# Exercises: Entity Relations

This document defines the **exercise assignments** for the ["Databases Advanced – EF Core" course @ Software University](https://softuni.bg/trainings/3221/entity-framework-core-february-2021).

<https://www.youtube.com/watch?v=tZg2kz4d4qs&t=1s>

///

More material from MS about EF Core Relationships :

<https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key>

Fluent Api for P;roperties !!! too

## Student System

Your task is to create a database for the **Student System**, using the **EF Core Code First** approach. It should look like this:



### Constraints

Your **namespaces** should be:

* P01\_StudentSystem – for your Startup class, if you have one
* P01\_StudentSystem.Data – for your DbContext
* P01\_StudentSystem.Data.Models – for your models

Your **models** should be:

* StudentSystemContext – your DbContext
* Student:
  + StudentId
  + Name (up to 100 characters, unicode)
  + PhoneNumber (exactly 10 characters, not unicode, not required)
  + RegisteredOn
  + Birthday (not required)
* Course:
  + CourseId
  + Name (up to 80 characters, unicode)
  + Description (unicode, not required)
  + StartDate
  + EndDate
  + Price
* Resource:
  + ResourceId
  + Name (up to 50 characters, unicode)
  + Url (not unicode)
  + ResourceType (enum – can be Video, Presentation, Document or Other)
  + CourseId
* Homework:
  + HomeworkId
  + Content (string, linking to a file, not unicode)
  + ContentType (enum – can be Application, Pdf or Zip)
  + SubmissionTime
  + StudentId
  + CourseId
* StudentCourse – mapping class between **Students** and **Courses**

Table relations:

* **One student** can have **many CourseEnrollments**
* **One student** canhave **many HomeworkSubmissions**
* **One course** can have **many StudentsEnrolled**
* **One course** can have **many Resources**
* **One course** can have **many HomeworkSubmissions**

You will need a constructor, accepting **DbContextOptions** to test your solution in **Judge**!

Go to Solution File right click Open PowerShell Window

dotnet ef migrations add AddDescriptionCourse

dotnet ef database update /// -> this is DB migration -> change and actualization of the DataBase

<https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>

migrations help us so we can do a revert to the changes of the database this is the most useful of the migtations.

1:36: ///

exec sp\_changedbowner 'sa'

Fluent API is more powerful instrument for DB setting than property attributes. But more code I needed to write !

dotnet ef database update <previous-migration-name> -> You c hose whick migration snapshot of the DB to return

<https://eee.bg/>

Into the Solution File P01\_StudentSystem

We create Folder : Data, Where we put DB Context file and the folder Models, where we put our Models –class files .

Context File :

using Microsoft.EntityFrameworkCore;

using P01\_StudentSystem.Data.Models;

namespace P01\_StudentSystem.Data

{

public class StudentSystemContext : DbContext

{

public StudentSystemContext()

{

}

public StudentSystemContext(DbContextOptions options)

: base(options)

{

}

public DbSet<Course> Courses { get; set; }

public DbSet<Homework> HomeworksSubmissions { get; set; }

public DbSet<Resource> Resources { get; set; }

public DbSet<Student> Students { get; set; }

public DbSet<StudentCourse> StudentCourses { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Server=.;Database=StudentSystem;Integrated Security=true");

}

base.OnConfiguring(optionsBuilder);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<StudentCourse>(x =>

{

x.HasKey(x => new { x.CourseId, x.StudentId });

});

base.OnModelCreating(modelBuilder);

}

}

}

/////////////

Model Folder :

namespace P01\_StudentSystem.Data.Models

{

public enum ContentType

{

Application = 10,

Pdf = 20,

Zip = 30

}

}

////

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace P01\_StudentSystem.Data.Models

{

public class Course

{

public Course()

{

this.HomeworkSubmissions = new HashSet<Homework>();

this.Recources = new HashSet<Resource>();

this.StudentsEnrolled = new HashSet<StudentCourse>();

}

public int CourseId { get; set; }

[Required]

[MaxLength(80)]

public string Name { get; set; }

public string Description { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

//[Required]

