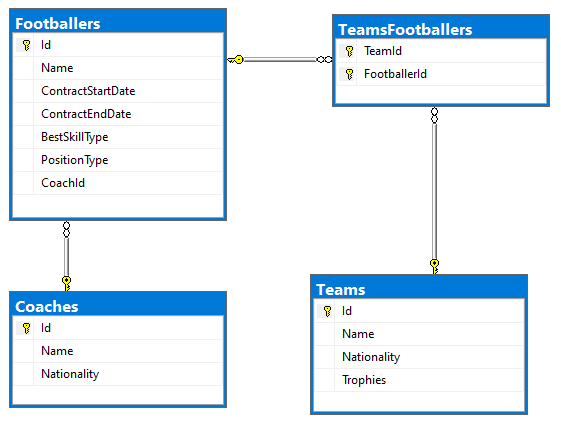
# Databases Advanced Exam - 06 August 2022

Exam problems for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/trainings/3709/entity-framework-core-june-2022). Submit your solutions in the **SoftUni Judge** system (delete all **bin**/**obj** and **packages** folders).

Your task is to create a **database application**, using **Entity Framework Core,** using the **Code First** approach. Design the **domain models** and **methods** for manipulating the data, as described below.

# Footballers



## Project Skeleton Overview

You are given a **project skeleton**, which includes the following folders:

1. Data – contains the FootballersContext class, Models folder, which contains the **entity classes** and the **Configuration** class with **connection string**
2. DataProcessor – contains the Serializer and Deserializerclasses, which are used for **importing** and **exporting** data
3. Datasets – contains the .json and .xml files for the import part
4. ImportResults – contains the **import** results you make in the Deserializer class
5. ExportResults – contains the **export** results you make in the Serializer class

## Model Definition (50 pts)

The application needs to store the following data:

### Footballer

* Id– integer, **Primary Key**
* Name– text with length **[2, 40]** (**required**)
* ContractStartDate– date and time (**required**)
* **ContractEndDate** – date and time (**required**)
* PositionType – enumeration of type PositionType, with possible values **(**Goalkeeper, Defender, Midfielder, Forward**)** (**required**)
* BestSkillType– enumeration of type BestSkillType, with possible values **(**Defence, Dribble, Pass, Shoot, Speed**)** (**required**)
* CoachId– integer, foreign key (required)
* Coach– Coach
* TeamsFootballers– collection of type TeamFootballer

**NOTE**: For the **enumeration** types, use the **default** associated values – **0**, **1**, **2**, … The **possible** values must be **ordered** as **listed in this document**.

### Team

* Id– integer, Primary Key
* Name– text with length **[3, 40]**. May contain **letters (lower and upper case)**, **digits, spaces, a dot sign ('.') and a dash ('-')**. (**required**)
* Nationality – **text** with length **[2, 40]** (**required**)
* Trophies– **integer** (**required**)
* TeamsFootballers– collection of type TeamFootballer

### Coach

* Id– integer, **Primary Key**
* Name– **text** with length **[2, 40]** (**required**)
* Nationality – **text (required)**
* Footballers– collection of type Footballer

### TeamFootballer

* TeamId– integer, Primary Key, foreign key (required)
* Team– Team
* FootballerId– integer, Primary Key, foreign key (required)
* Footballer – Footballer

## Data Import (25pts)

For the functionality of the application, you need to create several methods that manipulate the database. The **project skeleton** already provides you with these methods, inside the Deserializer class.

**NOTE:** Usage of DataTransferObjects and **AutoMapper** is **optional**. Should you choose to use **AutoMapper**, go to the **StartUp** class and uncomment the line below:

**Mapper.Initialize(config => config.AddProfile<FootballersProfile>());**

Use the provided **JSON** and **XML** files to populate the database with data. Import all the information from those files into the database.

You are **not allowed** to modify the provided **JSON** and **XML** files.

**If a record does not meet the requirements from the first section, print an error message:**

|  |
| --- |
| **Error message** |
| Invalid data! |

### XML Import

#### Import Coaches

Using the file **coaches.xml**, import the data from the file into the database. Print information about each imported object in the format described below.

##### Constraints

* If there are **any validation errors** for the **coach** entity (such as invalid **name** or **null or empty nationality**), **do not** import any part of the entity and **append an error message** to the **method output**.
* If there are **any validation errors** for the **footballer** entity (such as invalid **name**, **start** or **end contract date** are missing or invalid, **contract start date** is after **contract end date**), **do not import it (only the footballer itself, not the whole coach info)** and **append an error message to the method output**.

|  |
| --- |
| **Success message** |
| Successfully imported coach – {**coachName**} with {**footballersCount**} footballers. |

**NOTE**: Do not forget to use **CultureInfo.InvariantCulture.**

##### Example

|  |
| --- |
| **coaches.xml** |
| <?xml version='1.0' encoding='UTF-8'?>  <Coaches>  <Coach>  <Name>S</Name>  <Nationality>25/01/2018</Nationality>  <Footballers>  <Footballer>  <Name>Benjamin Bourigeaud</Name>  <ContractStartDate>22/03/2020</ContractStartDate>  <ContractEndDate>24/02/2026</ContractEndDate>  <BestSkillType>2</BestSkillType>  <PositionType>2</PositionType>  </Footballer >  <Footballer >  <Name>Martin Terrier</Name>  <ContractStartDate>29/12/2021</ContractStartDate>  <ContractEndDate>16/06/2024</ContractEndDate>  <BestSkillType>2</BestSkillType>  <PositionType>3</PositionType>  </Footballer>  </Footballers>  </Coach>  ...  </Coaches> |
| **Output** |
| Successfully imported coach - Bruno Genesio with 2 footballers.  Invalid data!  Successfully imported coach - Antonio Conte with 3 footballers.  Invalid data!  Invalid data!  ... |

Upon **correct import logic**, you should have imported **22 coaches** and **38 footballers**.

