Contents

[ORM 3](#_Toc147695726)

[Entity Framework Core 6.x 3](#_Toc147695727)

[Course Link 3](#_Toc147695728)

[Nuget Packages 3](#_Toc147695729)

[In Data Access 3](#_Toc147695730)

[In Web tier 3](#_Toc147695731)

[Make Migrations 4](#_Toc147695732)

[Primary Key Set 4](#_Toc147695733)

[Required properties 4](#_Toc147695734)

[Snapshot File 4](#_Toc147695735)

[Remove Migration or Update existing Table. 4](#_Toc147695736)

[Remove a table 4](#_Toc147695737)

[Roll back to old migration 4](#_Toc147695738)

[More Commands 4](#_Toc147695739)

[Seed Data 4](#_Toc147695740)

[Data annotations 5](#_Toc147695741)

[EF Core Power Tools 5](#_Toc147695742)

[Relations in EF Core 6](#_Toc147695743)

[One to One 6](#_Toc147695744)

[One To Many 7](#_Toc147695745)

[Many to Many automatic 8](#_Toc147695746)

[Many to Many creating intermediate table. 8](#_Toc147695747)

[Fluent API 8](#_Toc147695748)

[Common Commands 9](#_Toc147695749)

[One to One Relationship 9](#_Toc147695750)

[One To Many Relation 9](#_Toc147695751)

[Many To Many Relation 10](#_Toc147695752)

[Organizing Fluent API 11](#_Toc147695753)

[ApplicationDbContext utilities 11](#_Toc147695754)

[Database.EnsureCreated() 11](#_Toc147695755)

[Database.GetPendingMigrations() 11](#_Toc147695756)

[UseSqlServer(sqlConnectionString).LogTo(path) 12](#_Toc147695757)

[Link Single(u=> u.Id == id) 12](#_Toc147695758)

[EF.Functions 12](#_Toc147695759)

[Pagination query sql server 12](#_Toc147695760)

[Configure DbContext in Web Application 12](#_Toc147695761)

[Create seed data with add migration. 13](#_Toc147695762)

[Sql command executed log with Web application. 13](#_Toc147695763)

[Projections in EF Core (Convert data types) 14](#_Toc147695764)

[Load Complex object 14](#_Toc147695765)

[Eager loading 14](#_Toc147695766)

[Deferred execution (When sql queries are executed) 14](#_Toc147695767)

[IEnumerable vs IQueryable in EF core 15](#_Toc147695768)

[Update vs Attach 15](#_Toc147695769)

# 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**”

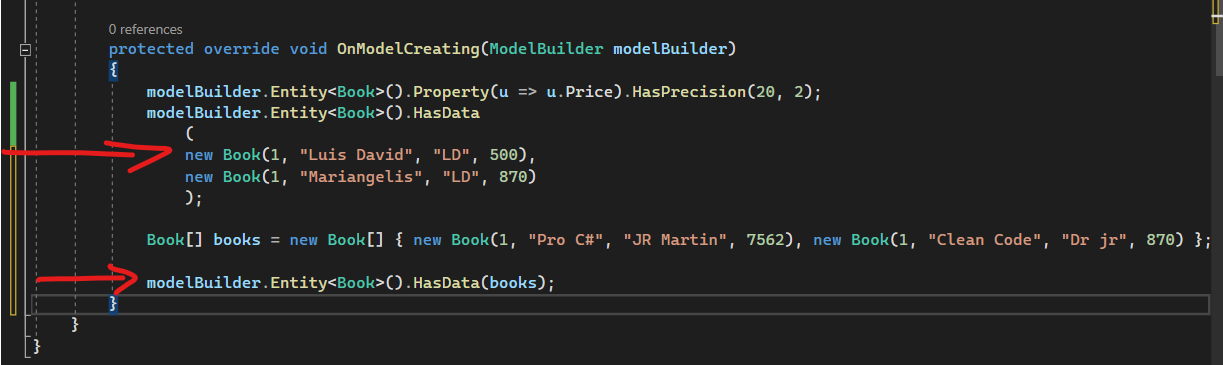
Then Just type “**Update-Database**” to applicate all migrations to database