// [Column(TypeName = "decimal(18,2))"]

public decimal Price { get; set; }

public ICollection<Homework> HomeworkSubmissions { get; set; }

public ICollection<Resource> Recources { get; set; }

public ICollection<StudentCourse> StudentsEnrolled { get; set; }

}

}

////

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Text;

namespace P01\_StudentSystem.Data.Models

{

public class Homework

{

public int HomeworkId { get; set; }

[Required]

[Column(TypeName = ("varchar(255)"))]

public string Content { get; set; }

public ContentType ContentType { get; set; }

public DateTime SubmissionTime { get; set; }

public int StudentId { get; set; }

public Student Student { get; set; }

public int CourseId { get; set; }

public Course Course { get; set; }

}

}

/////

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Text;

namespace P01\_StudentSystem.Data.Models

{

public class Resource

{

public int ResourceId { get; set; }

[Required]

[MaxLength(50)]

public string Name { get; set; }

[Required]

[Column(TypeName ="varchar(2048)")]

public string Url { get; set; }

public ResourceType ResourceType { get; set; }

public int CourseId { get; set; }

public Course Course { get; set; }

}

}

/////

namespace P01\_StudentSystem.Data.Models

{

public enum ResourceType

{

Video = 10,

Presentation = 20,

Document = 30,

Other = 40

}

}

/////

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Text;

namespace P01\_StudentSystem.Data.Models

{

public class Student

{

public Student()

{

this.HomeworkSubmissions = new HashSet<Homework>();

this.CourseEnrollments = new HashSet<StudentCourse>();

}

public int StudentId { get; set; }

[Required]

[MaxLength(100)]

public string Name { get; set; }

[Column(TypeName = "char(10)")] // [Column(TypeName = "char(10)")]

public string PhoneNumber { get; set; }

public DateTime RegisteredOn { get; set; } //by default DateTime is not null

public DateTime? Birthday { get; set; }

public ICollection<Homework> HomeworkSubmissions { get; set; }

public ICollection<StudentCourse> CourseEnrollments { get; set; }

}

}

////

namespace P01\_StudentSystem.Data.Models

{

public class StudentCourse

{

public int CourseId { get; set; }

public Course Course { get; set; }

public int StudentId { get; set; }

public Student Student { get; set; }

}

}

///

Our Program.Cs File into the Solution

using P01\_StudentSystem.Data;

using System;

namespace P01\_StudentSystem

{

public class Program

{

public static void Main(string[] args)

{

var context = new StudentSystemContext();

context.Database.EnsureDeleted();

context.Database.EnsureCreated();

}

}

}

/////

When we do Migrations using the Power Shell commands,

It is created intot he Solution Migrations Folder :

using System;

using Microsoft.EntityFrameworkCore.Migrations;

namespace P01\_StudentSystem.Migrations

{

public partial class InitialCreate : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Courses",

columns: table => new

{

CourseId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(80)", maxLength: 80, nullable: false),

StartDate = table.Column<DateTime>(type: "datetime2", nullable: false),

EndDate = table.Column<DateTime>(type: "datetime2", nullable: false),

Price = table.Column<decimal>(type: "decimal(18,2)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Courses", x => x.CourseId);

});

migrationBuilder.CreateTable(

name: "Students",

columns: table => new

{

StudentId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(100)", maxLength: 100, nullable: false),

PhoneNumber = table.Column<string>(type: "char(10)", nullable: true),

RegisteredOn = table.Column<DateTime>(type: "datetime2", nullable: false),

Birthday = table.Column<DateTime>(type: "datetime2", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Students", x => x.StudentId);

});

migrationBuilder.CreateTable(

name: "Resources",

columns: table => new

{

ResourceId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(50)", maxLength: 50, nullable: false),

Url = table.Column<string>(type: "varchar(2048)", nullable: false),

ResourceType = table.Column<int>(type: "int", nullable: false),

CourseId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Resources", x => x.ResourceId);

table.ForeignKey(

name: "FK\_Resources\_Courses\_CourseId",

column: x => x.CourseId,

principalTable: "Courses",

principalColumn: "CourseId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Homeworks",

columns: table => new

{

HomeworkId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Content = table.Column<string>(type: "varchar(255)", nullable: false),