### JSON Import

#### Import Teams

Using the file teams.json, import the data from that file into the database. Print information about each imported object in the format described below.

##### Constraints

* If any validation errors occur (such as invalid **name**, missing **nationality**, zero (0) or less **trophies**), **do not** import any part of the entity and **append an error message** to the **method output**.
* Take only the unique footballers.
* If a **footballer** does **not exist** in the database, **append an error message** to the **method output** and **continue** with the next **footballer**.

|  |
| --- |
| **Success message** |
| Successfully imported team - {**teamName**} with {**teamFootballersCount**} footballers. |

##### Example

|  |
| --- |
| **teams.json** |
| [  {  "Name": "Brentford F.C.",  "Nationality": "The United Kingdom",  "Trophies": "5",  "Footballers": [  28,  28,  39,  57  ]  },  {  "Name": "Chelsea F.C.",  "Nationality": "The United Kingdom",  "Trophies": "34",  "Footballers": [  38,  39,  59,  62,  57,  56  ]  }  …  ] |
| **Output** |
| Invalid data!  Invalid data!  Successfully imported team - Brentford F.C. with 1 footballers.  Invalid data!  **...** |

Upon **correct import logic**, you should have imported **24** **teams** and **35 footballers**.

## Data Export (25 pts)

**Use the provided methods in the** Serializer class**.** Usage of **Data Transfer Objects and AutoMapper** is **optional**.

### JSON Export

#### Export Teams With Most Footballers

Select the **top** 5 **teams** that have **at least one footballer** that **their contract start date is higher or equal** to the **given date. Select** them with their **footballers** who meet the **same criteria** (their contract start date is after or equals the given date). For each **team**, export their **name** and their **footballers.** For each **footballer**, export their **name** and contract **start date** (**must** be in format "**d**"), **contract end date** (**must** be in format "**d**"), **position** and **best skill** type**.** Order the **footballers** by **contract end date** (**descending**), then by **name** (**ascending**). Order the **teams** by **all** **footballers** (**meeting above condition**) **count** (**descending**), then by **name** (**ascending**).

**NOTE**: Do not forget to use **CultureInfo.InvariantCulture.** You **may** need to **call** **.ToArray()** function **before the selection** in order to **detach entities from the database** and **avoid runtime errors** (**EF Core bug**).

##### Example

|  |
| --- |
| Serializer.ExportTeamsWithMostFootballers(context, date) |
| [  {  "Name": "Manchester City F.C.",  "Footballers": [  {  "FootballerName": "Phil Foden",  "ContractStartDate": "12/30/2021",  "ContractEndDate": "04/13/2025",  "BestSkillType": "Dribble",  "PositionType": "Midfielder"  },  {  "FootballerName": "Ederson",  "ContractStartDate": "06/14/2021",  "ContractEndDate": "09/26/2024",  "BestSkillType": "Defence",  "PositionType": "Goalkeeper"  },  {  "FootballerName": "Ilkay Gundogan",  "ContractStartDate": "06/20/2020",  "ContractEndDate": "07/29/2024",  "BestSkillType": "Pass",  "PositionType": "Midfielder"  },  {  "FootballerName": "Kevin De Bruyne",  "ContractStartDate": "09/29/2020",  "ContractEndDate": "04/21/2024",  "BestSkillType": "Pass",  "PositionType": "Midfielder"  },  {  "FootballerName": "Bernardo Silva",  "ContractStartDate": "06/20/2020",  "ContractEndDate": "12/07/2022",  "BestSkillType": "Pass",  "PositionType": "Midfielder"  }  ]  }  …  ] |

### XML Export

#### Export Coaches with Their Footballers

Export all **coaches** that train at least **one** footballer. For each **coach**, export their **name** and **footballers count**. For each **footballer**, export their **name** and **position type.** Order the **footballers** by **name** (**ascending**). Order the **coaches** by **footballers count** (**descending**), then by **name** (**ascending**).

**NOTE**: You **may** need to **call** **.ToArray()** function **before the selection,** in order to **detach entities from the database** and **avoid runtime errors** (**EF Core bug**).

##### Example

|  |
| --- |
| **Serializer.ExportCoachWithTheirFootballers(context)** |
| <?xml version="1.0" encoding="utf-16"?>  <Coaches>  <Coach FootballersCount="5">  <CoachName>Pep Guardiola</CoachName>  <Footballers>  <Footballer>  <Name>Bernardo Silva</Name>  <Position>Midfielder</Position>  </Footballer>  <Footballer>  <Name>Ederson</Name>  <Position>Goalkeeper</Position>  </Footballer>  <Footballer>  <Name>Ilkay Gundogan</Name>  <Position>Midfielder</Position>  </Footballer>  <Footballer>  <Name>Kevin De Bruyne</Name>  <Position>Midfielder</Position>  </Footballer>  <Footballer>  <Name>Phil Foden</Name>  <Position>Midfielder</Position>  </Footballer>  </Footballers>  </Coach>  …  </Coaches> |