

LINQ

Language Integrated Query in Entity Framework Core



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

sli.do

#csharp-db

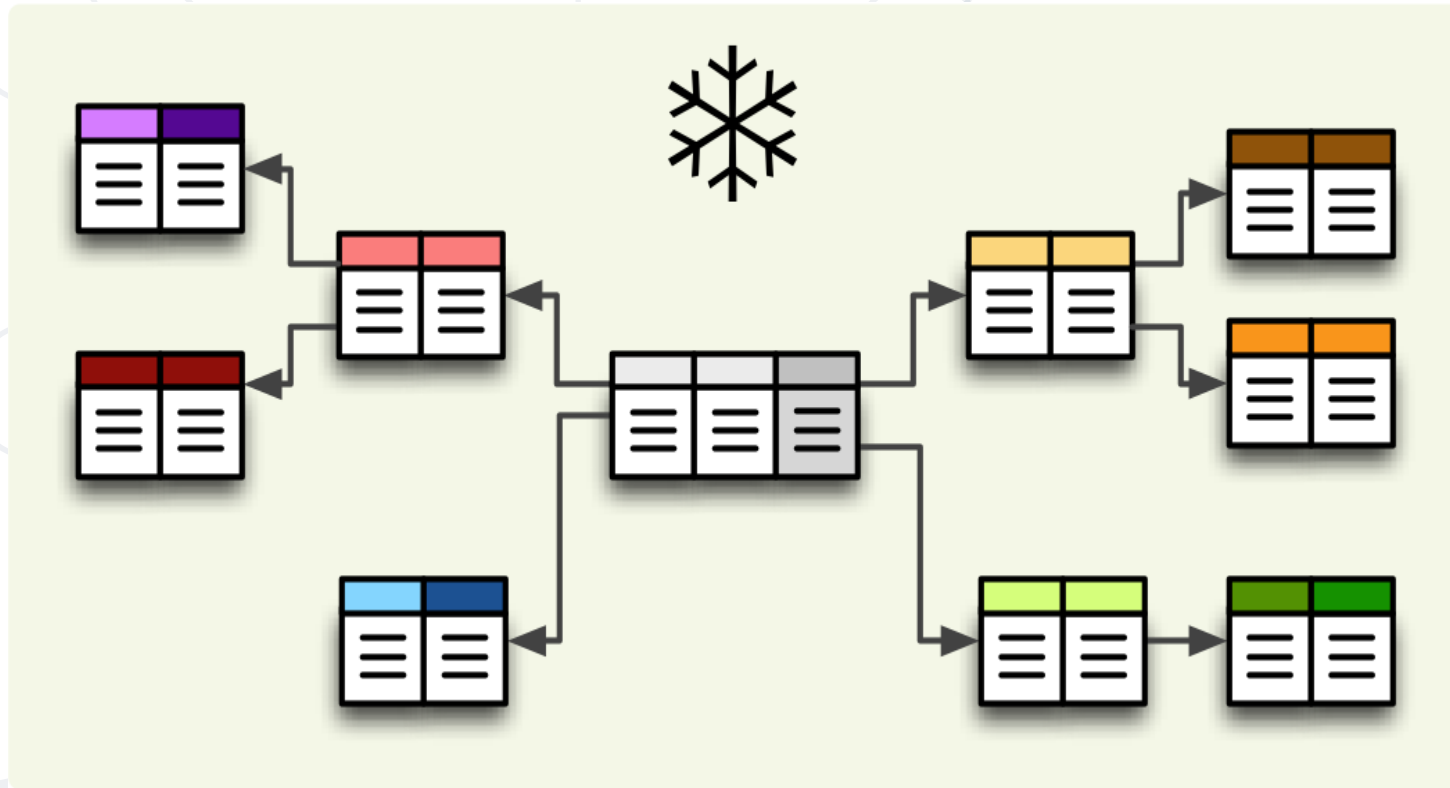
1. LINQ

- Filtering
- Select / Projection
- Aggregation
- Joining
- SelectMany

2. IEnumerable vs IQueryable

3. Result Models





Filtering and Aggregating Tables

Select, Join and Group Data Using LINQ

- **Where**

- Selects values that are based on a predicate function
- Syntax

```
string[] words = { "the", "quick", "brown", "fox", "jumps" };
```

```
IEnumerable<string> query =  
    words.Where(word => word.Length == 3);
```

```
IEnumerable<string> query = from word in words  
                           where word.Length == 3  
                           select word;
```

