# Databases Advanced Exam - 13 December 2019

Judge 🡪 <https://judge.softuni.bg/Contests/1973/CSharp-DB-Advanced-Exam-13-Dec-2019>

Exam problems for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/courses/entity-framework-core). 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.

# BookShop



## Project Skeleton Overview

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

* Data - contains the BookShopContext class, Models folder which contains the **entity classes** and the **Configuration** class with **connection string**
* DataProcessor - contains the Serializer and Deserializerclasses, 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

## Model Definition (50 pts)

The application needs to store the following data:

### Author

* Id - integer, Primary Key
* FirstName - text with length **[3, 30]**. (**required**)
* LastName - text with length **[3, 30]**. (**required**)
* Email - text (**required**). Validate it! There is attribute for this job.
* Phone - text. Consists only of **three groups** (**separated** by '-'), the **first two** consist of **three** **digits** and the **last** one - of **4** digits. (**required**)
* AuthorsBooks - collection of type AuthorBook

### Book

* Id - integer, **Primary Key**
* Name - text with length **[3, 30]**. (**required**)
* Genre - enumeration of type Genre, with possible values **(**Biography = 1, Business = 2, Science = 3**)** (**required**)
* Price - decimal in **range** between **0.01** and **max** value of the decimal
* Pages – integer in **range** between **50** and **5000**
* PublishedOn - date and time (**required**)
* AuthorsBooks - collection of type AuthorBook

### AuthorBook

* AuthorId - integer, Primary Key, Foreign key (required)
* Author - Author
* BookId -integer, Primary Key, Foreign key (required)
* Book - Book

## 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 DataTransferObjects 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.

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 Books

Using the file **books.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 **book** entity (such as invalid **name**, **genre**, **price**, **pages** or **published** **date**), **do not** import any part of the entity and **append an error message** to the **method output**.

**NOTE**: **Date** will be in format "MM/dd/yyyy", do not forget to use **CultureInfo.InvariantCulture**

|  |
| --- |
| **Success message** |
| Successfully imported book {**bookName**} for {**bookPrice**}. |

##### Example

|  |
| --- |
| **books.xml** |
| <?xml version='1.0' encoding='UTF-8'?>  <Books>  <Book>  <Name>Hairy Torchwood</Name>  <Genre>3</Genre>  <Price>41.99</Price>  <Pages>3013</Pages>  <PublishedOn>01/13/2013</PublishedOn>  </Book>  <Book>  <Name>Anise Burnet Saxifrage</Name>  <Genre>1</Genre>  <Price>-1.51</Price>  <Pages>2920</Pages>  <PublishedOn>12/02/2015</PublishedOn>  </Book>  <Book>  <Name>Hand Fern</Name>  <Genre>2</Genre>  <Price>3.57</Price>  <Pages>5303</Pages>  <PublishedOn>02/23/2018</PublishedOn>  </Book>  ...  </Books> |
| **Output** |
| **Successfully imported book Hairy Torchwood for 41.99.**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Successfully imported book Mojave Linanthus for 82.29.**  **...** |

Upon **correct import logic**, you should have imported **70** books.

### JSON Import

#### Import Authors

Using the file authors.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 **first name**, **last** **name**, **email** or **phone**), **do not** import any part of the entity and **append an error message** to the **method output**.
* If an **email** exists, do **not** **import** the **author** and append and **error message**.
* If a **book** does **not exist** in the database, **do not** **append an error message** and **continue** with the next **book**.
* If an **author** have **zero** **books** (all books are invalid) do **not import** the **author** and **append an error message** to the **method output**.

|  |
| --- |
| **Success message** |
| Successfully imported author - {**first name + last name**} with {**booksCount**} books. |

##### Example

|  |
| --- |
| **authors.json** |
| [  {  "FirstName": "K",  "LastName": "Tribbeck",  "Phone": "808-944-5051",  "Email": "btribbeck0@last.fm",  "Books": [  {  "Id": 79  },  {  "Id": 40  }  ]  },  {  "FirstName": "Maridel",  "LastName": "N",  "Phone": "658-437-4751",  "Email": "mdeamaya1@theatlantic.com",  "Books": [  {  "Id": 117  },  {  "Id": 88  }  ]  },  ...  ] |
| **Output** |
| **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Invalid data!**  **Successfully imported author - Nataniel Pembery with 2 books.**  **Successfully imported author - Aila Fallanche with 2 books.**  **...** |

Upon **correct import logic**, you should have imported **75** **authors** and **150 author books**.

## Data Export (25 pts)

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

### JSON Export

#### Export Most Craziest Authors

Select **all authors** along with their books. Select their name in format **first name** + ' ' + **last name**. For each **book** select its **name** and **price** formatted to the **second digit** after the **decimal point**. Order the **books by price** in **descending** order. Finally sort **all authors** by **book count** descending and then by author full name.

**NOTE**: **Before the orders, materialize the query (This is issue by Microsoft in InMemory database library)!!!**

##### Example

|  |
| --- |
| Serializer.ExportMostCraziestAuthors(context) |
| [  {  "AuthorName": "Angelina Tallet",  "Books": [  {  "BookName": "Allen Fissidens Moss",  "BookPrice": "78.44"  },  {  "BookName": "Earlyleaf Brome",  "BookPrice": "63.66"  },  {  "BookName": "Sky Mousetail",  "BookPrice": "13.14"  },  {  "BookName": "Arrowleaf Clover",  "BookPrice": "1.71"  }  ]  },  {  "AuthorName": "Ashia Esh",  "Books": [  {  "BookName": "Twoflower Melicgrass",  "BookPrice": "29.06"  },  {  "BookName": "Sky Mousetail",  "BookPrice": "13.14"  },  {  "BookName": "Candle Tree",  "BookPrice": "9.00"  },  {  "BookName": "Arrowleaf Clover",  "BookPrice": "1.71"  }  ]  },  ...  ] |

### XML Export

#### Export Oldest Books

Export **top 10** **oldest** books that are **published before** the given date and are of **type science**. For each book select its **name**, **date** (in format "**d**") and **pages**. Sort them by **pages** in **descending** order and then by **date** in **descending** order.

**NOTE**: **Before the orders, materialize the query (This is issue by Microsoft in InMemory database library)!!!**

##### Example

|  |
| --- |
| **Serializer.ExportOldestBooks(context, date)** |
| <?xml version="1.0" encoding="utf-16"?>  <Books>  <Book Pages="4881">  <Name>Sierra Marsh Fern</Name>  <Date>03/18/2016</Date>  </Book>  <Book Pages="4786">  <Name>Little Elephantshead</Name>  <Date>12/16/2014</Date>  </Book>  <Book Pages="3245">  <Name>Airplant</Name>  <Date>11/24/2016</Date>  </Book>  <Book Pages="3039">  <Name>Palo Blanco</Name>  <Date>06/25/2014</Date>  </Book>  <Book Pages="3013">  <Name>Hairy Torchwood</Name>  <Date>01/13/2013</Date>  </Book>  <Book Pages="1870">  <Name>Bigelow's Monkeyflower</Name>  <Date>11/20/2015</Date>  </Book>  ...  </Books> |