**ASP.NET Core Databases**



##### Working with Entity Framework Core



**SoftUni Team**

[](https://softuni.org/)Technical Trainers

**Software University**

[**https://softuni.bg**](https://softuni.bg/)



###### Entity Framework Core

* + Code First

Approach



**Table of Content**

###### EF Core Components

1. EF Core Configuration
   * Fluent API
2. CRUD Operations
3. Database Migrations



**Questions?**

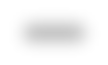
# sli.do

#csharp-web



## Code First Approach

Entity Framework Core



### Entity Framework Core: Overview



* The standard **ORM framework** for **.NET**

###### Provides LINQ-based data queries and **CRUD**

operations

* Automatic **change tracking** of in-memory objects

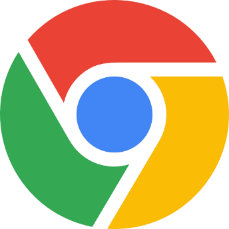
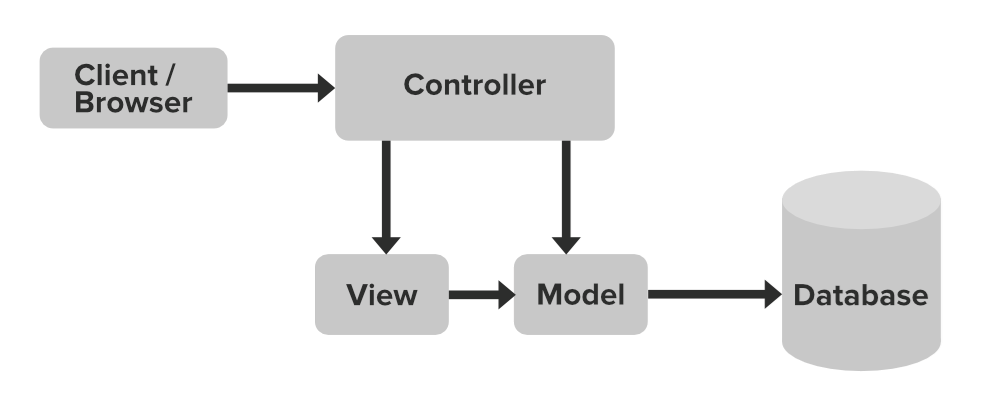
###### Works with many relational databases (with different providers)

* Open source with independent release cycle



**ASP.NET Core MVC + Entity Framework**

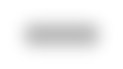
**ASP.NET MVC**



**controllers**

**EF Core**

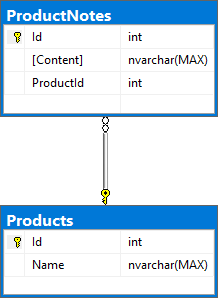
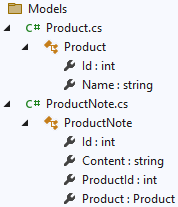
**Razor views (.cshtml)**



### What is the Code First Approach?



* **Code First** means to write the .NET classes and let EF Core create the **database** from the **mappings**

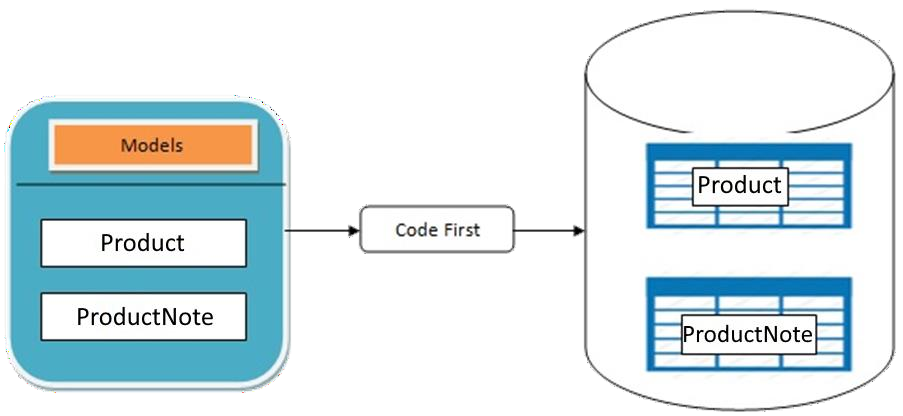




**Why Use Code First?**

* Write code **without** having to define **mappings** in XML or

**create** database **tables**

* Define objects in **C# format**
* Enables database persistence with no configuration
* Changes to code can be **reflected**

(migrated) in the schema

* Data Annotations or Fluent API

describe properties

* + **Key**, **Required**, **MinLength**, etc.

