# Databases Advanced Exam - 15 August 2022

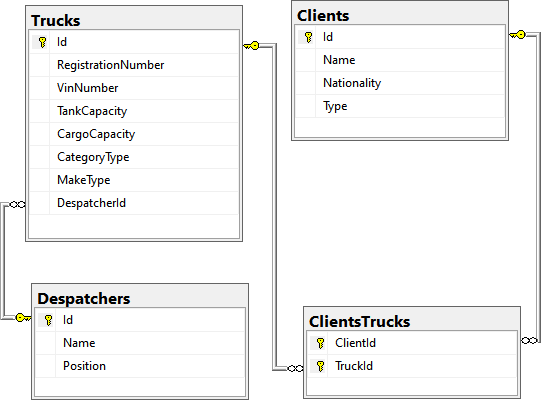
Exam problems for the [Databases Advanced - Entity Framework course @ SoftUni](https://softuni.bg/trainings/3966/entity-framework-core-february-2023).  
Submit your solutions in the **SoftUni Judge** system (delete all **bin**/**obj** and **packages** folders) [here](https://judge.softuni.org/Contests/3934/CSharp-DB-Advanced-Retake-Exam-15-August-2022).

**NOTE: If you want to submit your solution in .NET Core 3.1, please use** [**this link**](https://judge.softuni.org/Contests/3562/CSharp-DB-Advanced-Retake-Exam-15-August-2022) **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.

# Trucks



## Project Skeleton Overview

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

1. Data – contains the TrucksContext 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:

### Truck

Id– integer, **Primary Key**

RegistrationNumber– text with length **8.** First two characters are upper letters [A-Z], followed by four digits and the last two characters are upper letters [A-Z] again.

VinNumber– text with length **17** (**required**)

**TankCapacity** – integer in range [**950…1420**]

**CargoCapacity** – integer in range [**5000…29000**]

CategoryType – enumeration of type CategoryType, with possible values **(**Flatbed, Jumbo, Refrigerated, Semi**)** (**required**)

MakeType– enumeration of type MakeType, with possible values **(**Daf, Man, Mercedes, Scania, Volvo**)** (**required**)

DespatcherId– integer, foreign key (required)

Despatcher– Despatcher

ClientsTrucks– collection of type ClientTruck

### Client

* Id– integer, Primary Key
* Name– text with length **[3, 40]** (**required**)

Nationality – **text** with length **[2, 40]** (**required**)

Type– **text** (**required**)

* ClientsTrucks– collection of type ClientTruck

### Despatcher

* Id– integer, **Primary Key**
* Name– **text** with length **[2, 40]** (**required**)
* Position – **text**
* Trucks– collection of type Truck

### ClientTruck

ClientId– integer, Primary Key, foreign key (required)

Client– Client

TruckId– integer, Primary Key, foreign key (required)

Truck – Truck

## 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 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.

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 Despatchers

Using the file "**despatchers.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 **despatcher** entity (such as invalid **name**), **do not** import any part of the entity and **append an error message** to the **method output**.
* If there is a **null or empty position** for **despatcher** **entity**, **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 **truck** entity (such as invalid **registration number** or **missing VIN number**, **tank capacity** or **weight capacity** is invalid), **do not import it (only the truck itself, not the whole despatcher info)** and **append an error message to the method output**.

|  |
| --- |
| **Success message** |
| Successfully imported despatcher – {**despatcherName**} with {**trucksCount**} trucks. |

##### Example

|  |
| --- |
| **despatchers.xml** |
| <?xml version='1.0' encoding='UTF-8'?>  <Despatchers>  <Despatcher>  <Name>Genadi Petrov</Name>  <Position>Specialist</Position>  <Trucks>  <Truck>  <RegistrationNumber>CB0796TP</RegistrationNumber>  <VinNumber>YS2R4X211D5318181</VinNumber>  <TankCapacity>1000</TankCapacity>  <CargoCapacity>23999</CargoCapacity>  <CategoryType>0</CategoryType>  <MakeType>3</MakeType>  </Truck>  <Truck>  <RegistrationNumber>CB0818TP</RegistrationNumber>  <VinNumber>YS2R4X211D5318128</VinNumber>  <TankCapacity>1400</TankCapacity>  <CargoCapacity>29004</CargoCapacity>  <CategoryType>3</CategoryType>  <MakeType>0</MakeType>  </Truck>  </Trucks>  </Despatcher>  ...  </Despatchers> |
| **Output** |
| Invalid data!  Successfully imported despatcher - Genadi Petrov with 1 trucks.  Invalid data!  ... |

Upon **correct import logic**, you should have imported **30 despatchers** and **65 trucks**.

