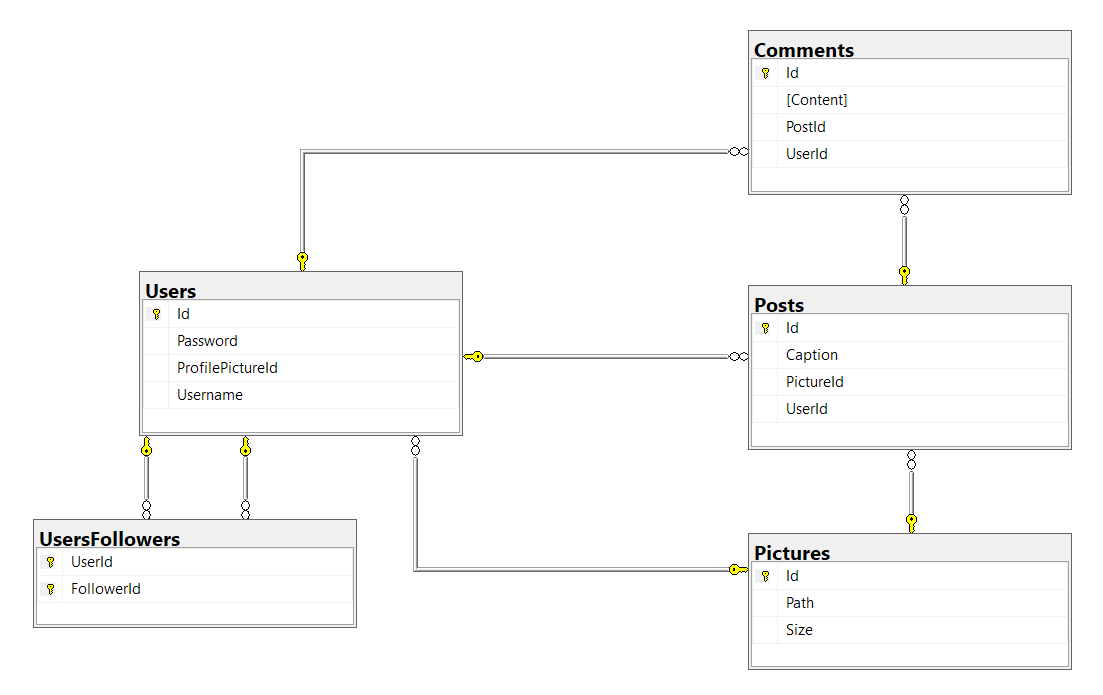
# Entity Framework Core: Exam Preparation 1

Exam problems for the ["Databases Advanced – EF Core" course @ Software University](https://softuni.bg/trainings/1741/databases-advanced-entity-framework-october-2017). Submit your solutions in the SoftUni judge system (delete all "bin"/"obj" 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.

# Instagraph

Some of you may know about Instagraph, some of you may not. For those who don’t, Instagraph is the backup **database** of **Instagram**. And guess what, you were chosen to work on it. This time your task is to create a whole **Entity** **Framework** **Core** **application** that will work with the **database**.



## Project Skeleton Overview

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

* Instagraph.App – contains the Startup class, which is the **entry point of the application**. Also contains an AutoMapperprofile (InstagraphProfile), which you can configure if you choose to use AutoMapper in your app.
* Instagraph.Data – contains the InstagraphContext class and the **connection string**
* Instagraph.Models – contains the **entity classes**
* Instagraph.DataProcessor – contains the Serializer and Deserializerclasses, which are used for **importing** and **exporting** data

## Problem 1. Model Definition (50 pts)

Every user has a **profile** **picture**, many **posts and** many **comments** onposts. Many **users** can have the same **profile** **picture**. In addition, a **user** can **follow** **many** **other** **users** and be **followed** **by** **many** **other** **users**. Any **picture** can be a part of **many** **posts**. Each post is **created** by a **single** **user** and is about a **certain** **picture**. A **post** can have **many** **comments**.