1. Define the data

model (**Code First** or **Scaffold from DB**)

1. Write & execute

query over

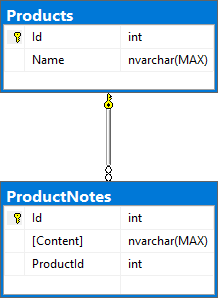
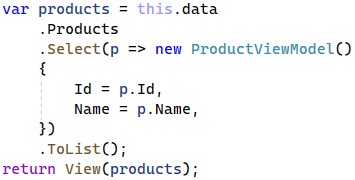
IQueryable

1. EF generates &

executes an **SQL query** in the **DB**



**Code First Basic Workflow (1)**

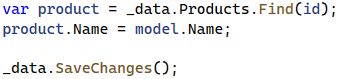
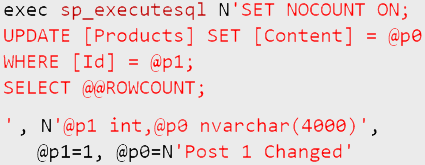
1. EF transforms the query results into

.NET objects

1. Modify data with C# code and call "**Save Changes()**"
2. Entity Framework generates & executes SQL command to modify the DB



**Code First Basic Workflow (2)**



**DbContext**

**Product**

**ProductNotes**

## EF Core Components

#### Overview of System Objects

###### Bunch of normal C# classes (POCO)

* + May contain **navigation properties** for **table relationships**

**public class ProductNote**

**{**

**Primary key**

**public int Id { get; set; }**

**public string Content { get; set; } public int ProductId { get; set; } public Product Product { get; set; }**

**Foreign key**

**} Navigation property**



**Domain Classes (Models) (1)**

* Recommended to be in a **separate class library**



**Domain Classes (Models) (2)**

###### Another example of a domain class (model)

**public class Product**

**{**

**public int Id { get; set; } public string Name { get; set; }**

**public IList<ProductNote> ProductNotes { get; set; }**

**= new List<ProductNote>();**

**}**

**One-to-many relationship**



**The DbContext Class**

* Usually named after the database, e.g., **ShoppingListDbContext**

Manages model classes using **DbSet<T>** type

**Inherits the**

**DbContext class**

**public class ShoppingListDbContext : DbContext**

* Easily navigate through **table relations**
* Managing database **creation**/**deletion**/**migration**
* Executing **LINQ queries** as native **SQL queries**
* **DbContext** properties
  + **Database** – **EnsureCreated**/**Deleted** methods, DB Connection
  + **ChangeTracker** – holds info about the **automatic change tracker**

### Defining DbContext Class

using Microsoft.EntityFrameworkCore;

EF Reference

public class ShoppingListDbContext : DbContext

**{**

public ShoppingListDbContext

Accepts options through the constructor

(DbContextOptions<ShoppingListDbContext> options)

: base(options)

=> Database.EnsureCreated(); Collections of entities

public DbSet<Product> Products { get; set; }

public DbSet<ProductNote> ProductNotes { get; set; }

protected override void OnModelCreating(ModelBuilder builder)

**{**

builder.Entity<Product>()



.HasMany(p => p.ProductNotes)

.WithOne(r => r.Product);

