# Introduction to Entity Framework Core

## The ORM Concept, Config, CRUD Operations


**SoftUni Team**

**Technical Trainers**

Software University

SoftUni Foundation





**Software University**

http://softuni.bg

# Table of Contents



# Have a Question?

SoftUni Foundation

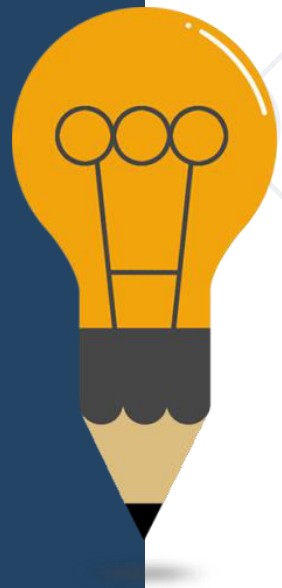
## sli.do

# #csharp-db


# Entity Framework Core

## Overview and Features

# Entity Framework Core: Overview

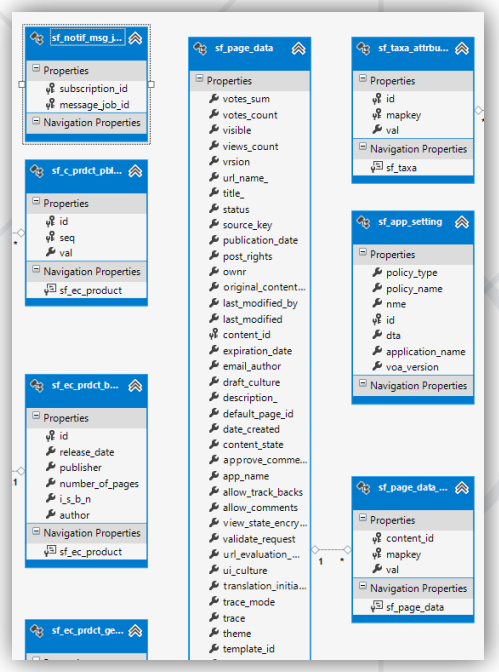


- The standard **ORM framework** for **.NET** and **.NET Core**
- Provides LINQ-based data queries and **CRUD** operations
- Automatic **change tracking** of in-memory objects
- Works with many relational databases (with different providers)
- Open source with independent release cycle

# EF Core: Basic Workflow

SoftUni Foundation

1. Define the data model (**Code First** or **Scaffold from DB**)




2. Write & execute query over **IQueryable**

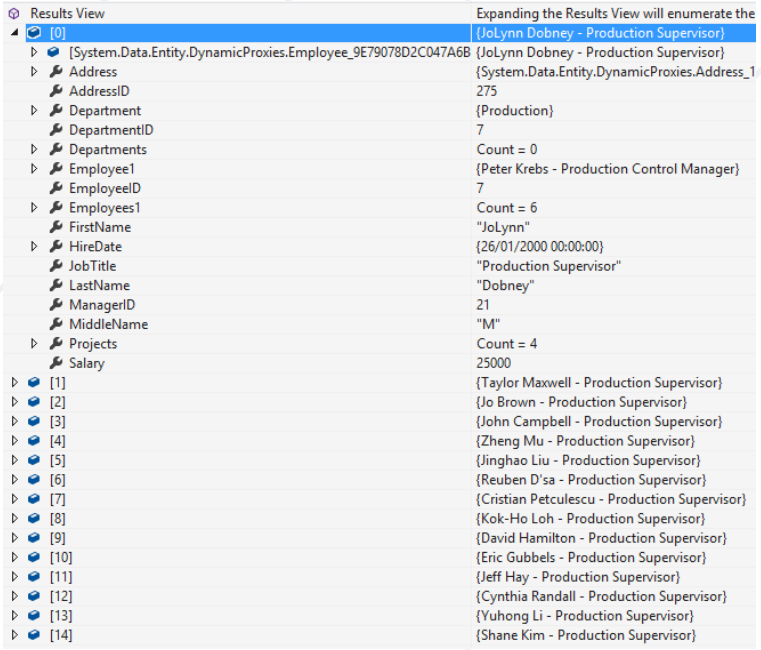
```
var toolName = "";

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;
}
```

3. EF generates & executes an **SQL query** in the **DB**

```
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
```

# EF Core: Basic Workflow (2)

4. EF transforms the query results into .NET objects



| 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} |

5. Modify data with C# code and call "**Save Changes()**"

```
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();
}
```

