# Entity Framework Core Retake Exam - 14 April 2024

Exam problems for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/trainings/4842/entity-framework-core-february-2025).  
Submit your solutions in the **SoftUni Judge** system (delete all **bin**/**obj** and **packages** folders) [here](https://judge.softuni.org/Contests/5393/Entity-Framework-Core-Retake-Exam-14-April-2025).

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 skip the attachments to the **ImportResults**, **ExportsResults,** and **Datasets** 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.

# Art Collective



## Project Skeleton Overview

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

1. Data – contains the ArtCollectiveDbContext class and Models folder, which contains the **entity classes**
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 (60 pts)

The application needs to store the following data:

### Artist

* Id– **integer**, **Primary Key**
* Username– **text** with length **[5, 30] (required)**
* Email– **text** with length **[6, 50]** **(required)**
* Password – **text** with a **minimum** **length** of **4 (required)**
* Artworks – acollection of type Artwork
* Feedbacks – acollection of type Feedback
* ArtistsGroups – acollection of type ArtistGroup

### Group

* Id– **integer**, **Primary Key**
* Title **– text** with length **[3, 50] (required)**
* StartedOn– **DateTime (required)**
* Feedbacks– acollection of type Feedback
* ArtistsGroups – acollection of type ArtistGroup

### ArtistGroup

* ArtistId– integer, Primary Key, Foreign Key (required)
* Artist– Artist
* GroupId– integer, Primary Key, Foreign Key (required)
* Group – Group

### Artwork

* Id– **integer**, **Primary Key**
* Title **– text** with length **[3, 50]** (**required**)
* Description **– text** with length **[10, 300]** (**NOT** **required**)
* CreatedOn– **DateTime (required)**
* ArtistId– **integer**, Foreign Key **(required)**
* Artist– **Artist**

### Collaboration

* ArtistOneId– **integer**, Foreign Key (required)
* ArtistOne - Artist
* ArtistTwoId – **integer**, Foreign Key (required)
* ArtistTwo - Artist

### Feedback

* Id– integer, **Primary Key**
* Content– **text** with length **[3, 200]** (**required**)
* GivenOn– **DateTime (required)**
* Status – enum Status(Pending = 0, Reviewed, Published, Rejected)(**required**)
* GroupId - **integer**, Foreign Key **(required)**
* Group - Group
* ArtistId - **integer**, Foreign Key **(required)**
* Artist – Artist

## Data Import (20pts)

For the functionality of the application, you need to create several methods that manipulate the database. The **project skeleton** already provides these methods within the Deserializer class. Usage of DataTransferObjects or **AutoMapper** is **optional**.

**Clarification on Initial Data Seeding:**

To ensure the application's functionality, it is essential to **populate the database with initial data**. Inside the **DbContext class**, you will find a **commented-out section** specifically designed for seeding data.   
**Before applying migrations** and updating the database, please **uncomment** this section **in the** OnModelCreating **method**.

If the **models are created correctly**, the **database is built successfully**, and the **seeding code is uncommented**, the following data should be automatically populated:

* 10 Artists
* 12 Collaborations
* 8 Groups
* 28 ArtistsGroups

**Serialize & Deserialize files:**

Use the provided **JSON** and **XML** files to populate the database with data. **Import all the valid information** from the 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 format. |

**If some data appears to be duplicated, do not import the entity; print a duplication data message:**

|  |
| --- |
| **Duplication message** |
| Data is duplicated. |

***Error messages and duplication messages will be provided as constants in the skeleton.***

### XML Import

#### Import Feedbacks

Using the file "**feedbacks.xml"**, **import the data from the file** into the database.

Each imported <Feedback> **should be validated** and **added to the database if it meets the specified constraints**.

The method should **return a string containing information about each import attempt**, formatted as described.

##### Constraints

* Each feedback entity must be validated against the following criteria:
  + **Content** – Must have a length between **3 and 200 characters** (inclusive)
  + GivenOn – Must be a **valid DateTime format**

The **DateTime** **data** in the document will be in the following format: "yyyy-MM-dd"  
Make sure you use CultureInfo.InvariantCulture