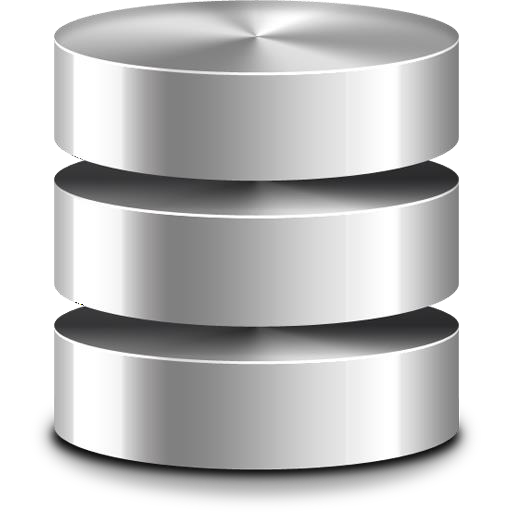
**}**

**}**

Use the Fluent API to describe

our table relations to EF Core

15



## EF Core Configuration

#### NuGet Packages, Configuration

##### To add EF Core support to a project in Visual Studio

* + Install it from **Package Manager Console**

**Install-Package Microsoft.EntityFrameworkCore**

* + Or using **.NET Core CLI**

**dotnet add package Microsoft.EntityFrameworkCore**

##### EF

Core is modular – any **data providers** must be

##### installed too



**EF Core Setup**

**Microsoft.EntityFrameworkCore.SqlServer**

* To use the Entity Framework Core **CLI tools**, install also

**Microsoft.EntityFrameworkCore.Design**



**How to Connect to SQL Server? (1)**

* In ASP.NET Core **connection string** is in the **appsettings.json**

file and has the following **properties**

**"ConnectionStrings": {**

**"DefaultConnection": "Server=(localdb)\\mssqllocaldb; Database=ShoppingList;Trusted\_Connection=True; MultipleActiveResultSets=true"}**



**How to Connect to SQL Server? (2)**

* Use the **DbContext** and tell it to use SQL with the **connection string** in in the **Program** class

**var connectionString = builder**

**.Configuration**

**.GetConnectionString("DefaultConnection");**

**builder**

**.Services**

**.AddDbContext<ShoppingListDbContext>(**

**x => x.UseSqlServer(connectionString));**



**Database.EnsureCreated()**

###### When you create the DB context, you can call

**Database.EnsureCreated()**

* This will **create the DB + schema**, when the DB is missing

**Database Exists?**

Yes

No

**Use Database**

**Create Database**

###### **EnsureCreated()** does not use migrations  you should drop the enrite DB when you change the DB schema



**Database.EnsureCreated() – Example**

**public class ShoppingListDbContext : DbContext**

**{**

**public ShoppingListDbContext( DbContextOptions<ShoppingListDbContext> options)**

**: base(options)**

**=> Database.EnsureCreated();**

**// This will create the DB schema if the DB does not exist**

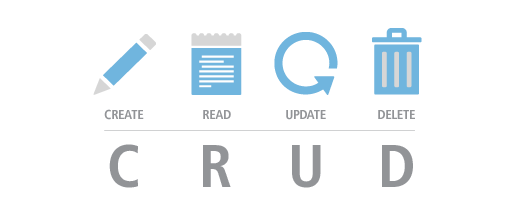
**// Any change in the data entities will not change the DB**

**// You should update the DB by hand**

**// or drop and re-create the DB after each entity change!**

**…**

**}**



## CRUD in ASP.NET Core MVC with EF

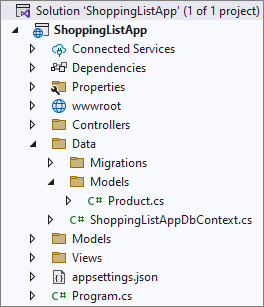
#### The "ShoppingList" App



**The "ShoppingList" App**

* Create an MVC app with the **models** and **db context**

###### from the previous slides

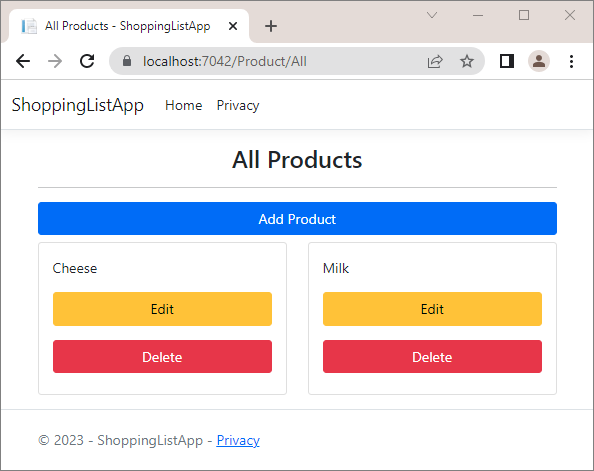
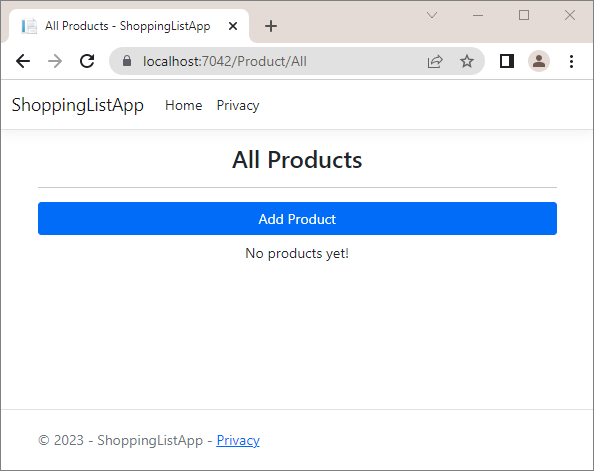