ContentType = table.Column<int>(type: "int", nullable: false),

SubmissionTime = table.Column<DateTime>(type: "datetime2", nullable: false),

StudentId = table.Column<int>(type: "int", nullable: false),

CourseId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Homeworks", x => x.HomeworkId);

table.ForeignKey(

name: "FK\_Homeworks\_Courses\_CourseId",

column: x => x.CourseId,

principalTable: "Courses",

principalColumn: "CourseId",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Homeworks\_Students\_StudentId",

column: x => x.StudentId,

principalTable: "Students",

principalColumn: "StudentId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "StudentCourses",

columns: table => new

{

CourseId = table.Column<int>(type: "int", nullable: false),

StudentId = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_StudentCourses", x => new { x.CourseId, x.StudentId });

table.ForeignKey(

name: "FK\_StudentCourses\_Courses\_CourseId",

column: x => x.CourseId,

principalTable: "Courses",

principalColumn: "CourseId",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_StudentCourses\_Students\_StudentId",

column: x => x.StudentId,

principalTable: "Students",

principalColumn: "StudentId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Homeworks\_CourseId",

table: "Homeworks",

column: "CourseId");

migrationBuilder.CreateIndex(

name: "IX\_Homeworks\_StudentId",

table: "Homeworks",

column: "StudentId");

migrationBuilder.CreateIndex(

name: "IX\_Resources\_CourseId",

table: "Resources",

column: "CourseId");

migrationBuilder.CreateIndex(

name: "IX\_StudentCourses\_StudentId",

table: "StudentCourses",

column: "StudentId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Homeworks");

migrationBuilder.DropTable(

name: "Resources");

migrationBuilder.DropTable(

name: "StudentCourses");

migrationBuilder.DropTable(

name: "Courses");

migrationBuilder.DropTable(

name: "Students");

}

}

}

/////

using Microsoft.EntityFrameworkCore.Migrations;

namespace P01\_StudentSystem.Migrations

{

public partial class AddDescriptionColumnCourse : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

}

protected override void Down(MigrationBuilder migrationBuilder)

{

}

}

}

////

// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using P01\_StudentSystem.Data;

namespace P01\_StudentSystem.Migrations

{

[DbContext(typeof(StudentSystemContext))]

partial class StudentSystemContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("ProductVersion", "5.0.3")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Course", b =>

{

b.Property<int>("CourseId")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("EndDate")

.HasColumnType("datetime2");

b.Property<string>("Name")

.IsRequired()

.HasMaxLength(80)

.HasColumnType("nvarchar(80)");

b.Property<decimal>("Price")

.HasColumnType("decimal(18,2)");

b.Property<DateTime>("StartDate")

.HasColumnType("datetime2");

b.HasKey("CourseId");

b.ToTable("Courses");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Homework", b =>

{

b.Property<int>("HomeworkId")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Content")

.IsRequired()

.HasColumnType("varchar(255)");

b.Property<int>("ContentType")

.HasColumnType("int");

b.Property<int>("CourseId")

.HasColumnType("int");

b.Property<int>("StudentId")

.HasColumnType("int");

b.Property<DateTime>("SubmissionTime")

.HasColumnType("datetime2");

b.HasKey("HomeworkId");

b.HasIndex("CourseId");

b.HasIndex("StudentId");

b.ToTable("Homeworks");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Resource", b =>

{

b.Property<int>("ResourceId")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CourseId")

.HasColumnType("int");

b.Property<string>("Name")

.IsRequired()

.HasMaxLength(50)

.HasColumnType("nvarchar(50)");

b.Property<int>("ResourceType")

.HasColumnType("int");

b.Property<string>("Url")

.IsRequired()

.HasColumnType("varchar(2048)");

b.HasKey("ResourceId");

b.HasIndex("CourseId");

b.ToTable("Resources");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Student", b =>

{

b.Property<int>("StudentId")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime?>("Birthday")

.HasColumnType("datetime2");

b.Property<string>("Name")

.IsRequired()

.HasMaxLength(100)

.HasColumnType("nvarchar(100)");

b.Property<string>("PhoneNumber")

.HasColumnType("char(10)");

b.Property<DateTime>("RegisteredOn")

.HasColumnType("datetime2");

b.HasKey("StudentId");

b.ToTable("Students");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.StudentCourse", b =>