6. Entity Framework generates & executes SQL command to modify the DB

```
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: Setup

SoftUni Foundation

- To add **EF Core support** to a project in **Visual Studio**:
  - Install it from **Package Manager Console**

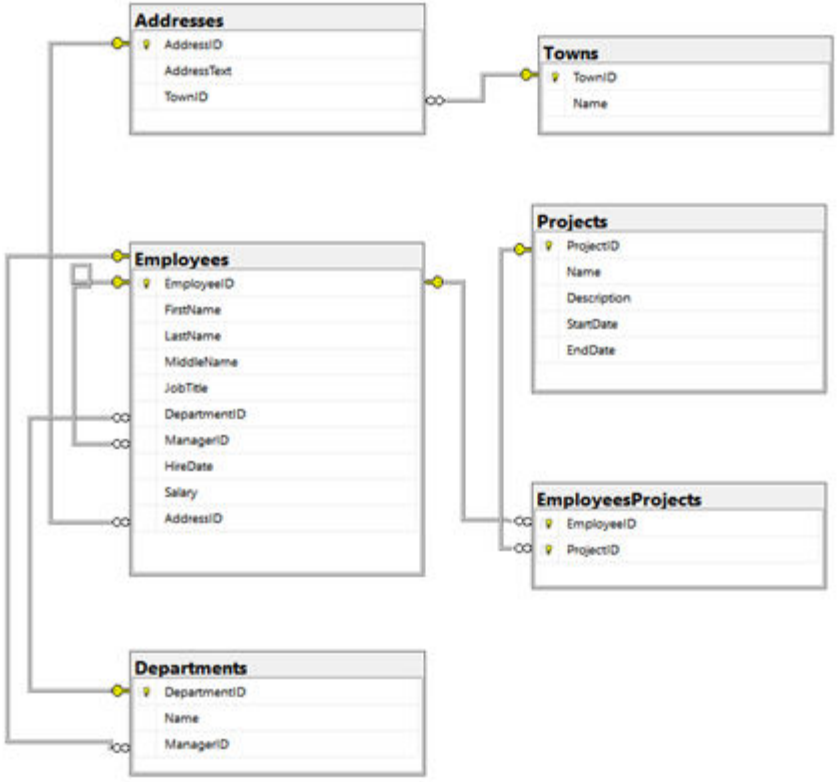
```
Install-Package Microsoft.EntityFrameworkCore
```

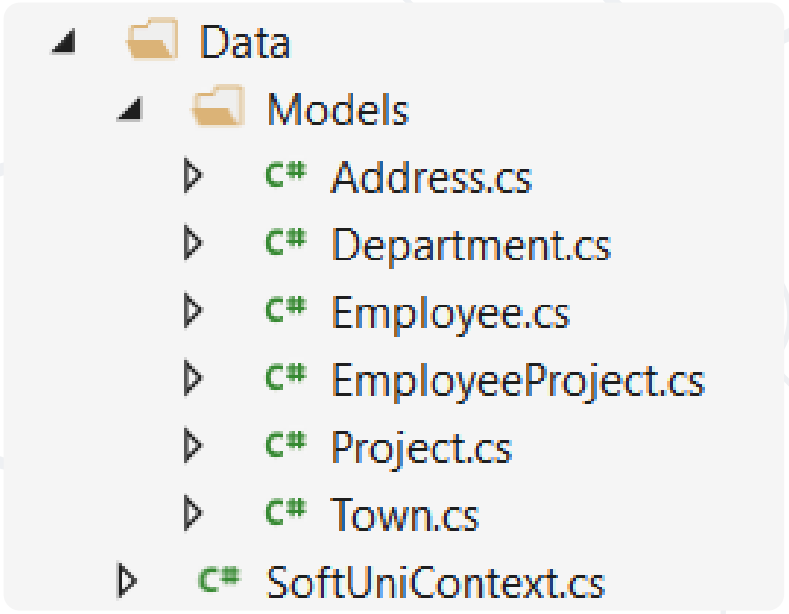
  - EF Core is modular – any **data providers** must be installed too:

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

# Database First Model

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



# Database First Model: Setup

- Scaffolding DbContext from DB with **Scaffold-DbContext** command in **Package Manager Console**:

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

- Scaffolding requires the following packages beforehand:

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

# EF Components

SoftUni Foundation

- The **DbContext** class
  - Holds the **database connection** and the **entity classes**
  - Provides **LINQ-based** data access
  - Provides **identity tracking**, **change tracking**, and an API for **CRUD** operations
- Entity classes
  - Hold **entities** (objects with their attributes and relations)
  - Each database **table** is typically mapped to a single **C# class**

# EF Components (2)

- **Associations** (relationship mappings)
  - An association is a primary key / foreign key-based relationship between two entity classes
  - Allows navigation from one entity to another

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

- **Concurrency control**
  - **Entity Framework** uses **optimistic concurrency control**
    - No locking by default
  - Automatic concurrency conflict detection

# Reading Data

## Querying the DB using Entity Framework

# The DbContext Class

SoftUni Foundation

- **DbContext** provides:
  - CRUD Operations
    - A way to **access entities**
    - Methods for **creating** new entities (**Add()** method)
    - Ability to **manipulate database data by** modifying **objects**
- Easily navigate through **table relations**
- Executing **LINQ queries** as native **SQL queries**
- Managing database **creation**/**deletion**/**migration**

# Using DbContext Class

SoftUni Foundation

- First create instance of the **DbContext**:

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

- In the constructor you can pass a database connection string
- **DbContext** properties:
  - **Database** – **EnsureCreated**/**Deleted** methods, DB Connection
  - **ChangeTracker** – Holds info about the **automatic change tracker**
  - All entity classes (tables) are listed as properties
    - e.g. **DbSet<Employee> Employees { get; set; }**

# Reading Data with LINQ Query

SoftUni Foundation

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

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

EF translates this to an SQL query

