Databases Advanced Exam – 04 December 2021

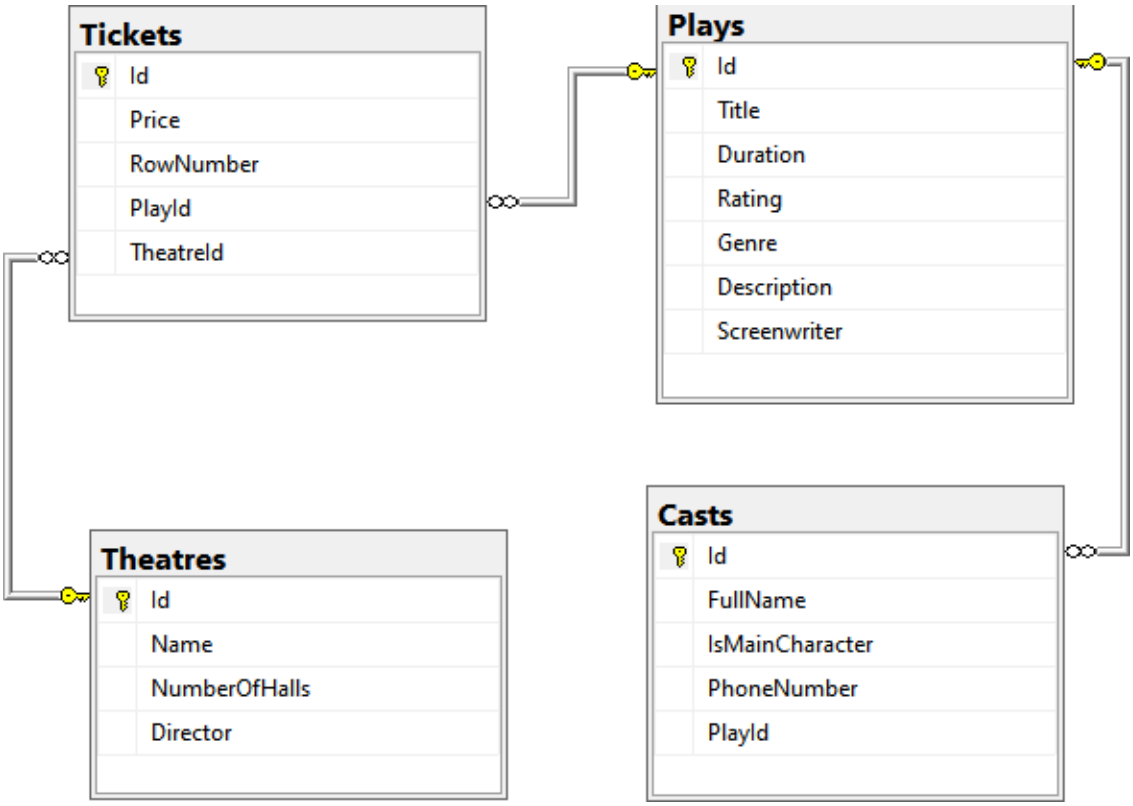
Exam problems for the Databases Advanced - Entity Framework course @ SoftUni.
Submit your solutions in the **SoftUni Judge** system (delete all **bin/obj** and **packages** folders) here.

NOTE: If you want to submit your solution in .NET Core 3.1, please use this link and the resources that are available in the Judge contest.

Before submitting your solutions in the **SoftUni Judge** system, delete all **bin/obj** and **packages** folders. If the **zip** file is still too large, you can delete the **ImportResults**, **ExportsResults** and **Datasets** folders too.

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.

Theatre



1. Project Skeleton Overview

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

- **Data** – contains the **TheatreContext** class, **Models** folder which contains the **entity classes,** and the **Configuration** class with **the connection string;**
- **DataProcessor** – contains the **Deserializer** and **Serializer** classes, which are used for **importing** and **exporting** data;
- **Datasets** – contains the **.json** and **.xml** files for the import part;
- **ImportResults** – contains the **import** results you make in the **Deserializer** class;

- **ExportResults** – contains the **export** results you make in the **Serializer** class.

2. Model Definition (50 pts)

Note: Foreign key navigation properties are required!