{

b.Property<int>("CourseId")

.HasColumnType("int");

b.Property<int>("StudentId")

.HasColumnType("int");

b.HasKey("CourseId", "StudentId");

b.HasIndex("StudentId");

b.ToTable("StudentCourses");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Homework", b =>

{

b.HasOne("P01\_StudentSystem.Data.Models.Course", "Course")

.WithMany("HomeworkSubmissions")

.HasForeignKey("CourseId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("P01\_StudentSystem.Data.Models.Student", "Student")

.WithMany("HomeworkSubmissions")

.HasForeignKey("StudentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Course");

b.Navigation("Student");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Resource", b =>

{

b.HasOne("P01\_StudentSystem.Data.Models.Course", "Course")

.WithMany("Recources")

.HasForeignKey("CourseId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Course");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.StudentCourse", b =>

{

b.HasOne("P01\_StudentSystem.Data.Models.Course", "Course")

.WithMany("StudentsEnrolled")

.HasForeignKey("CourseId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("P01\_StudentSystem.Data.Models.Student", "Student")

.WithMany("CourseEnrollments")

.HasForeignKey("StudentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Course");

b.Navigation("Student");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Course", b =>

{

b.Navigation("HomeworkSubmissions");

b.Navigation("Recources");

b.Navigation("StudentsEnrolled");

});

modelBuilder.Entity("P01\_StudentSystem.Data.Models.Student", b =>

{

b.Navigation("CourseEnrollments");

b.Navigation("HomeworkSubmissions");

});

#pragma warning restore 612, 618

}

}

}

////

## Football Betting

Your task is to create a database for a **Football Bookmaker System**, using the **Code First** approach. It should look like this:



### Constraints

Your **namespaces** should be:

* P03\_FootballBetting – for your Startup class, if you have one
* P03\_FootballBetting.Data – for your DbContext
* P03\_FootballBetting.Data.Models – for your models

Your models should be:

* **FootballBettingContext** – your DbContext (install the corresponding MS Nuget Packages !!!)
* **Team** – TeamId, Name, LogoUrl, Initials (JUV, LIV, ARS…), Budget, PrimaryKitColorId, SecondaryKitColorId, TownId
* **Color** – ColorId, Name /// 1:50;
* **Town** – TownId, Name, CountryId
* **Country** – CountryId, Name
* **Player** – PlayerId, Name, SquadNumber, TeamId, PositionId, IsInjured
* **Position** – PositionId, Name
* **PlayerStatistic** – GameId, PlayerId, ScoredGoals, Assists, MinutesPlayed
* **Game** – GameId, HomeTeamId, AwayTeamId, HomeTeamGoals, AwayTeamGoals, DateTime, HomeTeamBetRate, AwayTeamBetRate, DrawBetRate, Result)
* **Bet** – BetId, Amount, Prediction, DateTime, UserId, GameId
* **User** – UserId, Username, Password, Email, Name, Balance
* **// 2**:07 min;

Table relationships:

* **A Team** has one **PrimaryKitColor** and one **SecondaryKitColor**
* **A Color** has **many PrimaryKitTeams** and **many SecondaryKitTeams**
* **A Team** residents in one **Town**
* **A Town** can host **several** **Teams**
* **A Game** has one **HomeTeam** and one **AwayTeam** and a **Team** can have **many** **HomeGames** and **many** **AwayGames**
* **A Town** can be placed in **one** **Country** and a **Country** can have many **Towns**
* **A Player** can play for **one** **Team** and **one** **Team** can have many **Players**
* **A Player** can play at one **Position** and one **Position** can be played by **many** **Players**
* **One** **Player** can play in **many** **Games** and in each **Game**, **many** **Players** take part (both collections must be named PlayerStatistics)
* **Many** **Bets** can be placed on **one** **Game**, but **a** **Bet** can be only on **one** **Game**
* Each bet for given game must have **Prediction** result
* **A Bet** can be placed by only **one** **User** and one **User** can place many **Bets**

Separate the **models**, **data** and **client** into **different layers** (projects).

Go on Solution right click-> Add Project -> into the search write .net core library then choose,

A Project for crating class Library that targets net.Core - > net => give it a name : P03\_FootballBetting.Models

Move folder Models into this new project. Delete those files from old place-project so not to repeat them.

