Contents

[ORM 2](#_Toc146563218)

[Entity Framework Core 6.x 2](#_Toc146563219)

[Course Link 2](#_Toc146563220)

[Nuget Packages 3](#_Toc146563221)

[In Data Access 3](#_Toc146563222)

[In Web tier 3](#_Toc146563223)

[Make Migrations 3](#_Toc146563224)

[Primary Key Set 3](#_Toc146563225)

[Required properties 3](#_Toc146563226)

[Snapshot File 3](#_Toc146563227)

[Remove Migration or Update existing Table. 3](#_Toc146563228)

[Remove a table 3](#_Toc146563229)

[Roll back to old migration 3](#_Toc146563230)

[More Commands 4](#_Toc146563231)

[Seed Data 4](#_Toc146563232)

[Data annotations 4](#_Toc146563233)

[EF Core Power Tools 4](#_Toc146563234)

[Relations in EF Core 6](#_Toc146563235)

[One to One 6](#_Toc146563236)

[One To Many 6](#_Toc146563237)

[Many to Many automatic 7](#_Toc146563238)

[Many to Many creating intermediate table. 7](#_Toc146563239)

# ORM

Object relational mapper. It is a technique that lets you query and manipulate data from database using an object-oriented-programing paradigm.

# Entity Framework Core 6.x

It is the new version of entity framework. It helps us to bring models from database and vice versa.

# Course Link

<https://dotnetmastery.com/Home/Details?courseId=11>

<https://github.com/bhrugen/CodingWiki_EF>

# Nuget Packages

## In Data Access

Microsoft.EntityFrameworkCore.SqlServer, this one include Microsoft.EntityFrameworkCore (the main one)

Microsoft.EntityFrameworkCore.Tools, this include Microsoft.EntityFrameworkCore.Desing

## In Web tier

Microsoft.EntityFrameworkCore.Desing

# Make Migrations

Make sure of **start project** and **default project.**

1. Set dataAccess start project
2. Command “**add-migration**” + a tag name example “CreatingBookTable”
3. Command “**Update-Database**” to create all object in the migration

## Primary Key Set

1. If the table only has one integer property with name ID
2. with [Key]
3. If the table only has one integer property THAT ENDS IN ID

## Required properties

If we do not put a property nullable It will assume that property as required

## Snapshot File

It has information about all migrations made.

# Remove Migration or Update existing Table.

It is recommend for non-updated migrations “**remove-migration**”

## Remove a table

Remove the dataset from dbcontext

## Roll back to old migration

Using “Update-Database” + the migration .cs file without date part of the name.

example “**Update-Database creatingBookTable**”

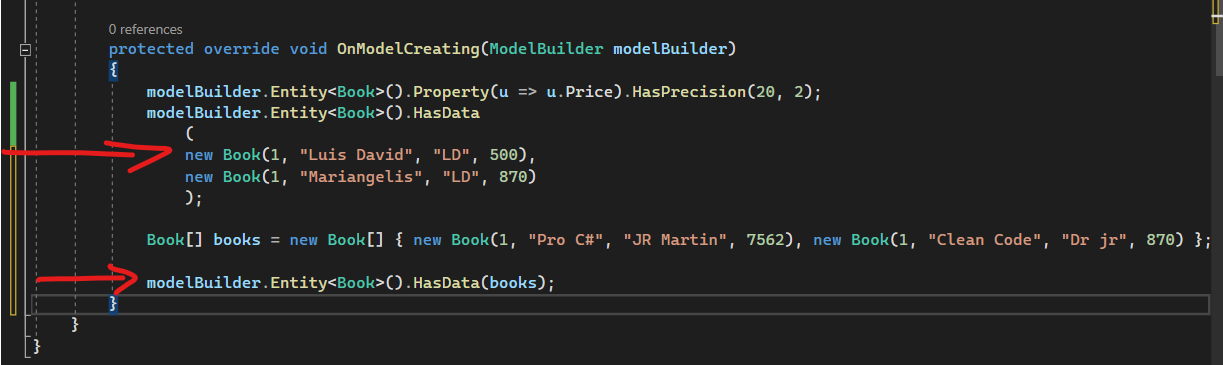
Then Just type “**Update-Database**” to applicate all migrations to database