The application needs to store the following data:

Theatre

- **Id** – integer, **Primary Key**
- **Name** – **text** with **length [4, 30] (required)**
- **NumberOfHalls** – **sbyte** between **[1…10] (required)**
- **Director** – **text** with **length [4, 30] (required)**
- **Tickets** – a collection of type **Ticket**

Play

- **Id** – integer, **Primary Key**
- **Title** – **text** with **length [4, 50] (required)**
- **Duration** – **TimeSpan** in format **{hours:minutes:seconds}, with a minimum length of 1 hour. (required)**
- **Rating** – **float** in the range **[0.00….10.00] (required)**
- **Genre** – **enumeration** of type **Genre**, with possible values **(Drama, Comedy, Romance, Musical) (required)**
- **Description** – **text** with **length** up to **700** characters **(required)**
- **Screenwriter** – **text** with **length [4, 30] (required)**
- **Casts** – a collection of type **Cast**
- **Tickets** – a collection of type **Ticket**

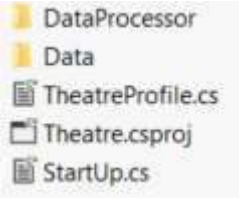
Cast

- **Id** – integer, **Primary Key**
- **FullName** – **text with length [4, 30] (required)**
- **IsMainCharacter** – **Boolean represents if the actor plays the main character in a play (required)**
- **PhoneNumber** – **text** in the following format: **"+44-{2 numbers}-{3 numbers}-{4 numbers}"**. Valid phone numbers are: **+44-53-468-3479, +44-91-842-6054, +44-59-742-3119 (required)**
- **PlayId** – **integer**, foreign key **(required)**

Ticket

- **Id** – integer, **Primary Key**
- **Price** – **decimal in the range [1.00….100.00] (required)**
- **RowNumber** – **sbyte in range [1….10] (required)**
- **PlayId** – integer, foreign key **(required)**
- **TheatreId** – integer, foreign key **(required)**

Test your solution in judge, by uploading a .zip file with the following files:



3. 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**. Usage of **Data Transfer Objects** and **Automapper** is **optional.**

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

NOTE: Do not modify the provided JSON and XML files, otherwise even on correct logic, you will not receive full points.

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

Error message
`Invalid data!`

XML Import

Import Plays

Using the file "**`plays.xml`**", 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: **`title/genre/rating/description/screenwriter`**
  - **`Duration`** of the play is less than **`1`** **(one) hour**

Do not import any part of the entity and **append an error message "`Invalid data!`"** to the **method output**.

- **Durations** will always be in the format '**`c`**'. Do not forget to use **`CultureInfo.InvariantCulture`**!

Success message
`Successfully imported {playTitle} with genre {genreType} and a rating of {rating}!`

Example

plays.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<Plays>
  <Play>
    <Title>The Hsdfoming</Title>
    <Duration>03:40:00</Duration>
    <Rating>8.2</Rating>
    <Genre>Action</Genre>
    <Description>A guyat Pinter turns into a debatable conundrum as oth ordinary and
menacing. Much of this has to do with the fabled "Pinter Pause," which simply
mirrors the way we often respond to each other in conversation, tossing in
```

```
remainders of thoughts on one subject well after having moved on to
another.</Description>
    <Screenwriter>Roger Nciotti</Screenwriter>
  </Play>
  <Play>
    <Title>Candida</Title>
    <Duration>02:21:00</Duration>
    <Rating>6.5</Rating>
    <Genre>Romance</Genre>
    <Description>What to do about Shaw? So many of his plays zing as comedies and
also still work as social commentary. Looking over his canon (pun sort of intended),
it struck me that this one of the 'Plays Pleasant' series might be most
important.</Description>
    <Screenwriter>Carmina Pollak</Screenwriter>
  </Play>
  <Play>
    <Title>The Hsdfasdng</Title>
    <Duration>03:40:00</Duration>
    <Rating>8.2</Rating>
    <Genre>Horror</Genre>
    <Description>A guyat Pinter turns into a debata Much of this has to do with the
fabled "Pinter Pause," which simply mirrors the way we often respond to each other
in conversation, tossing in remainders of thoughts on one subject well after having
moved on to another.</Description>
    <Screenwriter>Roger Ncioasdtti</Screenwriter>
  </Play>
  <Play>
    <Title>The Persianasd</Title>
    <Duration>00:35:00</Duration>
    <Rating>4.1</Rating>
    <Genre>Comedy</Genre>
    <Description></Description>
    <Screenwriter>Fidel Skirlin</Screenwriter>
  </Play>
</Plays>
…
```

Output