* Perform **CRUD operations** on the database to create the following functionalities
  + Display all products
  + Add a product 
  + Edit a product
  + Delete a product



**The "All Products" Page (Reading Data)**

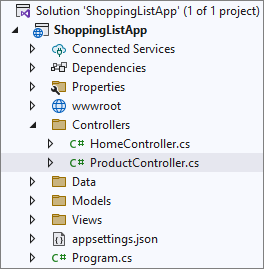
* It should display **all added products** with their **content** + **[Edit]** and **[Delete]** buttons + **[Add Product]** button



**ProductController Class**

* Create a new **ProductController** in the "**Controllers**" folder
* Inject the **ShoppingListAppDbContext** through the

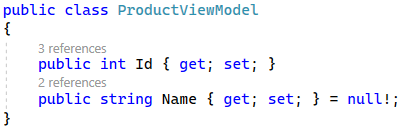
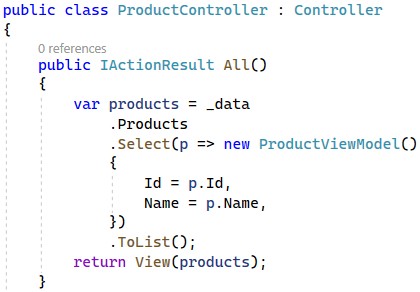
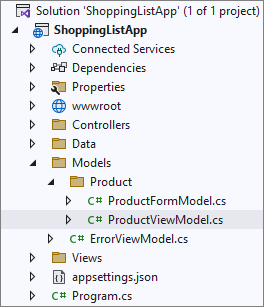
constructor



* + And assign it to a **variable** to use it



### Reading Data (Controller + Model)

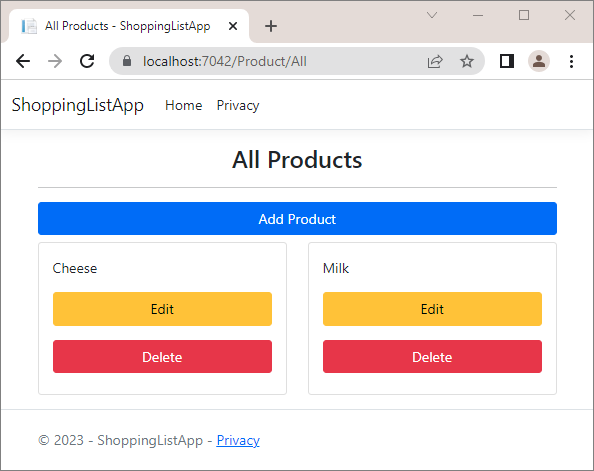
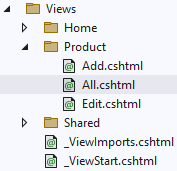