# More Commands

1. **Get-migration** show up all migrations in console
2. **Drop-database** drop the database that is in connection string

# Seed Data

Create data and attach it to .HasData(), then add-migration

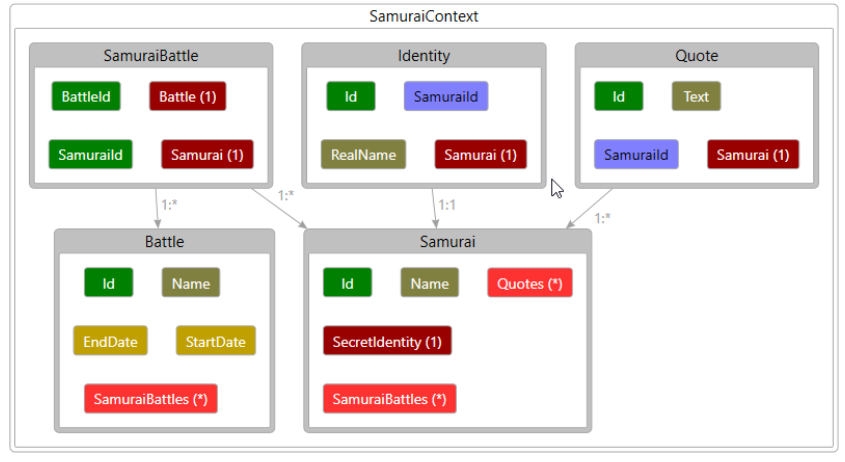


# Data annotations

1. Table name **[Table(“tb\_Category”)]**
2. Column name **[Column(“tb\_Category”)]**
3. **[Required]** put some column required.
4. **[Key]** put primary key.
5. **[MaxLength(50)]** varchar(50)
6. **[NotMapped]** It property is not added as new column
7. **[ForeignKey(“PropertyName from base foreign key table”)]**

# EF Core Power Tools

To create Table diagram in visual studio

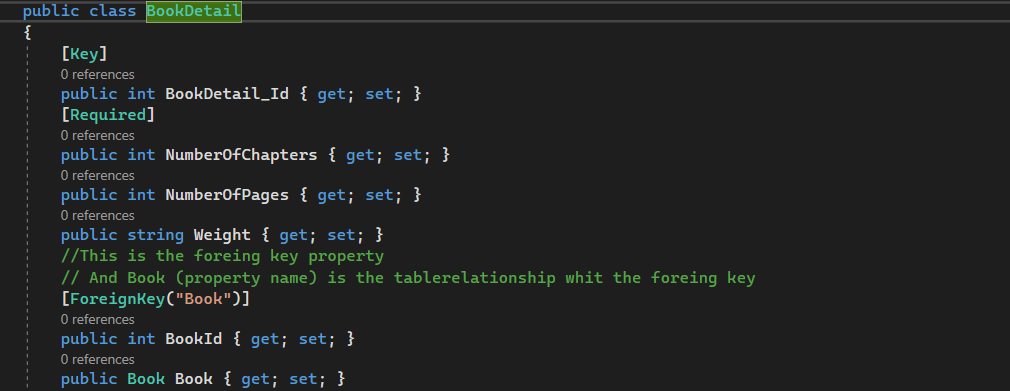


# Relations in EF Core

## One to One

Example BookDetail has a foreign key to Book.