* + **Status** - Must be a **valid enum value**: **[ Pending (0), Reviewed (1), Published (2), Rejected (3)]**
  + **GroupIdId** – Must reference an **existing Group** in the database
  + **ArtistId** – Must reference an **existing Artist** in the database
* **Duplication Check** - Before adding feedback to the database**, ensure there are no existing records  
   With the same**:
  + **Content, GivenOn, Status, and ArtistId** within the same **Group**
  + If such **feedback already exists**, **do not import** it and append a **duplication message** to the output
* If **any validation error occurs** for a feedback entity **or if any field references a non-existent Group OR Artist**, the **feedback should not be imported**. The appropriate **error message** or **duplication message should be appended** to the method's output
* **Success Messages**
  + For **each successfully imported feedback**, append a **success message** to the output, formatted as **Successfully imported feedback (Given on: {"yyyy-MM-dd"}, Status: {Status})**
* **Data Persistence**
  + After processing all feedbacks from the XML file:
    - **Add the valid feedbacks entities** to the proper collection
    - **Save the changes** to the database

|  |
| --- |
| **Success message** |
| Successfully imported feedback – (Given on: {"yyyy-MM-dd"}, Status: {Status}) |

##### Example

|  |
| --- |
| feedbacks.xml |
| <?xml version="1.0" encoding="UTF-8"?>  <Feedbacks>    <Feedback GivenOn="2025-03-01">  <Content>The group's creative direction is truly refreshing.</Content>  <Status>Published</Status>  <GroupId>1</GroupId>  <ArtistId>1</ArtistId>  </Feedback>  <Feedback GivenOn="2025-03-02">  <Content>Excited to work with such a talented team!</Content>  <Status>Pending</Status>  <GroupId>1</GroupId>  <ArtistId>2</ArtistId>  </Feedback>  <!-- Invalid Content (empty) -->  <Feedback GivenOn="2025-04-04">  <Content></Content>  <Status>Published</Status>  <GroupId>1</GroupId>  <ArtistId>3</ArtistId>  </Feedback>  <Feedback GivenOn="2025-03-03">  <Content>Impressive vision for modern art evolution.</Content>  <Status>Reviewed</Status>  <GroupId>1</GroupId>  <ArtistId>4</ArtistId>  </Feedback>  …  <Feedbacks> |
| **Output** |
| Successfully imported feedback (Given on: 2025-03-01, Status: Published)  Successfully imported feedback (Given on: 2025-03-02, Status: Pending)  Invalid data format.  Successfully imported feedback (Given on: 2025-03-03, Status: Reviewed)  Successfully imported feedback (Given on: 2025-03-04, Status: Published)  Successfully imported feedback (Given on: 2025-03-05, Status: Reviewed)  ... |

Upon **correct import logic**, you should have imported **35 feedbacks**

### JSON Import

#### Import Posts

Using the file **"**artworks.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 error occurs** for the post entity (**invalid date format, missing required fields, non-existent artist ID**), **do not** import the entity and append an **error message** to the method output.
* **Duplication Check** - Before adding an artwork to the database**, ensure there are no existing records  
   with the same:** **Title and ArtistId**

The **CreatedAt** data in the document will be in the following format: "yyyy-MM-dd"  
Make sure you use CultureInfo.InvariantCulture

* **The artists associated with every artwork** are represented by an **integer ID**, which **must match an existing record** in the database.

|  |
| --- |
| **Success message** |
| Successfully imported artwork (Artist: {artistUsername}, Created on: {"yyyy-MM-dd"}) |

##### Example

|  |
| --- |
| **artworks.json** |
| [  {  "Title": "Florence at Dawn",  "Description": "Soft hues reflecting the quiet stillness of Florence as the sun rises.",  "CreatedOn": "2025-03-01",  "ArtistId": 1  },  {  "Title": "Architectural Echoes",  "Description": "A study in symmetry, inspired by Renaissance columns and arches.",  "CreatedOn": "2025-03-02",  "ArtistId": 1  },  {  "Title": "Urban Renaissance",  "Description": "Modern city lines blended with classical composition.",  "CreatedOn": "2025-03-03",  "ArtistId": 1  },  {  "Title": "Golden Streets",  "Description": "An interpretation of golden hour in ancient alleyways.",  "CreatedOn": "2025-03-04",  "ArtistId": 1  },  {  "Title": "Perspective Shift",  "Description": "An exploration of inverted space and mirrored light.",  "CreatedOn": "2025-03-05",  "ArtistId": 1  },  ...  ] |
| **Output** |
| Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-01)  Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-02)  Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-03)  Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-04)  Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-05)  Successfully imported artwork (Artist: alessandro\_rossi, Created on: 2025-03-06)  Invalid data format.  Data is duplicated.  Successfully imported artwork (Artist: marie\_dupont, Created on: 2025-03-01)  **...** |

Upon **correct import logic**, you should have imported **46** **artworks**

## Data Export (20 pts)

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

### XML Export

#### Export Artists With Collaborations Count And Their Artworks

Export **all artists,** along with **their collaboration counts and artworks**. The exported data should be in **XML format**.