Extract the products from the database

Project products to a model collection

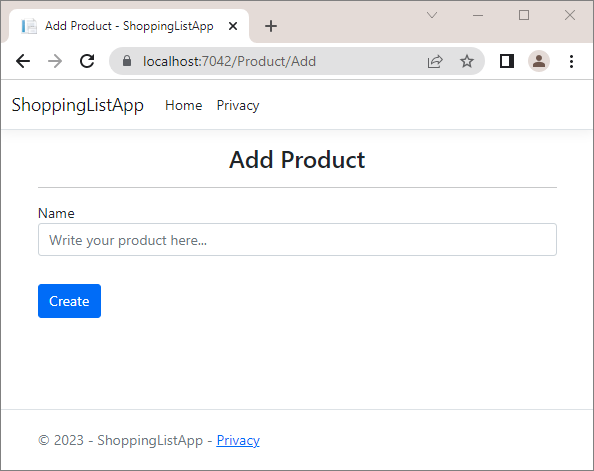
Passes the model collection to a view

### Reading Data (View)



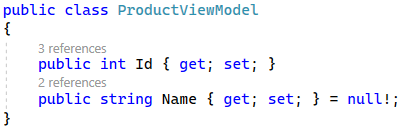
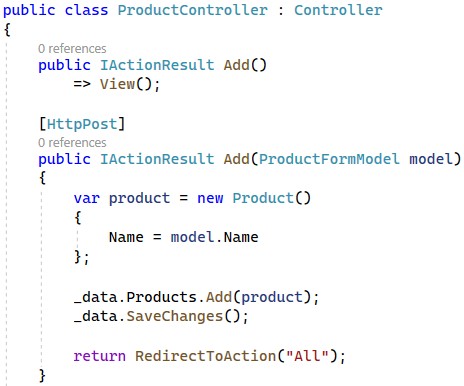
**The "Add Product" Page (Creating New Data)**

* It should display a **form** for **adding a product**



**Creating New Data (Controller + Model)**

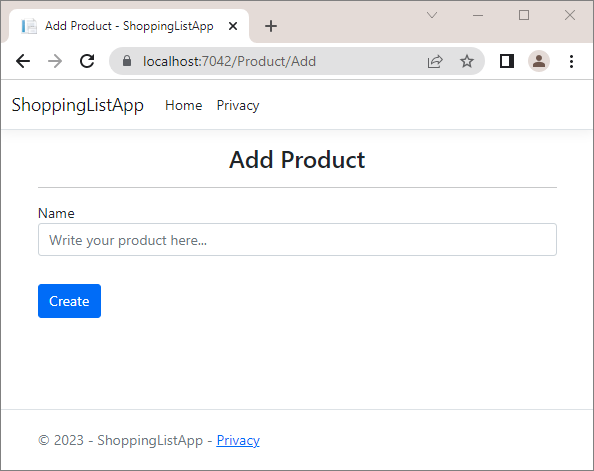
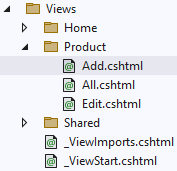
Create a new Product object



Add the object to the DbSet

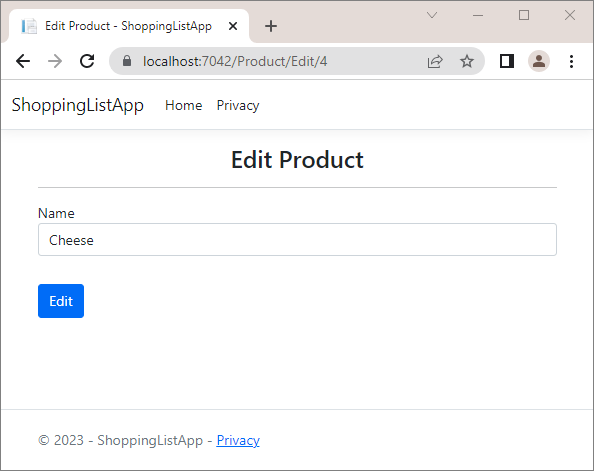
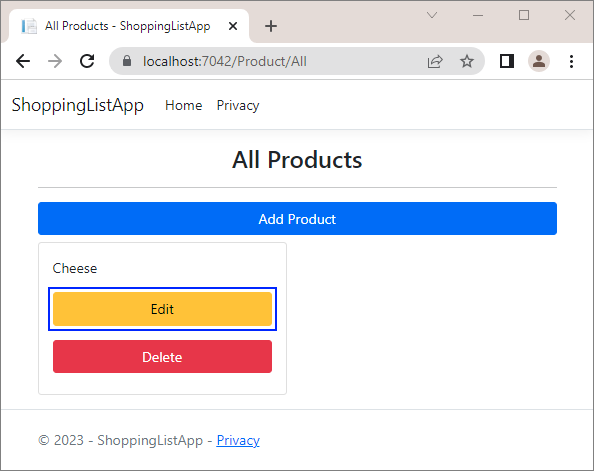
Execute SQL statements

### Creating New Data (View)



**The "Edit Product" Page (Updating Existing Data)**

* + To **edit a product**, click on its **[Edit]** button



* + It should display a **form for editing a product** with the product data in the fields