# 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

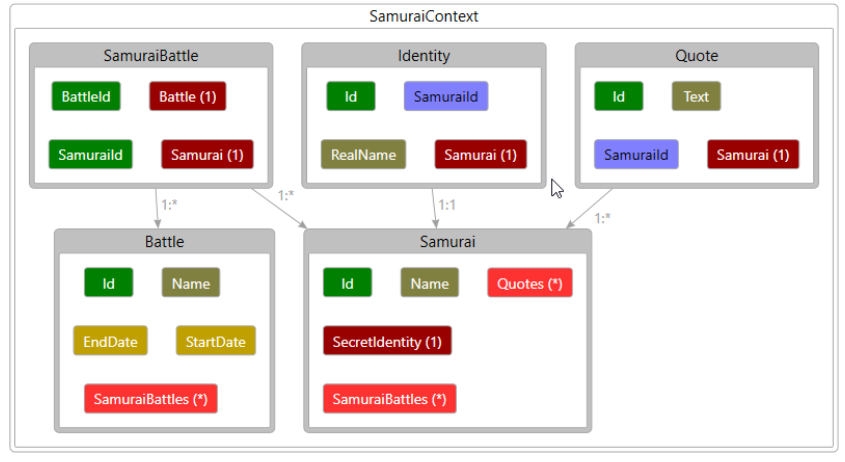


# Data annotations

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

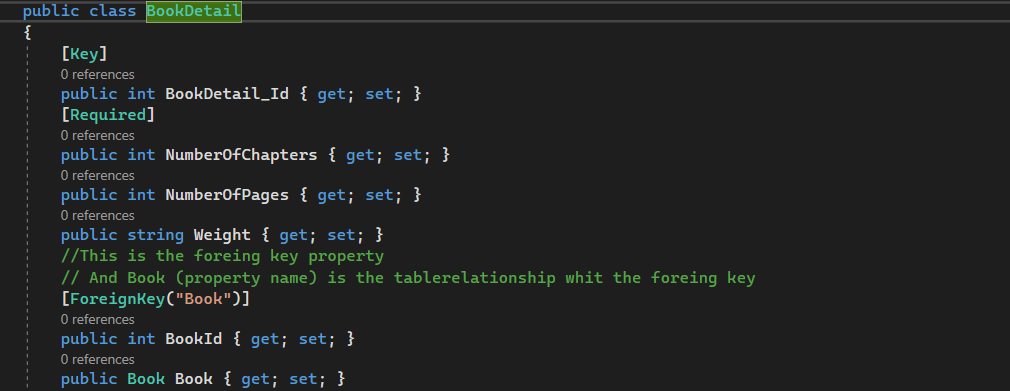


# Relations in EF Core

## 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

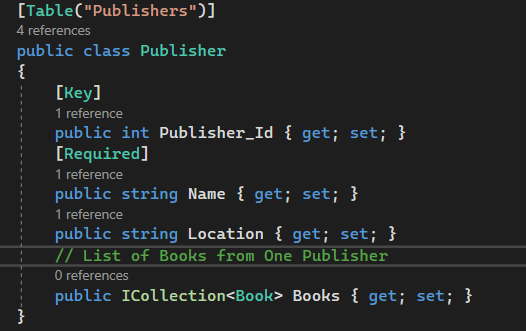
One **publisher** can have many **books**.