```
Invalid data!
Successfully imported Candida with genre Romance and a rating of 6.5!
Invalid data!
Invalid data!
...
```

Upon **correct import logic**, you should have imported **30 plays**.

Import Casts

Using the file "**`casts.xml`**", 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: **full name/phone number**

Do not import any part of the entity and **append an error message "`Invalid data!`"** to the **method output**. **`PlayId`** will be always valid.

Success message
`Successfully imported actor {fullName} as a {main/lesser} character!`

Example

casts.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<Casts>
  <Cast>
    <FullName>Van Tyson</FullName>
    <IsMainCharacter>false</IsMainCharacter>
    <PhoneNumber>+44-35-745-2774</PhoneNumber>
    <PlayId>26</PlayId>
  </Cast>
  <Cast>
    <FullName>Carlina Desporte</FullName>
    <IsMainCharacter>false</IsMainCharacter>
    <PhoneNumber>+44-00-715-9959</PhoneNumber>
    <PlayId>17</PlayId>
  </Cast>
  <Cast>
    <FullName>Elke Kavanagh</FullName>
    <IsMainCharacter>true</IsMainCharacter>
    <PhoneNumber>+44-53-468-3479</PhoneNumber>
    <PlayId>4</PlayId>
  </Cast>
  <Cast>
    <FullName>Lorry Ferreo</FullName>
    <IsMainCharacter>false</IsMainCharacter>
    <PhoneNumber>+44-03-229-7456</PhoneNumber>
    <PlayId>8</PlayId>
  </Cast>
  <Cast>
    <FullName>Vonny Henlon</FullName>
    <IsMainCharacter>true</IsMainCharacter>
    <PhoneNumber>+44-29-590-5125</PhoneNumber>
    <PlayId>2</PlayId>
  </Cast>
</Casts>
…
```

Output

```
Successfully imported actor Van Tyson as a lesser character!
Successfully imported actor Carlina Desporte as a lesser character!
Successfully imported actor Elke Kavanagh as a main character!
Successfully imported actor Lorry Ferreo as a lesser character!
Successfully imported actor Vonny Henlon as a main character!
...
```

Upon **correct import logic**, you should have imported **287 `actors`**.

JSON Import

Import Projections

Using the file "**`theatres-and-tickets.json`**", 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, **do not import any part of the entity** and **append an error message to the method output**.

- If there are any **`Ticket`** validation errors, do not import the invalid ticket, print **`"Invalid data!",`** and continue to the next ticket. **`PlayId`** will be always valid.

Success message
`Successfully imported theatre {theatreName} with #{totalNumber} tickets!`

Example

theatres-and-tickets.json

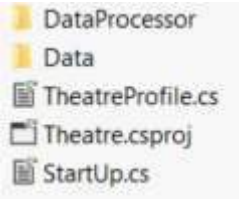
```
[
  {
    "Name": "",
    "NumberOfHalls": 7,
    "Director": "Ulwin Mabosty",
    "Tickets": [
      {
        "Price": 7.63,
        "RowNumber": 5,
        "PlayId": 4
      },
      {
        "Price": 47.96,
        "RowNumber": 9,
        "PlayId": 3
      }
    ]
  },
  {
    "Name": "Corona Theatre",
    "NumberOfHalls": 7,
    "Director": "Alwin MacCosty",
    "Tickets": [
      {
        "Price": 7.63,
        "RowNumber": -5,
        "PlayId": 4
      },
      {
        "Price": 47.96,
        "RowNumber": 9,
        "PlayId": 3
      },
      ….
    ]
  }
]
…
```

Output

```
Invalid data!
Invalid data!
Invalid data!
Invalid data!
Successfully imported theatre Corona Theatre with #17 tickets!
...
```

Upon **correct import logic**, you should have imported **30 theaters and 704 tickets**.

Test your solution in judge, by uploading a .zip file with the following files:



4. Data Export (25 pts)

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

JSON Export

Export Top Theaters

The given method in the project's skeleton receives a number representing the number of halls. Export **all theaters** where the hall's count is bigger or equal to the given and **have 20 or more tickets available**. For each **theater**, export its **`Name`**, **`Halls`**, **`TotalIncome`** of tickets which are between the first and fifth row inclusively, and **`Tickets`**. For each **ticket (between first and fifth row inclusively)**, export its **price**, and **the row number.** Order the **theaters** by the number of halls **descending**, then by name (**ascending**). Order the tickets by **price descending**.