One again second time : Go on Solution right click-> Add Project -> into the search write .net core library then choose, A Project for crating class Library that targets net.Core - > net => give it a name : P03\_FootballBetting.Data

Move Context and DataFolder here, Delete from old Project those files too.

Move the net.Core SQL and Design packages to the P03\_FootballBetting.Data file (two clicks )-> See the video od Stoyan to better inderstand this how it happens.

///

Migrations into the Data Folder

We have the models -> folder Models (all our Classes in C# )

And separately our project that can call them !

See the Video again and rewite this Task because I am not sure that is 100 % true and runs normally

//

using System;

using System.Collections.Generic;

using System.Text;

namespace P03\_FootballBetting.Data.Models

{

public class Bet

{

public int BetId { get; set; }

public decimal Amount { get; set; }

//possible errro in Judge

public double Prediction { get; set; }

public DateTime DateTime { get; set; }

public int UserId { get; set; }

public User User { get; set; }

public int GameId { get; set; }

public Game Game { get; set; }

//ICollections Game

}

}

///

using System;

using System.Collections.Generic;

using System.Text;

namespace P03\_FootballBetting.Data.Models

{

public class Color

{

public int ColorId { get; set; }

public string Name { get; set; }

//ICollection Teams

public ICollection <Team>PrimaryKitTeams { get; set; }

public ICollection <Team>SecondaryKitTeams { get; set; }

}

}

////

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Country

{

public int CountryId { get; set; }

public string Name { get; set; }

//Icollection Towns

public ICollection<Town>Towns { get; set; }

}

}

////

using System;

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Game

{

public int GameId { get; set; }

public int HomeTeamId { get; set; }

public Team HomeTeam { get; set; }

public int AwayTeamId { get; set; }

public Team AwayTeam { get; set; }

public int AwayTeamGoals { get; set; }

public int HomeTeamGoals { get; set; }

public DateTime DateTime { get; set; }

public double HomeTeamBetRate { get; set; }

public double AwayTeamBetRate { get; set; }

public double DrawBetRate { get; set; }

// possible error in judge

public int Result { get; set; }

public ICollection<PlayerStatistic> PlayerStatistics { get; set; }

public ICollection<Bet> Bets { get; set; }

}

}

////

using System;

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Game

{

public int GameId { get; set; }

public int HomeTeamId { get; set; }

public Team HomeTeam { get; set; }

public int AwayTeamId { get; set; }

public Team AwayTeam { get; set; }

public int AwayTeamGoals { get; set; }

public int HomeTeamGoals { get; set; }

public DateTime DateTime { get; set; }

public double HomeTeamBetRate { get; set; }

public double AwayTeamBetRate { get; set; }

public double DrawBetRate { get; set; }

// possible error in judge

public int Result { get; set; }

public ICollection<PlayerStatistic> PlayerStatistics { get; set; }

public ICollection<Bet> Bets { get; set; }

}

}

///

using System;

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Game

{

public int GameId { get; set; }

public int HomeTeamId { get; set; }

public Team HomeTeam { get; set; }

public int AwayTeamId { get; set; }

public Team AwayTeam { get; set; }

public int AwayTeamGoals { get; set; }

public int HomeTeamGoals { get; set; }

public DateTime DateTime { get; set; }

public double HomeTeamBetRate { get; set; }

public double AwayTeamBetRate { get; set; }

public double DrawBetRate { get; set; }

// possible error in judge

public int Result { get; set; }

public ICollection<PlayerStatistic> PlayerStatistics { get; set; }

public ICollection<Bet> Bets { get; set; }

}

}

////

using System;

using System.Collections.Generic;

using System.Text;

namespace P03\_FootballBetting.Data.Models

{

public class PlayerStatistic

{

public int GameId { get; set; }

public Game Game { get; set; }

public int PlayerId { get; set; }

public Player Player { get; set; }

public int ScoredGoals { get; set; }

public int Assists { get; set; }

public int MinutesPlayed { get; set; }

}

}

///

using System;

using System.Collections.Generic;

using System.Text;