1. Create a [ForeignKey(“Book”)] Book is the property name from the base table
2. Create property foreign key
3. Create property table base of foreign key
4. Create BookDetail property in Book



A screen shot of a computer

Description automatically generated

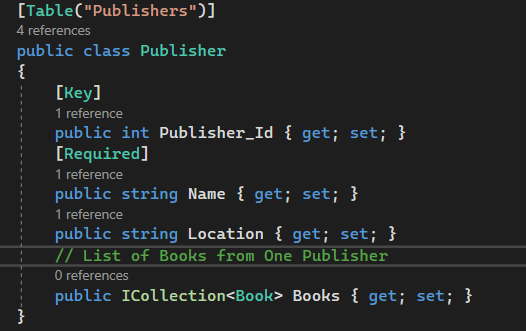
## One To Many

One **publisher** can have many **books**.

1. **Publisher\_Id** and Publisher **navigation** property in Book
2. **List<Book>** in Publisher, because One publisher can have many books

A screen shot of a computer code

Description automatically generated



## Many to Many automatic

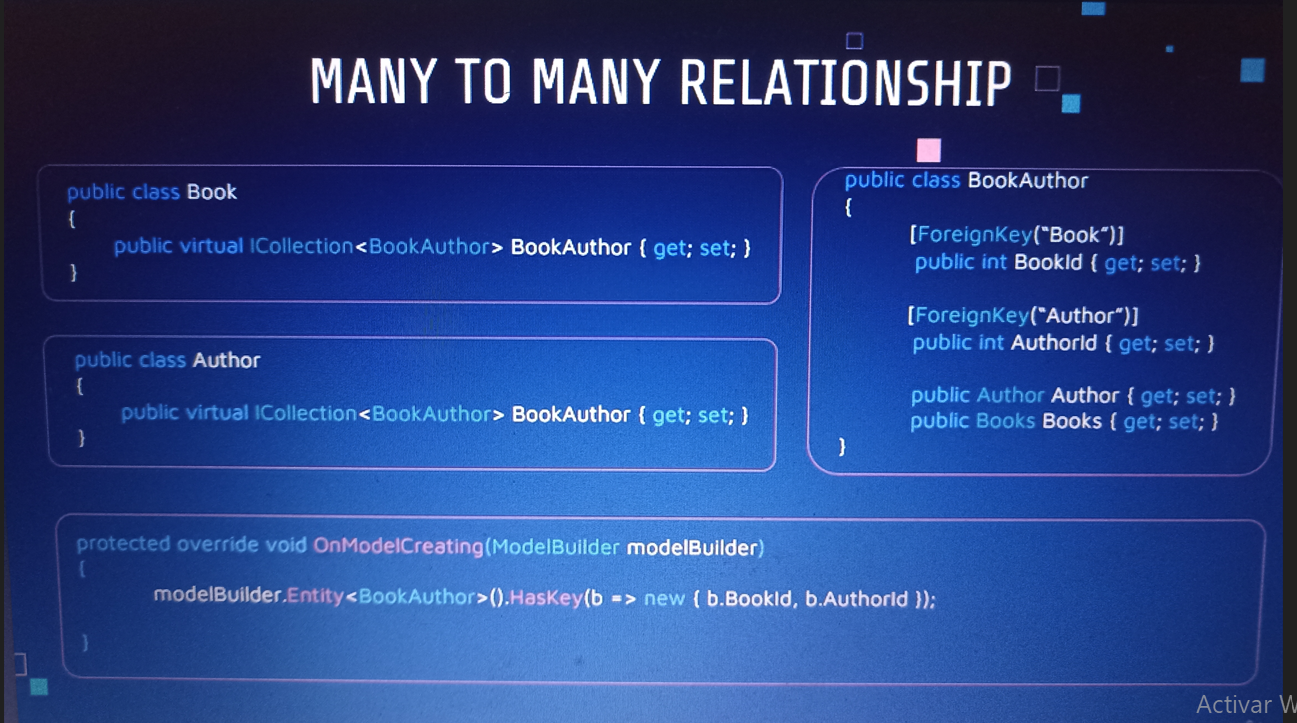
One **Book** can have multiples **Authors**. One Author can have multiples Books.