- Limit network traffic by reducing the queried columns
- Syntax

```
var employeesWithTown = context
    .Employees
    .Select(employee => new
    {
        EmployeeName = employee.FirstName,
        TownName = employee.Address.Town.Name
    });
```

- SQL Server Profiler

```
SELECT [employee].[FirstName] AS [EmployeeName], [employee.Address.Town].[Name] AS [TownName]
FROM [Employees] AS [employee]
LEFT JOIN [Addresses] AS [employee.Address] ON [employee].[AddressID] = [employee.Address].[AddressID]
LEFT JOIN [Towns] AS [employee.Address.Town] ON [employee.Address].[TownID] =
[employee.Address.Town].[TownID]
```

Good Reasons Not to Use Select

- Data that is selected is **not** of the **initial entity type**
 - **Anonymous type**, generated at runtime

```
[?] (local variable) System.Collections.Generic.List<'a> employeesWithTown
```

Anonymous Types:

```
'a is new { string EmployeeName, string TownName }
```

Local variable 'employeesWithTown' is never used

- **Data cannot be modified** (updated, deleted)
 - Entity is of a **different type**
 - Not associated with the **context** anymore

- Aggregate functions perform calculations on a set of input values and return a value
 - **Average** - Calculates the average value of a collection of values
 - **Count** - Counts the elements in a collection, optionally only those elements that satisfy a predicate function
 - **Max** and **Min** - Determine the maximum and the minimum value in a collection
 - **Sum** - Calculates the sum of the values in a collection

Joining Tables in EF: Using Join()

- Join tables in EF with **LINQ / extension methods** on **IEnumerable<T>** (like when joining collections)

```
var employees =  
    softUniEntities.Employees.Join(  
        softUniEntities.Departments,  
        (e => e.DepartmentID),  
        (d => d.DepartmentID),  
        (e, d) => new {  
            Employee = e.FirstName,  
            JobTitle = e.JobTitle,  
            Department = d.Name  
        }  
    );
```

- Grouping also can be done by LINQ
 - The same way as with collections in LINQ
- Grouping with LINQ

```
var groupedEmployees =  
    from employee in softUniEntities.Employees  
    group employee by employee.JobTitle;
```

- Grouping with extension methods

```
var groupedCustomers = softUniEntities.Employees  
    .GroupBy(employee => employee.JobTitle);
```

SelectMany – Example

```
public class PhoneNumber
{
    public string Number { get; set; }
}
```

```
public class Person
{
    public IEnumerable<PhoneNumber> PhoneNumbers { get; set; }
    public string Name { get; set; }
}
```

SelectMany – Example

```
IEnumerable<Person> people = new List<Person>();

// "Select" gets a list of lists of phone numbers
IEnumerable<IEnumerable<PhoneNumber>> phoneLists =
    people.Select(p => p.PhoneNumbers);

// "SelectMany" flattens it to just a list of phone numbers
IEnumerable<PhoneNumber> phoneNumbers =
    people.SelectMany(p => p.PhoneNumbers);

// To include data from the parent in the result pass an expression
// to the second parameter (resultSelector) in the overload
var directory = people.SelectMany(p => p.PhoneNumbers,
    (parent, child) => new { parent.Name, child.Number });
```



IEnumerable vs IQueryable

- **IEnumerable<T>** is an interface that is available in the **System.Collection.Generic** namespace
- Implementation of the Iterator design pattern
- **IEnumerable** or **IEnumerable<T>** interface should be used only for **in-memory data objects**
- LINQ methods over **IEnumerable<T>** use **Func<>** parameters