### Updating Existing Data (Controller)

HTTP GET  display

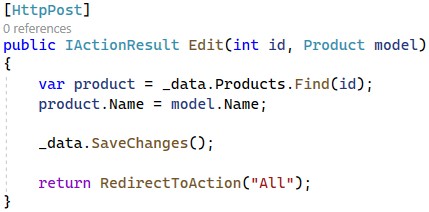
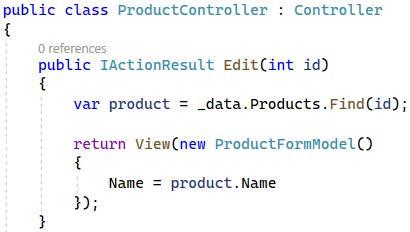
the edit form

HTTP POST 

update the DB

SELECT the product for update

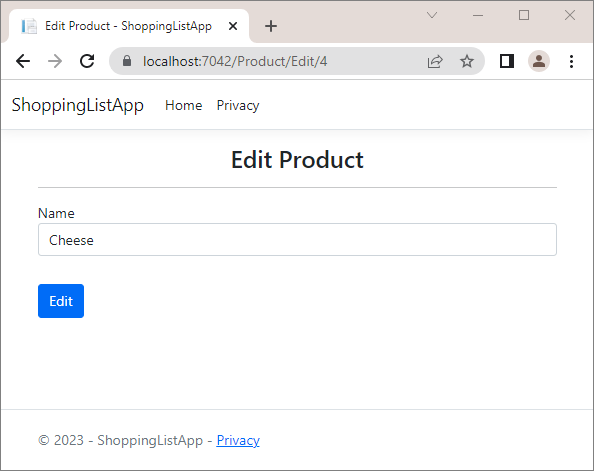
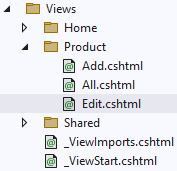
Pass a product



model to a view

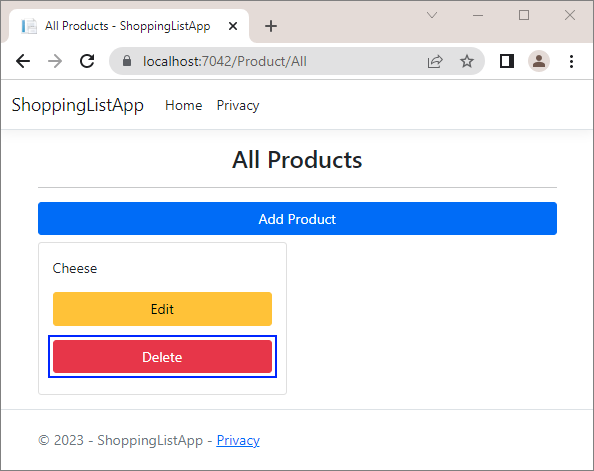
Execute an SQL UPDATE

### Updating Existing Data (View)



**Deleting Existing Data**

* To **delete a product**, press its **[Delete]** button



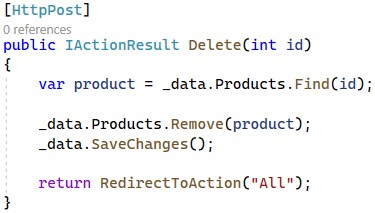
**o**

**It will send a "POST" request t the controller**

Execute the SQL DELETE

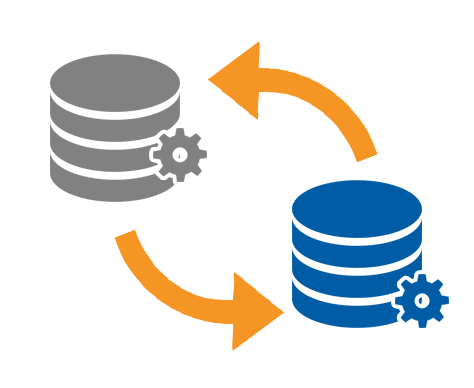


**Deleting Existing Data (Controller)**



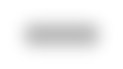
command

Mark the entity for deleting at the next save



## Database Migrations

Scripts for Modifying Table Structure in the DB



### What Are Database Migrations?



* Updating database schema **without losing data**
  + Adding/dropping tables, columns, etc.
* **Migrations** in EF Core keep their **history**
  + Entity Classes, DB Context versions are all **preserved**
* **Automatically** generated by certain EF tools