### JSON Import

#### Import Clients

Using the file **"**clients.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 or invalid **nationality** or type "**usual**"), **do not** import any part of the entity and **append an error message** to the **method output**.
* Take only the unique trucks.
* If a **truck** does **not exist** in the database, **append an error message** to the **method output** and **continue** with the next **truck**.

|  |
| --- |
| **Success message** |
| Successfully imported client - {**clientName**} with {**clientTrucksCount**} trucks. |

##### Example

|  |
| --- |
| **clients.json** |
| [  {  "Name": "Kuenehne + Nagel (AG & Co.) KGKuenehne + Nagel (AG & Co.) KGKuenehne + Nagel (AG & Co.) KG",  "Nationality": "The Netherlands",  "Type": "golden",  "Trucks": [  1,  68,  73,  17,  98,  98  ]  },  {  "Name": "DHL SERVICES LIMITED",  "Nationality": "The United Kingdom",  "Type": "golden",  "Trucks": [  4,  17,  17,  98  ]  }  …  ] |
| **Output** |
| Invalid data!  Invalid data!  Successfully imported client - DHL SERVICES LIMITED with 2 trucks.  **...** |

Upon **correct import logic**, you should have imported **32** **clients** and **113 trucks**.

## Data Export (25 pts)

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

### JSON Export

#### Export Clients With Most Trucks

Select the **top** **10** **clients** that have **at least one truck** that **their tank capacity is bigger or equal** to the **given capacity. Select** them with their **trucks** which meet the **same criteria** (their tank capacity is bigger or equal to the given one). For each **client**, export their **name** and their **trucks.** For each **truck**, export its **registration number**, **VIN** **number**, **tank capacity**, **cargo capacity**, **category** and **make** **type.** Order the **trucks** by **make type (ascending),** then by **cargo capacity** (**descending**). Order the **clients** by **all** **trucks** (**meeting above condition**) **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.ExportClientsWithMostTrucks(context, capacity) |
| [  {  "Name": "Gebr. Mayer GmbH & Co. KG",  "Trucks": [  {  "TruckRegistrationNumber": "CT5206MM",  "VinNumber": "WDB96341311261287",  "TankCapacity": 1420,  "CargoCapacity": 28058,  "CategoryType": "Flatbed",  "MakeType": "Daf"  },  {  "TruckRegistrationNumber": "CT4453MP",  "VinNumber": "WDB96341311269859",  "TankCapacity": 1420,  "CargoCapacity": 28058,  "CategoryType": "Jumbo",  "MakeType": "Man"  },  {  "TruckRegistrationNumber": "CT6631TT",  "VinNumber": "XLRTE47MS1G141929",  "TankCapacity": 1200,  "CargoCapacity": 27303,  "CategoryType": "Refrigerated",  "MakeType": "Scania"  },  {  "TruckRegistrationNumber": "CT5204MM",  "VinNumber": "WDB96341311261293",  "TankCapacity": 1420,  "CargoCapacity": 28058,  "CategoryType": "Jumbo",  "MakeType": "Volvo"  },  {  "TruckRegistrationNumber": "CT2706TT",  "VinNumber": "YS2R4X211D5333237",  "TankCapacity": 1400,  "CargoCapacity": 27000,  "CategoryType": "Flatbed",  "MakeType": "Volvo"  }  ]  }  …  ] |

### XML Export

#### Export Despatchers with Their Trucks

Export all **despatchers** that are managing at least **one** truck. For each **despatcher**, export their **name** and **trucks count**. For each **truck**, export its registration number and **make type.** Order the **trucks** by **registration number** (**ascending**). Order the **despatchers** by **trucks 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.ExportDespatchersWithTheirTrucks(context)** |
| <?xml version="1.0" encoding="utf-16"?>  <Despatchers>  <Despatcher TrucksCount="6">  <DespatcherName>Vladimir Hristov</DespatcherName>  <Trucks>  <Truck>  <RegistrationNumber>CT2462BX</RegistrationNumber>  <Make>Scania</Make>  </Truck>  <Truck>  <RegistrationNumber>CT2699CK</RegistrationNumber>  <Make>Daf</Make>  </Truck>  <Truck>  <RegistrationNumber>CT5203MM</RegistrationNumber>  <Make>Mercedes</Make>  </Truck>  <Truck>  <RegistrationNumber>CT5204MM</RegistrationNumber>  <Make>Volvo</Make>  </Truck>  <Truck>  <RegistrationNumber>CT5205MM</RegistrationNumber>  <Make>Scania</Make>  </Truck>  <Truck>  <RegistrationNumber>CT5206MM</RegistrationNumber>  <Make>Daf</Make>  </Truck>  </Trucks>  </Despatcher>  …  </Despatchers> |