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

**Book Entity**

A screen shot of a computer code

Description automatically generated



## Many to Many automatic

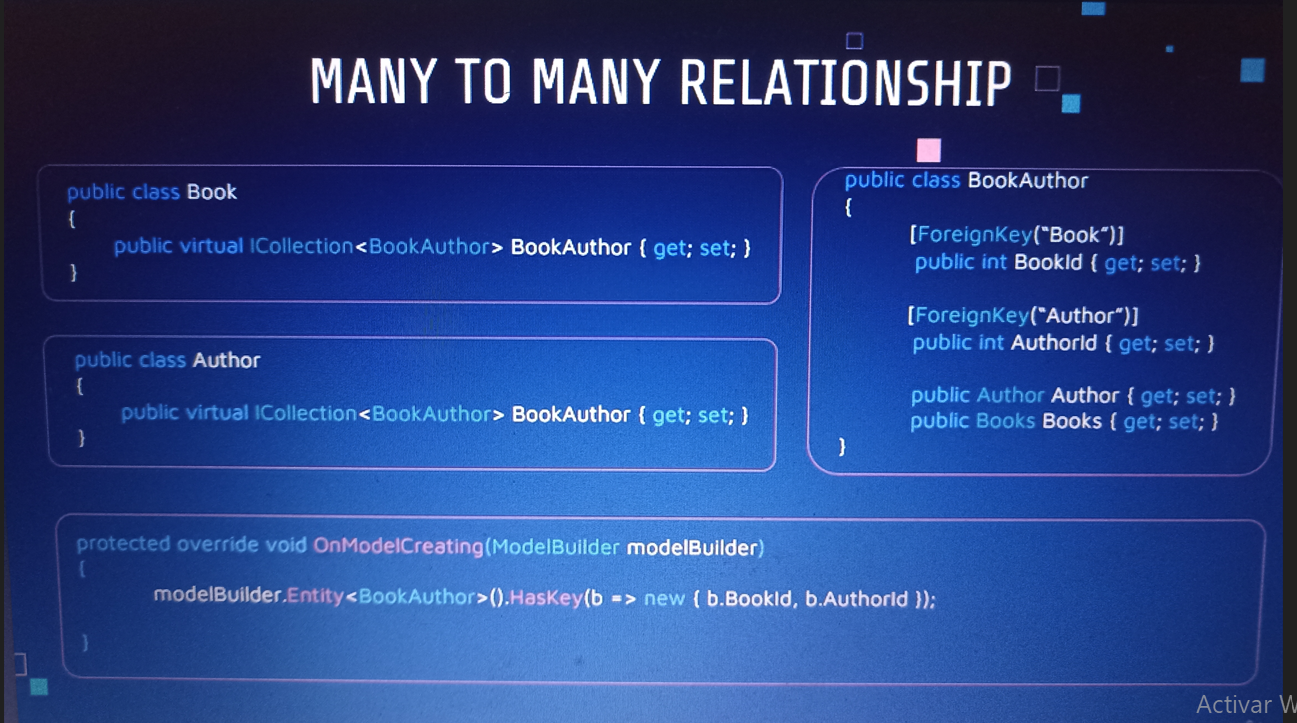
One **Book** can have multiples **Authors**. One Author can have multiples Books.

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.

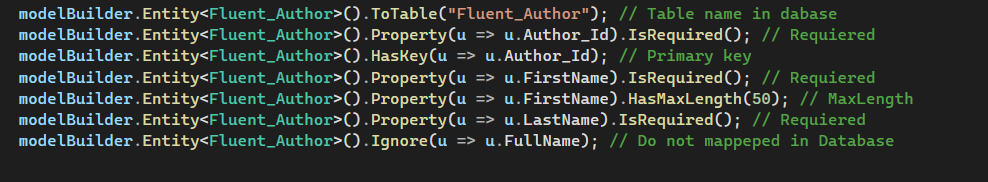
1. Create BookAuthor Table with bookId and authorId. And Author and Book references
2. Property virtual Icollection<BookAunthor> in Book
3. Property virtual Icollection<BookAunthor> in Author
4. Compose primary key in OnModelCreating