### Picture

* Id – an **integer**, **Primary Key**
* Path – a **string**
* Size – a **decimal**
* Users – a **Collection** of type User
* Posts – a **Collection** of type Post

### User

* Id – an **integer**, **Primary Key**
* Username – a **string** with **max length 30**, **Unique**
* Password – a **string** with **max length 20**
* ProfilePictureId – an **integer**
* ProfilePicture – a Picture
* Followers – a **Collection** of type UserFollower
* UsersFollowing – a **Collection** of type UserFollower
* Posts – a **Collection** of type Post
* Comments – a **Collection** of type Comment

### Post

* Id – an **integer, Primary Key**
* Caption – a **string**
* UserId – an **integer**
* User – a User
* PictureId – an **integer**
* Picture – a Picture
* Comments – a **Collection** of type **Comment**

### Comment

* Id – an **integer**, **Primary Key**
* Content – a **string** with **max** **length** **250**
* **UserId** – an **integer**
* User – a User
* **PostId** – an **integer**
* Post – a Post

### UserFollower

* UserId – an **integer**, **Primary Key**
* User – a User
* **FollowerId** – an **integer**, **Primary Key**
* **Follower** – a **User**

**All data** is **REQUIRED**, unless it is explicitly said that **null** is **allowed**.

## Problem 2. Data Import (35pts)

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 Instagraph.DataProcessor project inside your solution. Use **Data Transfer Objects** as needed.

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.

Make sure all fields have been entered and are valid, otherwise the import **entity** data **should NOT be considered valid**.

You will also have to print a simple message indicating if the data has been imported successfully or there was an error.

In case of **SUCCESS** the message format is:

* **Picture**: "Successfully imported Picture {picturePath}."
* **User**: "Successfully imported User {username}."
* **Post**: "Successfully imported Post {postCaption}.”.
* **Comment**: "Successfully imported Comment {commentContent}."
* **User - Follower**: "Successfully imported Follower {followerUsername} to User {userUsername}."

In case of **ERROR** you always print "Error: Invalid data.".

### JSON Import (15 pts)

#### Importing Pictures

Using the file **pictures.json**, import the data from that file into the database.

#### Pictures

A picture is **valid** when it has a **unique** **path**, which is not an **empty** **string** and has a size, which is **bigger** **than 0**.

##### Example

|  |
| --- |
| **pictures.json** |
| **[**  **{**  **"Path" : "src/folders/resources/images/profile/blocked/bmp/kjOJjKpKh4.bmp",**  **"Size" : 32495.57**  **},**  **{**  **"Path" : "src/folders/resources/images/post/timeline/png/27kLXVm22Q.png",**  **"Size" : 44273.27**  **},**  **{**  **"Path" : "src/folders/resources/images/profile/browsed/bmp/Q52q15Zefa.bmp"**  **},**  **…**  **]** |
| **Output** |
| Successfully imported Picture src/folders/resources/images/profile/blocked/bmp/kjOJjKpKh4.bmp.  Successfully imported Picture src/folders/resources/images/post/timeline/png/27kLXVm22Q.png.  Error: Invalid data.  **…** |

#### Users

A **user** must have a **valid** **profile** **picture**, **username** and **password**.

##### Example

|  |
| --- |
| **users.json** |
| **[**  **{**  **"Username" : "UnderSinduxrein",**  **"Password" : "4l8nYGTKMW",**  **"ProfilePicture" : "src/folders/resources/images/post/formed/digi/6YLvj97k03.digi"**  **},**  **{**  **"Username" : "BlaAntigadsa",**  **"Password" : ":Q5wjT4[e"**  **},**  **{**  **"Password" : "El[MwhxY)J",**  **"ProfilePicture" : "src/folders/resources/images/profile/blocked/jpg/pgfMG75k4e.jpg"**  **},**  **…**  **]** |
| **Output** |
| Successfully imported User UnderSinduxrein.  Error: Invalid data.  Error: Invalid data.  **…** |

#### Followers

To make **someone** a **follower** of **another** **user**, both of them **must** **exist** in the **database**.

##### Example

|  |
| --- |
| **users\_followers.json** |
| **[**  **{**  **"User" : "BlaImmagiIana",**  **"Follower" : "ScoreSinduxIana"**  **},**  **{**  **"User" : "BlaSinduxrein",**  **"Follower" : "RoundInspecindi"**  **},**  **{**  **"User" : "AryaNinehow",**  **"Follower" : "DarkImmagidsa"**  **}, …**  **]** |
| **Output** |
| Successfully imported Follower ScoreSinduxIana to User BlaImmagiIana.  Successfully imported Follower RoundInspecindi to User BlaSinduxrein.  Successfully imported Follower DarkImmagidsa to User AryaNinehow.  **…** |

### XML Import (5 pts)

The other **2 tables** must be populated with data in **XML** format.