- **IQueryable<T>** is an interface and it is available in **System.Linq**
- Provides functionality to evaluate queries against a specific **data source** where the type of the data may not be specified
- The **IQueryable** interface is intended for implementation by query providers
- LINQ methods over **IQueryable<T>** use **Expression<Func<>>** parameters (expression trees)
 - Entity Framework can convert expression trees directly into SQL

- **IEnumerable<T>**

- **System.Collections.Generic**

- Base type for almost all .NET collections
 - LINQ methods works with **Func<>**
 - Good for **in-memory** data


- **IQueryable<T>**

- **System.Linq** namespace
 - Derives the base interface from **IEnumerable<T>**
 - LINQ methods works with **Expression<Func<>>**
 - Good for queries over **data stores** such as databases



Result Models

Simplifying Models

- 
- **Select()**, **GroupBy()** can work with **custom classes**
 - Allow you to **pass them** to methods and use them as a return type
 - Require some **extra code** (class definition)
 - Sample Result Model

```
public class UserResultModel
{
    public string FullName { get; set; }
    public string Age { get; set; }
}
```

- Assign the fields as you would with an anonymous object

```
var currentUser = context.Users
    .Where(u => u.Id == 8)
    .Select(u => new UserResultModel
    {
        FullName = u.FirstName + " " + u.LastName,
        Age = u.Age
    })
    .SingleOrDefault();
```

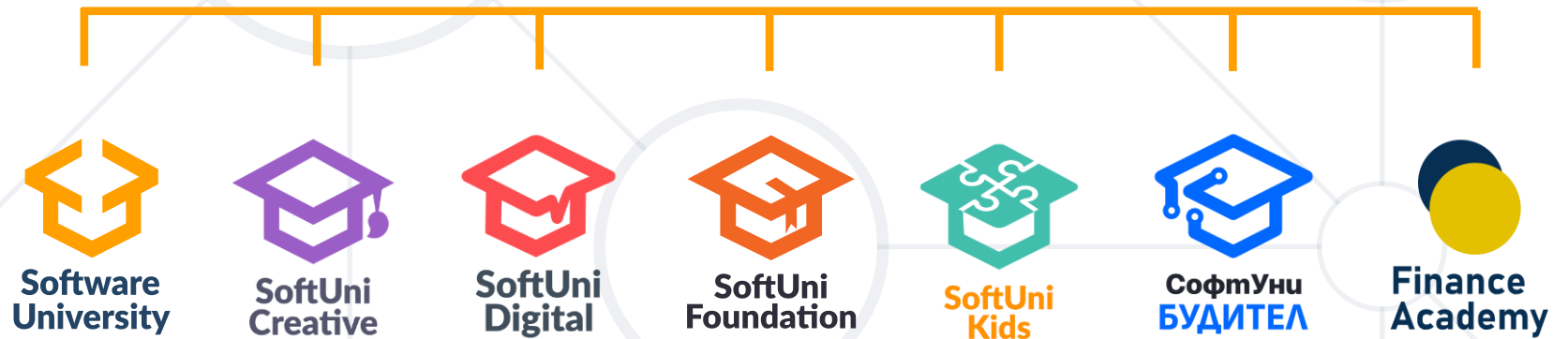
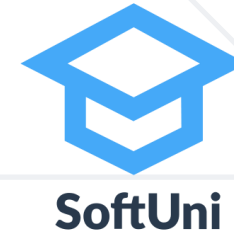
- The new type can be used in a method signature

```
public UserResultModel GetUserInfo(int Id) { ... }
```

- **LINQ**
 - Filtering, Aggregation, SelectMany, Joins
- **IEnumerable**
- **IQueryable**
- **Differences** between IEnumerable and IQueryable
- **Result Models**



Questions?



SoftUni Diamond Partners



THE CROWN IS YOURS



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>



- 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