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

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

# Reading Data with LINQ Query

SoftUni Foundation

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

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

**ToList()** materializes the query

- Find element by **ID**

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

# LINQ: Simple Operations

- **Where()**
  - Searches by given condition
- **First/Last()** / **FirstOrDefault/LastOrDefault()**
  - Gets the **first**/**last** element which matches the condition
  - Throws **InvalidOperationException** without **OrDefault**
- **Select()**
  - Projects (conversion) collection to another type
- **OrderBy()** / **ThenBy()** / **OrderByDescending()**
  - Orders a collection

# LINQ: Simple Operations (2)

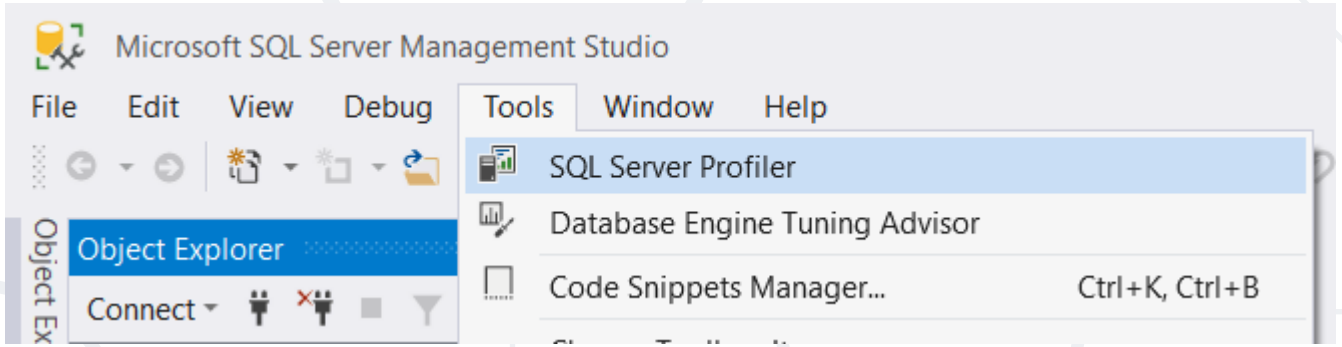
SoftUni Foundation

- **Any()**
  - Checks if any element matches a condition
- **All()**
  - Checks if all elements match a condition
- **Distinct()**
  - Returns only unique elements
- **Skip()** / **Take()**
  - Skips or takes X number of elements

# Logging the Native SQL Queries

SoftUni Foundation

- Queries sent to SQL Server can be monitored with SQL Server Profiler
  - Included in **SQL Server Management Studio**:




- Queries can also be monitored with Express Profiler

https://expressprofiler.codeplex.com/


# CRUD Operations

## With Entity Framework

# Creating New Data

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

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

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

Create a new **Project** object

Add the object to the **DbSet**

Execute SQL statements

# Cascading Inserts

- We can also add cascading entities to the database:

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

- The **Project** will be added when the **Employee** entity (employee) is inserted to the database

# Updating Existing Data

- **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

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

SELECT the first order

Execute an SQL UPDATE

# Deleting Existing Data

SoftUni Foundation

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

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

Mark the entity for deleting at the next save

Execute the SQL DELETE command

# Summary

- ORM frameworks maps database schema to objects in a programming language
- Entity Framework Core is the standard .NET ORM
- LINQ can be used to query the DB through the DB context

# Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

# SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

INDEAVR
Serving the high achievers

INFRAGISTICS®

STEMO®
Computer Systems & Software

SUPERHOSTING.BG

# SoftUni Organizational Partners

SoftUni Foundation




# Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg


# License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license