1. List of Book in Author
2. List of Author in Book

EF Core 6.x Create an **intermediate table** itself.

## Many to Many creating intermediate table.

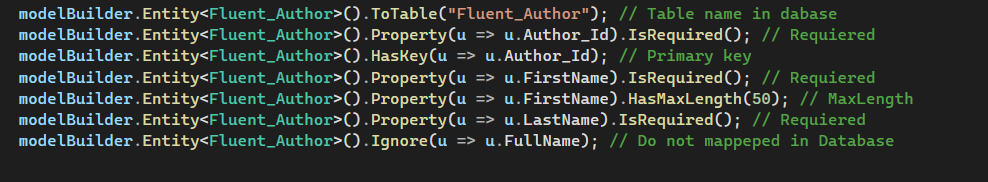
1. Create BookAuthor Table with bookId and authorId. And Author and Book references
2. Property virtual Icollection<BookAunthor> in Book
3. Property virtual Icollection<BookAunthor> in Author
4. Compose primary key in OnModelCreating



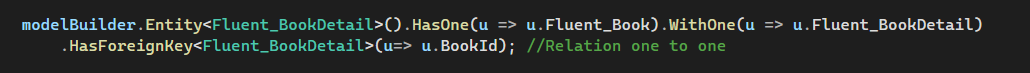
# Fluent API

Fluent API is an advanced way of specifying model configuration that covers everything that data annotation can do in addition to some advanced configuration which are not possible with data annotation. It is in **OnModelCreating**.

## Common Commands



## One to One Relationship



Fluent\_BookDetail

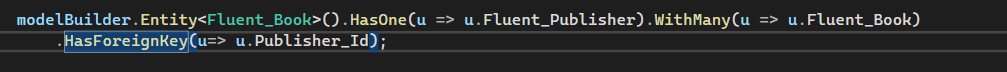
A black background with white text

Description automatically generated

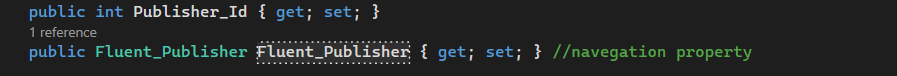
Fluent\_Book



## One To Many Relation



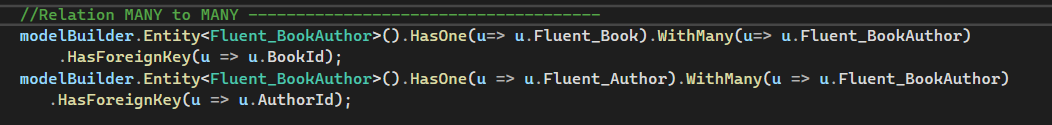
Fluent\_Book



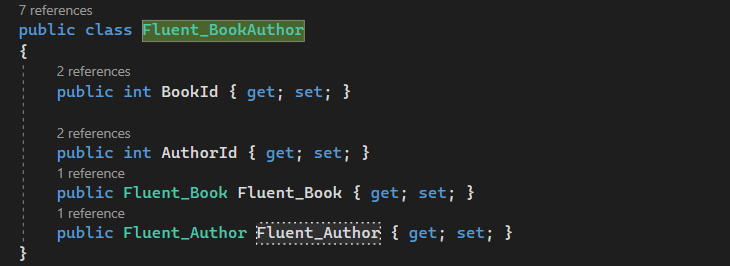
Fluent\_Publisher



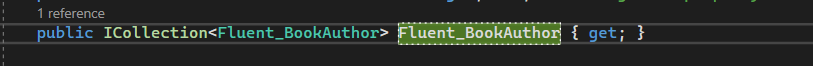
## Many To Many Relation



Fluent\_BookAuthor



Fluent\_Book



Fluent\_Author