* To **add a migration** in EF Core
  + Use the **EF CLI Tools**

**dotnet ef migrations add {MigrationName}**

* + Use the **Package Manager Console**

**Add-Migration {MigrationName}**

* To **undo a migration**, use one of the two ways

dotnet ef migrations remove {MigrationName}

Remove-Migration

* **Commit changes** to the database

**dotnet ef database update**

Removes the last migration



**Migrations in EF Core**

**Update-Database**

**db.Database.Migrate(); // Auto migrate at start**

**Migrates any DB**

**changes on startup**



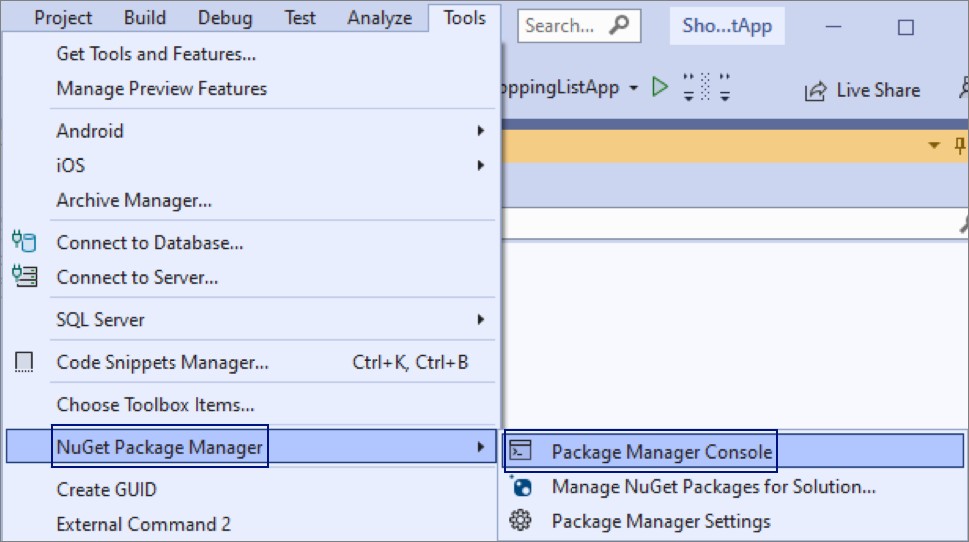
**Migrate the "ShoppingListDemo" App (1)**

**Microsoft.EntityFrameworkCore.Tools**

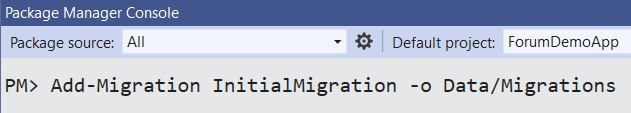
* Install the

package

* + Open the **Package Manager Console**



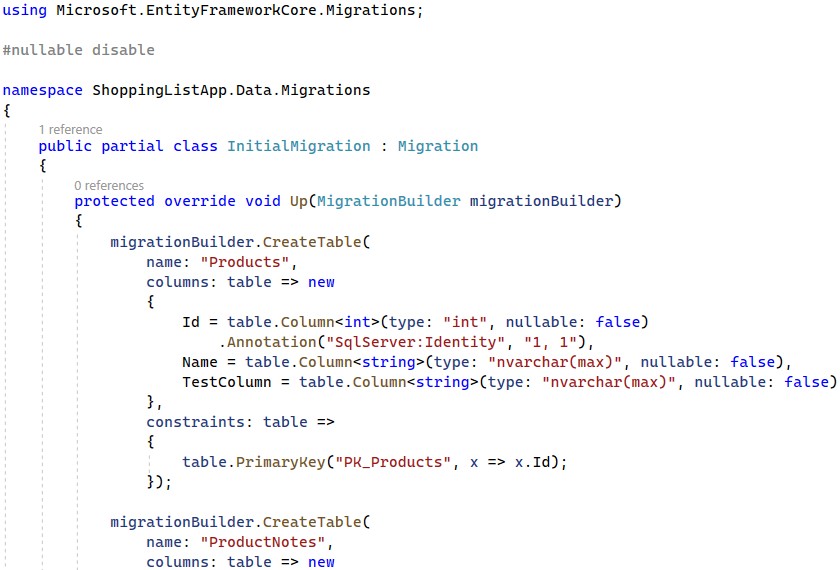
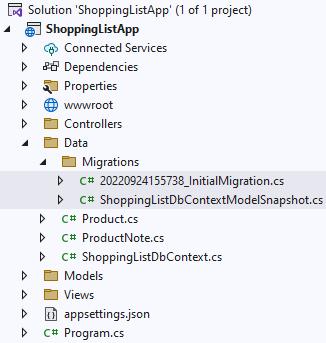
* + Create a **migration**