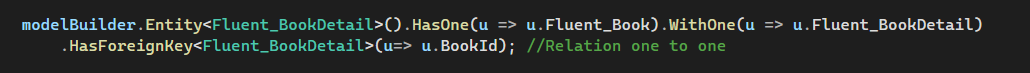
# Fluent API

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

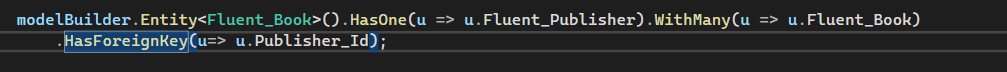
A black background with white text

Description automatically generated

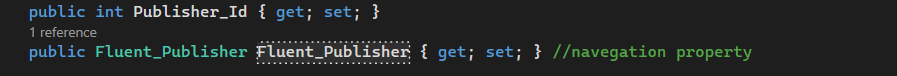
Fluent\_Book



## One To Many Relation



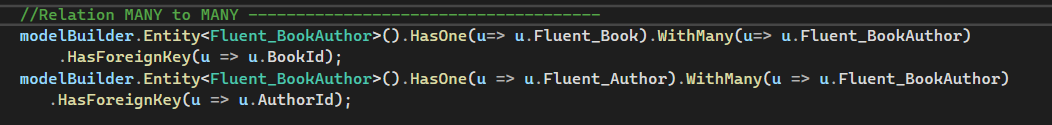
Fluent\_Book



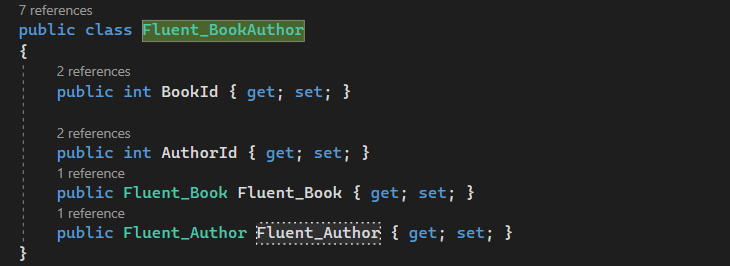
Fluent\_Publisher



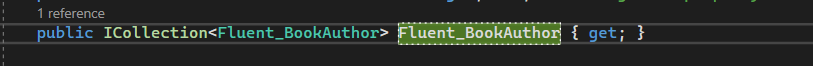
## Many To Many Relation



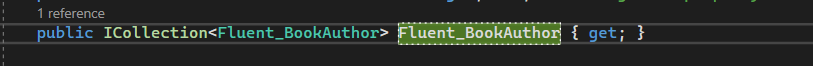
Fluent\_BookAuthor



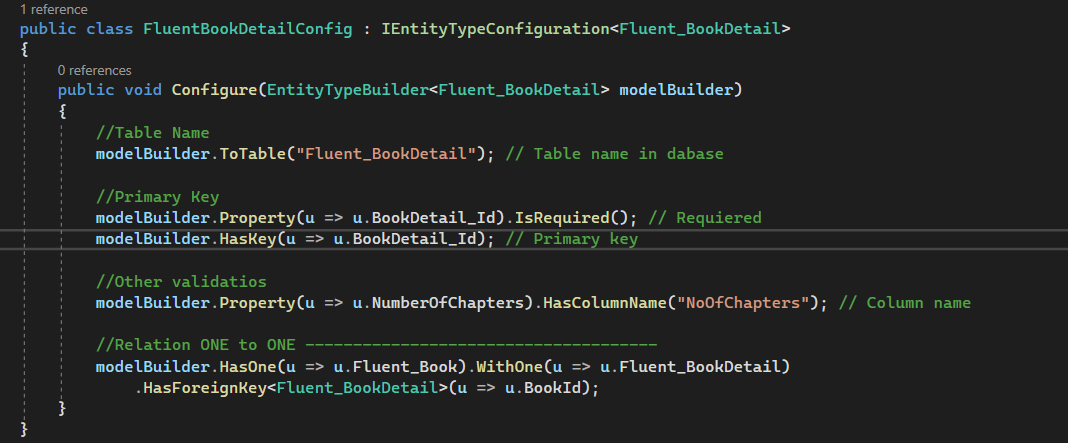
Fluent\_Book



Fluent\_Author



## Organizing Fluent API



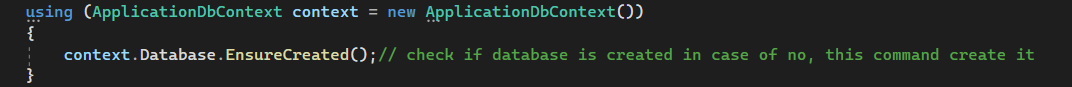
A black screen with colorful text

Description automatically generated

# ApplicationDbContext utilities

Database.EnsureCreated()

check if database is created in case of no, this command creates it



## Database.GetPendingMigrations()

Look for pending migrations

A black background with white text

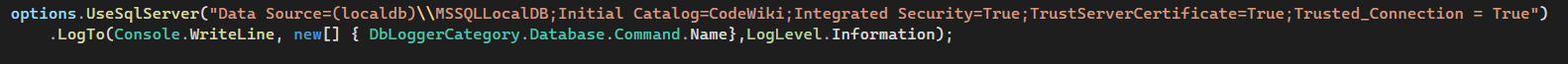
Description automatically generated

## UseSqlServer(sqlConnectionString).LogTo(path)

Show the sql commands executed by each one ApplicationDbContext action



Additional configuration for getting information about how is ef retrieving data



## Link Single(u=> u.Id == id)

If there is more than one throw an exception.

## EF.Functions

Like or another database functions

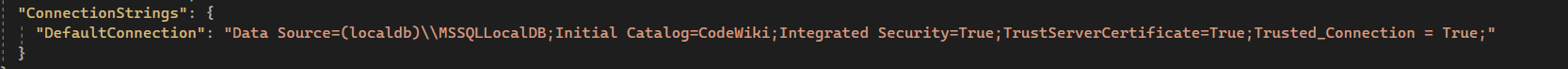
# Pagination query sql server

A close-up of a computer code

Description automatically generated