#### Posts

A **post** should only be inserted if the **user** and **picture** **already** **exist** in the **database**.

##### Example

|  |
| --- |
| **posts.xml** |
| **<?xml version="1.0" encoding="utf-8"?>**  **<posts>**  **<post>**  **<caption>#everything #swag #sunglasses #smiley #justdoit #ocean</caption>**  **<user>ScoreAntigarein</user>**  **<picture>src/folders/resources/images/story/blocked/png/1S2el3wJ3v.png</picture>**  **</post>**  **<post>**  **<caption>#cool #justdoit #sky #ocean #reason #gram #faith #hope</caption>**  **<user>HighAsmahow</user>**  **<picture>src/folders/resources/images/profile/blocked/jpg/pgfMG75k4e.jpg</picture>**  **</post>**  …  **</posts>** |
| **Output** |
| Successfully imported Post #everything #swag #sunglasses #smiley #justdoit #ocean.  Successfully imported Post #cool #justdoit #sky #ocean #reason #gram #faith #hope.  **…** |

#### Comments

Comments **should** only be **added** for **existing** **users** and **posts**.

##### Example

|  |
| --- |
| **comments.xml** |
| **<?xml version="1.0" encoding="utf-8"?>**  **<comments>**  **<comment>**  **<content>Wow! Wow, epic!! How?</content>**  **<user>RoundAntigaBel</user>**  **<post id="22" />**  **</comment>**  **<comment>**  **<content>This is just... this is the best! Why?</content>**  **<user>BlaImmagiIana</user>**  **<post id="50" />**  **</comment>**  **…**  **</comments>** |
| **Output** |
| Successfully imported Comment Wow! Wow, epic!! How?.  Error: Invalid data.  **…** |

## Problem 3. Data Export (15pts)

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 Instagraph.DataProcessor project inside your solution. Use **Data Transfer Objects** as needed.

### JSON Export (5 pts)

#### Uncommented Posts

Export all posts which do **NOT** have **ANY comments**. Extract everything and **order** it by **id** in **ascending order**.

Extract the Id, Path as "**Picture**" and the **poster**’s Username as "U**ser**".

##### Example

|  |
| --- |
| **UncommentedPosts.json** |
| **[**  **{**  **"Id": 1,**  **"Picture": "src/folders/resources/images/story/blocked/png/1S2el3wJ3v.png",**  **"User": "ScoreAntigarein"**  **},**  **{**  **"Id": 4,**  **"Picture": "src/folders/resources/images/story/blocked/png/1S2el3wJ3v.png",**  **"User": "DarkImmagidsa"**  **},**  **…**  **]** |

#### Popular users

Extract all **users** who have **posts**, **commented** by **their followers**.

Extract the user’s username as “**user**”, and the user’s count of followers as “**followers**”.

Order the data by **id** in **descending order**.

##### Example

|  |
| --- |
| **PopularUsers.json** |
| **[**  **{**  **"Username": "ZendArmyhow",**  **"Followers": 7**  **}**  **]** |

### XML Export (5 pts)

#### Comments on Posts

Extract **all users**. For each **user**, **extract** the **post** which has the **most comments**.

Extract the user’s username as "**Username**" and the count of comments on the top post as   
"**MostComments**".

If a **person** has **no posts**, set his **comments** to **0**.

**Order** the **data** in **descending order** by **MostComments, then by Username ascending**.

##### Example

|  |
| --- |
| **CommentsOnPosts.xml** |
| **<users>**  **<user>**  **<Username>ScoreAntigarein</Username>**  **<MostComments>3</MostComments>**  **</user>**  **<user>**  **<Username>AryaDenotehow</Username>**  **<MostComments>2</MostComments>**  **</user>**  **<user>**  **<Username>HighAsmahow</Username>**  **<MostComments>2</MostComments>**  **</user>**  **…**  **</users>** |