Each artist must include:

* Their **username**
* The **number of collaborations** they have (based on the **Collaborations** **table**, where the artist appears as **either ArtistOneId or ArtistTwoId**
* All their **artworks**, **ordered by Id** (ascending)
* The result should be ordered **alphabetically by artist username**

**Data Fields**:

* **Artist**
  + **Username** – the username of the artist
  + **Collaborations** – the total number of collaborations the artist is involved in
  + **Artworks** – the collection of the artist's artworks
* **Artwork**
  + **Title** – the title of the artwork
  + **CreatedOn** – the date the artwork was created, formatted as **"yyyy-MM-dd"**

**Expected XML Output**:

* The root element should be <Artists>
* Each artist should be represented by an <Artist> **element** with an **attribute** Collaborations=X' indicating the **number of collaborations they are involved in**
* The username must be inside a <Username> **element**
* The artworks must be wrapped in an <Artworks> **element**, each artwork being represented by a <Artwork> element with:
  + **<Title>**
  + **<CreatedOn>**

##### Example

|  |
| --- |
| ExportArtistsWithCollaborationsCountAndTheirArtworks(dbContext) |
| <?xml version="1.0" encoding="utf-16"?>  <Artists>  <Artist Collaborations="2">  <Username>alessandro\_rossi</Username>  <Artworks>  <Artwork>  <Title>Florence at Dawn</Title>  <CreatedOn>2025-03-01</CreatedOn>  </Artwork>  <Artwork>  <Title>Architectural Echoes</Title>  <CreatedOn>2025-03-02</CreatedOn>  </Artwork>  <Artwork>  <Title>Urban Renaissance</Title>  <CreatedOn>2025-03-03</CreatedOn>  </Artwor  ...  </Artist  ...  <Artists> |

### JSON Export

#### Export All Groups With Feedbacks Chronologically

Export all **groups** along with their **feedback**. The exported data should be in **JSON format** and follow the structure below:

* **Selection Criteria**:
  + Select **all groups** in the database
  + For each group, export its **ID**, **title,** and **starte on**
  + Include **all feedbacks given to the group**
* **Data Fields**:
  + Group – **Id, Title, StartedOn**
  + Feedback – **Content, GivenOn, Status, ArtistUsername**
* **Ordering:**
  + Groups should be **ordered by StartedOn ascending**
  + Feedbacksshould be **ordered by GivenOn ascending**
* **DateTime Format:**
  + **All date and time fields (**StartedOnandGivenOn**)** must be formatted as **"**yyyy-MM-dd**"**
  + **Example: "**2025-02-24**"**
  + Ensure thatCultureInfo.InvariantCultureis used for correct serialization

##### Example

|  |
| --- |
| **ExportGroupsWithFeedbacksChronologically(dbContext)** |
| [  {  "Id": 1,  "Title": "Impressionisti Moderni",  "StartedOn": "2024-01-15",  "Feedbacks": [  {  "Content": "The group's creative direction is truly refreshing.",  "GivenOn": "2025-03-01",  "Status": 2,  "ArtistUsername": "alessandro\_rossi"  },  {  "Content": "Excited to work with such a talented team!",  "GivenOn": "2025-03-02",  "Status": 0,  "ArtistUsername": "marie\_dupont"  },  {  "Content": "Impressive vision for modern art evolution.",  "GivenOn": "2025-03-03",  "Status": 1,  "ArtistUsername": "antoine\_leblanc"  },  {  "Content": "The collaboration feels seamless and inspiring.",  "GivenOn": "2025-03-04",  "Status": 2,  "ArtistUsername": "nicolas\_moreau"  },  {  "Content": "This group embraces innovation — love it!",  "GivenOn": "2025-03-05",  "Status": 1,  "ArtistUsername": "chiara\_ferrari"  },  {  "Content": "I believe we're setting a new standard in the art scene.",  "GivenOn": "2025-03-06",  "Status": 2,  "ArtistUsername": "élise\_martin"  }  ]  },  {  "Id": 2,  "Title": "Art Contemporain",  "StartedOn": "2024-02-10",  "Feedbacks": [  ...  ]  }  ...  ] |

## Prepare the Solution File Archive

A screenshot of a computer

AI-generated content may be incorrect.