namespace P03\_FootballBetting.Data.Models

{

public class Player

{

public int PlayerId { get; set; }

public string Name { get; set; }

public int SquadNumber { get; set; }

public int TeamId { get; set; }

public Team Team { get; set; }

public int PositionId { get; set; }

public Position Position { get; set; }

public bool IsInjured { get; set; }

public ICollection <PlayerStatistic> PlayerStatistics{ get; set; }

public ICollection<Bet>Bets { get; set; }

}

}

///

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Position

{

public int PositionId { get; set; }

public string Name { get; set; }

//ICollection Players

public ICollection<Player> Players { get; set; }

}

}

////

using System;

using System.Collections.Generic;

using System.Text;

namespace P03\_FootballBetting.Data.Models

{

public class Team

{

public int TeamId { get; set; }

public string Name { get; set; }

public string LogoUrl { get; set; }

public string Initials { get; set; }

public decimal Budget { get; set; }

public int? PrimaryKitColorId { get; set; }

public Color PrimaryKitColor { get; set; }

public int? SecondaryKitColorId { get; set; }

public Color SecondaryKitColor { get; set; }

public int TownId { get; set; }

public Town Town { get; set; }

//ICollection Players

//ICollection Games

public ICollection<Player> Players { get; set; } // Stoyan documents Teams

public ICollection<Game> HomeGames { get; set; }

public ICollection<Game> AwayGames { get; set; }

}

}

///

using System.Collections.Generic;

namespace P03\_FootballBetting.Data.Models

{

public class Town

{

public int TownId { get; set; }

public string Name { get; set; }

public int CountryId { get; set; }

public Country Country { get; set; }

public ICollection<Team> Teams { get; set; }

}

}

///

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace P03\_FootballBetting.Data.Models

{

public class User

{

public int UserId { get; set; }

public string Username { get; set; }

public string Password { get; set; }

// [EmailAddress]

public string Email { get; set; }

public string Name { get; set; }

public decimal Balance { get; set; }

public ICollection<Bet> Bets { get; set; }

}

}

///

using Microsoft.EntityFrameworkCore;

using P03\_FootballBetting.Data.Models;

namespace P03\_FootballBetting.Data

{

class FootballBettingContext :DbContext

{

public FootballBettingContext()

{

}

public FootballBettingContext(DbContextOptions options)

: base(options)

{

}

public DbSet<Bet> Bets { get; set; }

public DbSet<Color> Colors { get; set; }

public DbSet<Country> Coiuntries { get; set; }

public DbSet<Game> Games { get; set; }

public DbSet<Player> Players { get; set; }

public DbSet<PlayerStatistic> PlayerStatistics { get; set; }

public DbSet<Position> Positions { get; set; }

public DbSet<Team> Teams { get; set; }

public DbSet<Town> Towns { get; set; }

public DbSet<User> Users { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Server=.;Database=Football;Integrated Security=True");

}

base.OnConfiguring(optionsBuilder);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Team>(x =>

{

x.HasOne(x => x.PrimaryKitColor)

.WithMany(x => x.PrimaryKitTeams)

.HasForeignKey(x => x.PrimaryKitColorId)

.OnDelete(DeleteBehavior.Restrict);

x.HasOne(x => x.SecondaryKitColor)

.WithMany(x => x.SecondaryKitTeams)

.HasForeignKey(x => x.SecondaryKitColorId)

.OnDelete(DeleteBehavior.Restrict);

//x.HasOne(x => x.Town) //one team have one town

//.WithMany(x => x.Teams) // the town has many teams

//.HasForeignKey(x => x.TownId); // into the table team - has FK townId

});

modelBuilder.Entity<Game>(x =>

{

x.HasOne(x => x.HomeTeam)

.WithMany(x => x.HomeGames)

.HasForeignKey(x => x.HomeTeamId);

x.HasOne(x => x.AwayTeam)

.WithMany(x => x.AwayGames)

.HasForeignKey(x => x.AwayTeamId);

});

modelBuilder.Entity<PlayerStatistic>(x =>

{

x.HasKey(x => new { x.PlayerId, x.GameId });

});

base.OnModelCreating(modelBuilder);

}

}

}

////

We have to add the foreign class Collections into each Class -> and to initialize the Constructor with each collection into the Real Application !