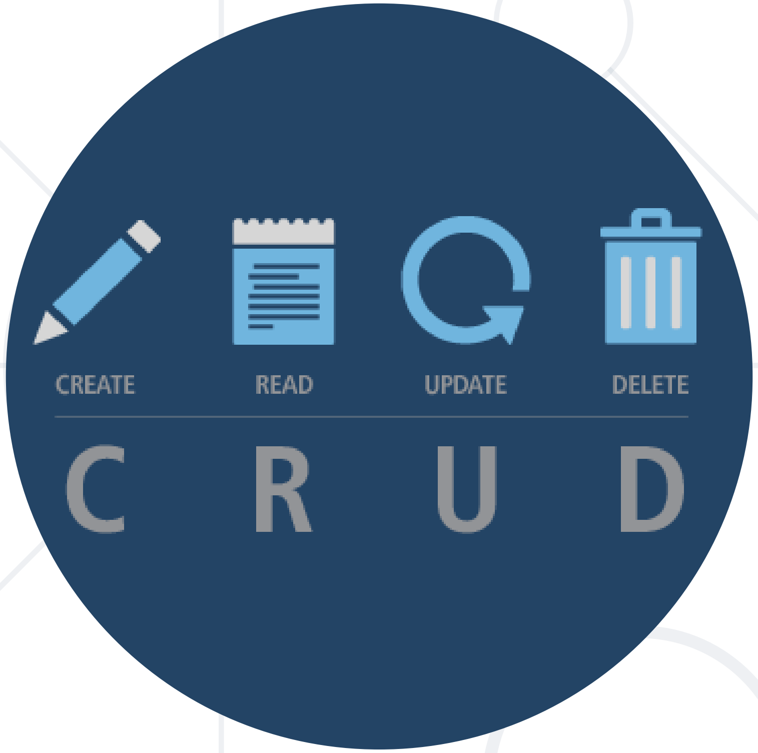
- Find element by **ID**

```
var context = new SoftUniEntities()
var project = context.Projects
     .FirstOrDefault(p => p.Id == 2);
Console.WriteLine(project.Name);
```

Change Tracking

Change Tracking

- Each **DbContext** instance tracks changes made to entities
  - These tracked entities in turn drive the changes to the database when **SaveChanges** is called
- Entity instances become tracked when they are
  - Returned from a query, executed against the database
  - Explicitly attached to the **DbContext** by **Add**, **Attach**, **Update** or similar methods
  - Detected as new entities connected to existing tracked entities



CRUD Operations

Creating New Entities

- To create a new table row use the method **Add(…)** of the **corresponding DbSet**

```
var project = new Project()
{
  Name = "Judge System"
};

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

Create a new Project object

Add the object to the DbSet

Execute SQL statements

Updating Existing Entities

- DbContext allows modifying entity properties and persisting them in the database
  - Just load an entity, modify it and call SaveChanges()
- The DbContext automatically tracks all changes made on its entity objects

```
var employee =
  context.Employees.FirstOrDefault();
employees.FirstName = "Alex";
context.SaveChanges();
```

SELECT the first employee

Execute an SQL UPDATE

Deleting Existing Data

- Delete is done by **Remove()** on the specified entity collection
- **SaveChanges()** method performs the delete action in the database

```
var employee =
  context.Employees.First();
context.Employees.Remove(employee);
context.SaveChanges();
```

Mark the entity for deleting at the next save

Execute the SQL DELETE command

Summary

- **ORM frameworks** map database schema to objects in a programming language
- **LINQ** can be used to query the DB through the **DB Context**

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

pre zero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

SmartIT

createX

SUPER HOSTING .BG

Educational Partners

Software University

CODE IT UP

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License