# Configure DbContext in Web Application

**Step 1:**  set connection string in application.json in **ConnectionStrings**  section



**Step 2:**  In program.cs

A screen shot of a computer code

Description automatically generated

**Step 3:** In application Db Context

A black background with blue text

Description automatically generated

# Create seed data with add migration.

**Step 1**: add migration empty.

A screen shot of a computer

Description automatically generated

**Step 2**: Create data manually.

A screen shot of a computer program

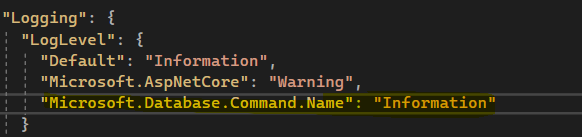
Description automatically generated

**Step 3**: update database

**A black background with white text

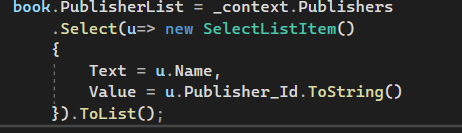
Description automatically generated**

# Sql command executed log with Web application.



# Projections in EF Core (Convert data types)

Converting data



# Load Complex object

A screenshot of a computer program

Description automatically generated

# Eager loading

Populate all the sub objects in a model using same sql statement.

Reserve word **Include** to include sub objects.

****

# Deferred execution (When sql queries are executed)

Sql command are executed just when the code is usen it. Or when you put a ToList(), or access to object properties.

# IEnumerable vs IQueryable in EF core

**IEnumerable**: Return all records and then apply the filter.



**IQueryable**: excute the sql command with the where all time



# Update vs Attach

**Update**: Update the complete object

**Attach**: Just update the properties that have been changed.