NOTE: If you receive correct output when running the query locally, but judge gives an error, materialize the query (**`.ToArray()`**, **`.ToList()`**, etc.) before the **`.Where()`** statement.

Example

`Serializer.ExportTheaters(context, numbersOfHalls)`

```
[
  {
    "Name": "Capitol Theatre Building",
    "Halls": 10,
    "TotalIncome": 860.02,
    "Tickets": [
      {
        "Price": 93.48,
        "RowNumber": 3
      },
      {
        "Price": 93.41,
        "RowNumber": 1
      },
      {
        "Price": 86.21,
        "RowNumber": 5
      },
      {
        "Price": 86.14,
        "RowNumber": 5
      },
      {
        "Price": 85.64,
        "RowNumber": 4
      },
      {
```

```
        "Price": 85.09,
        "RowNumber": 1
      },
      {
        "Price": 79.01,
        "RowNumber": 3
      },
      {
        "Price": 68.56,
        "RowNumber": 1
      },
      {
        "Price": 62.14,
        "RowNumber": 3
      },
      {
        "Price": 55.96,
        "RowNumber": 1
      },
      {
        "Price": 40.18,
        "RowNumber": 5
      },
      {
        "Price": 13.83,
        "RowNumber": 3
      },
      {
        "Price": 10.37,
        "RowNumber": 3
      }
    ]
  },  ...
]
```


XML Export

Export Plays

Use the method provided in the project skeleton, which receives a rating. Export all plays with a rating equal or smaller to the given. For each **play**, export **`Title`**, **`Duration`** **(**in the format: '**c'), `Rating`,** **`Genre`,** and **`Actors`** which play the **main character only**.

Keep in mind:

- If the **rating is `0`**, you should print **`"Premier".`**
- For each actor display:
  - **`FullName`**
  - **`MainCharacter`** in the format: **`"Plays main character in '{playTitle}'."`**

Order the result by **play title (ascending), then by genre (descending).** Order **actors by their full name descending.**

Example

`Serializer.ExportPlays(context, rating)`

```
<?xml version="1.0" encoding="utf-16"?>
<Plays>
  <Play Title="A Raisin in the Sun" Duration="01:40:00" Rating="5.4" Genre="Drama">
    <Actors>
```

```
      <Actor FullName="Sylvia Felipe" MainCharacter="Plays main character in 'A Raisin in the
Sun'." />
      <Actor FullName="Sella Mains" MainCharacter="Plays main character in 'A Raisin in the
Sun'." />
      <Actor FullName="Sela Hillett" MainCharacter="Plays main character in 'A Raisin in the
Sun'." />
      <Actor FullName="Rodney O'Neill" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
      <Actor FullName="Robbert Tuvey" MainCharacter="Plays main character in 'A Raisin in the
Sun'." />
      <Actor FullName="Reamonn Maleby" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
      <Actor FullName="Loutitia Joy" MainCharacter="Plays main character in 'A Raisin in the
Sun'." />
      <Actor FullName="Irving Houlridge" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
      <Actor FullName="Cristine Van Brug" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
      <Actor FullName="Clerissa Fellgate" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
      <Actor FullName="Caye Blacklawe" MainCharacter="Plays main character in 'A Raisin in
the Sun'." />
    </Actors>
  </Play>
  <Play Title="Awake and Sing" Duration="02:41:00" Rating="3.8" Genre="Drama">
    <Actors>
      <Actor FullName="Whitney Standering" MainCharacter="Plays main character in 'Awake and
Sing'." />
      <Actor FullName="Mannie Plomer" MainCharacter="Plays main character in 'Awake and
Sing'." />
      <Actor FullName="Karlene Vasyutochkin" MainCharacter="Plays main character in 'Awake
and Sing'." />
      <Actor FullName="Estelle Haycox" MainCharacter="Plays main character in 'Awake and
Sing'." />
      <Actor FullName="Christian Geere" MainCharacter="Plays main character in 'Awake and
Sing'." />
      <Actor FullName="Aura Wauchope" MainCharacter="Plays main character in 'Awake and
Sing'." />
      <Actor FullName="Andie Greatham" MainCharacter="Plays main character in 'Awake and
Sing'." />
    </Actors>
…
</Plays>
```

Test your solution in judge, by uploading a .zip file with the following files: