Prerequisites

1. Create a WebAPI project
2. Install the CLI tool - **dotnet-ef**
   https://learn.microsoft.com/pl-pl/ef/core/cli/dotnet
3. Install Nuget packages:
   Microsoft.EntityFrameworkCore.Design
   Microsoft.EntityFrameworkCore.SqlServer

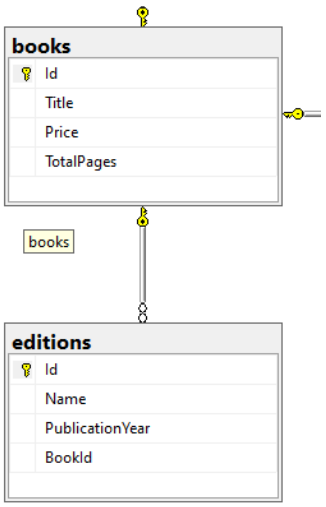
Codefirst

Add ConnectionString to the **appsettings.json** file. Example contents of this file:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Default": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=codefirst;Integrated Security=True;"
  }
}
```

At first, create models in OOP based on the relational diagram.

Example relation one to many:



Models should be in separate files.

```
public class Book
{
    public int Id { get; set; }
    public string Title { get; set; } = string.Empty;
    public double Price { get; set; }
    public int TotalPages { get; set; }

    public ICollection<Edition> Editions { get; set; } = new
HashSet<Edition>();
}

public class Edition
{
    public int Id { get; set; }
    public string Name { get; set; } = string.Empty;
    public int PublicationYear { get; set; }
    public int BookId { get; set; }

    public Book Book { get; set; } = null!;
}
```

Create database context. This is a class that inherits from DbContext.

```
public class ApplicationContext : DbContext
{

}
```

Generate constructors to be used in dependency injection. In Rider you can use shortcut Alt+Insert.

```
protected ApplicationContext()
{
}

public ApplicationContext(DbContextOptions options) : base(options)
{
}
```

Add DbSets that represent tables in database.

```
public DbSet<Author> Authors { get; set; }
public DbSet<Book> Books { get; set; }
public DbSet<Edition> Editions { get; set; }
public DbSet<Award> Awards { get; set; }
public DbSet<BookAward> BookAwards { get; set; }
```

Then add model compatibility with the database, such as primary keys, constraints, etc. For that you can choose two methods:

- Annotations
- Entity definition through OnModelCreating

The first option is faster, but the second is used more often in production.

```
[Table("books")]
public class Book
{
    [Key]
    public int Id { get; set; }
    [MaxLength(100)]
    public string Title { get; set; } = string.Empty;
    [Precision(3)]
    public double Price { get; set; }
    public int TotalPages { get; set; }
```

```
    public ICollection<Author> Authors { get; set; } = new HashSet<Author>();

    public ICollection<Edition> Editions { get; set; } = new
HashSet<Edition>();

    public ICollection<BookAward> BookAwards { get; set; } = new
HashSet<BookAward>();
}
```

[Key] – creates Primary Key for that property

[MaxLength(100)] – defines length constraint

[Table("books")] – sets table name

[ForeignKey(nameof(BookId))] – sets foreign key for defined connection

To create a compound key use the [PrimaryKey] annotation. You cannot use two [Key] annotations for that.

```
[Table("book_awards")]
[PrimaryKey(nameof(BookId), nameof(AwardId))]
public class BookAward
{
    public int Year { get; set; }
    public int BookId { get; set; }
    public int AwardId { get; set; }

    [ForeignKey(nameof(BookId))]
    public Book Book { get; set; } = null!;
    [ForeignKey(nameof(AwardId))]
    public Award Award { get; set; } = null!;
}
```

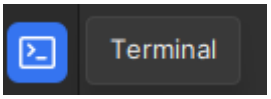
To use an entity definition, in context override OnModelCreating method. Using modelbuilder and extension methods set all entity properties.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Author>(e =>
    {
        e.ToTable("authors");

        e.HasKey(e => e.Id);
        e.Property(e => e.FirstName).HasMaxLength(100);
        e.Property(e => e.LastName).HasMaxLength(100);

        e.HasData(new List<Author>()
        {
            new() { Id = 1, FirstName = "John", LastName = "Doe"},
            new() { Id = 2, FirstName = "Ann", LastName = "Smith"},
            new() { Id = 3, FirstName = "Jack", LastName = "Taylor"}
        });
    });
}
```

To seed startup data to the database you have to do it in the method OnModelCreating.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<Author>().HasData(new List<Author>()
    {
        new() { Id = 1, FirstName = "John", LastName = "Doe"},
        new() { Id = 2, FirstName = "Ann", LastName = "Smith"},
        new() { Id = 3, FirstName = "Jack", LastName = "Taylor"}
    });
}
```

After whole setup, start the terminal in Rider. Below is a sample terminal icon:



Using the command line interface, change the directory to the folder with the project. By default, the terminal will start in the solution directory, so go to the project by executing such a command:

```
cd NameOfTheProject
```

At first add migrations using this command:

```
dotnet ef migrations add Init
```

It'll create migrations directory with migration class that Entity Framework will execute later.

To create tables in the database run this command:

```
dotnet ef database update
```