**Migration will be created in the "Data/Migrations" folder**



**Migrate the "ShoppingListDemo" App (2)**



**Auto Run Migration Scripts at Startup**

public class ShoppingListDbContext : DbContext

**{**

**public ShoppingListDbContext( DbContextOptions<ShoppingListDbContext> options) :**

**base(options)**

**=> Database.Migrate();**

**}**

* + This will **apply the migration scripts** (if not yet applied)
  + Simple, but **can cause problems**  not recommended in production
  + Recommended approach: migrate the database by hand

**dotnet ef database update**



**Summary**

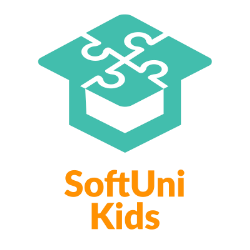
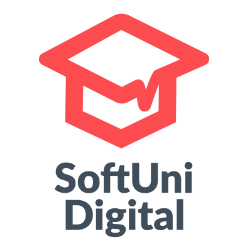
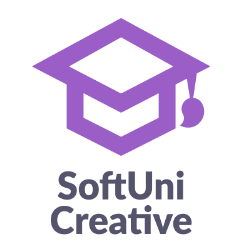
* + - **EF Core** maps database objects to database schema



* +  **Code First approach** creates a database,

based on C# classes that we create

* + - LINQ can be used to **query the DB** through the DB context
    - **Database migrating** updates the database schema to match app data models





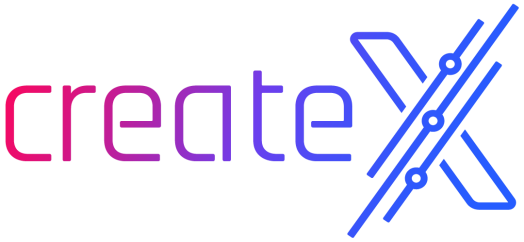
# A picture containing text, sign, vector graphics Description automatically generatedQuestions?

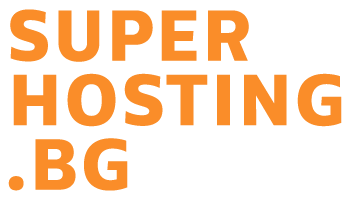
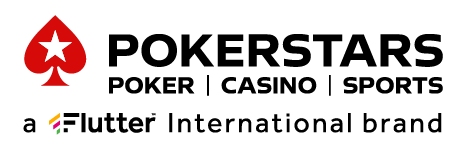


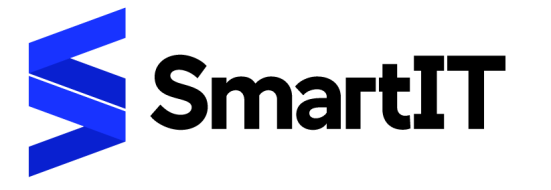
[](https://indeavr.com/)



**SoftUni Diamond Partners**



[](https://en.superhosting.bg/) [](https://bg.coca-colahellenic.com/bg/working-with-us) [](https://www.pokerstars.bg/)

[](https://smartit.bg/)



**Educational Partners**



**Trainings @ Software University (SoftUni)**

* [](https://softuni.org/)Software University – High-Quality Education, Profession and Job for Software Developers
  + [softuni.bg](https://softuni.bg/), [about.softuni.bg](https://about.softuni.bg/)
* Software University Foundation
  + [](https://www.facebook.com/SoftwareUniversity)[softuni.foundation](https://softuni.foundation/)
* Software University @ Facebook
  + [](https://forum.softuni.bg/)[facebook.com/SoftwareUniversity](https://www.facebook.com/SoftwareUniversity)
* Software University Forums
  + [forum.softuni.bg](https://forum.softuni.bg/)



**License**

###### 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](https://softuni.bg/)