Databases Advanced Retake Exam – 16 December 2021

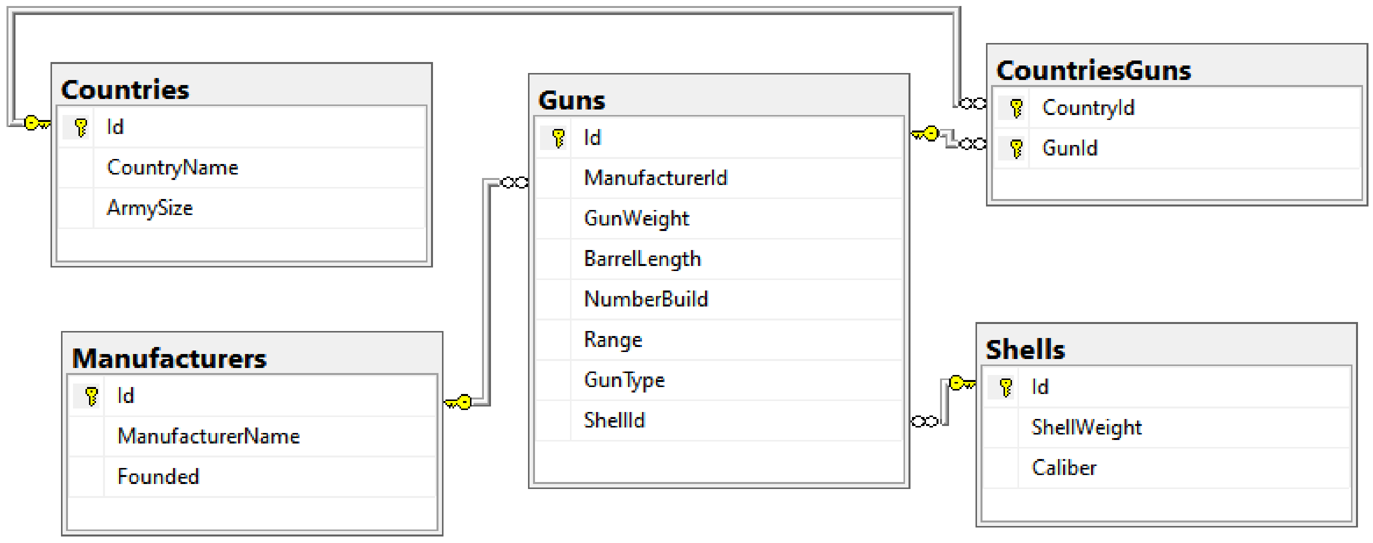
Exam problems for the Databases Advanced - Entity Framework course @ SoftUni.
Submit your solutions in the **SoftUni Judge** system (delete all **bin/obj** and **packages** folders) here.

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.

Artillery



1. Project Skeleton Overview

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

- Data – contains the **ArtilleryContext** 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:

Country

- **Id** – integer, **Primary Key**
- **CountryName** – **text with length [4, 60] (required)**
- **ArmySize** – integer **in the range [50_000….10_000_000] (required)**
- **CountriesGuns** – a collection of **CountryGun**

Manufacturer

- **Id** – integer, **Primary Key**
- **ManufacturerName** – **unique** text with **length [4…40] (required)**
- **Founded** – **text with length [10…100] (required)**
- **Guns** – a collection of **Gun**

Shell

- **Id** – integer, **Primary Key**
- **ShellWeight** – double in range **[2…1_680] (required)**
- **Caliber** – text with length **[4…30] (required)**
- **Guns** – a collection of **Gun**

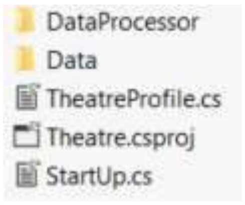
Gun

- **Id** – integer, **Primary Key**
- **ManufacturerId** – **integer, foreign key (required)**
- **GunWeight**– integer **in range [100…1_350_000] (required)**
- **BarrelLength** – **double in range [2.00….35.00] (required)**
- **NumberBuild** – **integer**
- **Range** – integer **in range [1….100_000] (required)**
- **GunType** – **enumeration** of **GunType**, with possible values **(Howitzer, Mortar, FieldGun, AntiAircraftGun, MountainGun, AntiTankGun) (required)**
- **ShellId** – **integer, foreign key (required)**
- **CountriesGuns** – a collection of **CountryGun**

CountryGun

- **CountryId** – **Primary Key integer, foreign key (required)**
- **GunId** – **Primary Key integer, foreign key (required)**

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


DataProcessor
Data
TheatreProfile.cs
Theatre.csproj
StartUp.cs

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.

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 Countries

Using the file "**`countries.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 unvalid **country name or army size, do not** import any part of the entity and **append an error message "`Invalid data.`"** to the **method output**.

Success message
`Successfully import {countryName} with {armySize} army personnel.`

Example

countries.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<Countries>
  <Country>
    <CountryName>Afghanistan</CountryName>
    <ArmySize>1697064</ArmySize>
  </Country>
  <Country>
    <CountryName>Afghan</CountryName>
    <ArmySize>16</ArmySize>
  </Country>
  <Country>
    <CountryName>Albania</CountryName>
    <ArmySize>6296389</ArmySize>
  </Country>
  <Country>
    <CountryName></CountryName>
    <ArmySize>2401223</ArmySize>
  </Country>
  <Country>
    <CountryName>Algeria</CountryName>
    <ArmySize>1284683</ArmySize>
  </Country>
…
</Countries>
```

Output

```
Successfully import Afghanistan with 1697064 army personnel.
Invalid data.
Successfully import Albania with 6296389 army personnel.
Invalid data.
```

```
Successfully import Algeria with 1284683 army personnel.
...
```

Upon **correct import logic**, you should have imported **88 countries**.

Import Manufacturers

Using the file "**manufacturers.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 **manufacturer name or founded**, **do not** import any part of the entity and **append an error message "Invalid data."** to the **method output**.

The **Founded** entity will be separated by comma and space **", "**.

Success message
`Successfully import manufacturer {manufacturerName} founded in {townName, countryName}.`

Example

manufacturers.xml

```xml
<?xml version='1.0' encoding='UTF-8'?>
<Manufacturers>
  <Manufacturer>
    <ManufacturerName>BAE Systems</ManufacturerName>
    <Founded>30 November 1999, London, England</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>BAE</ManufacturerName>
    <Founded>30 November 1999, London, England</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>Aviation Industry Corporation of China</ManufacturerName>
    <Founded>April 1, 1951, Chaoyang District, Beijing, China</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>General Dynamics</ManufacturerName>
    <Founded>February 7, 1899, Reston, Virginia, United States</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>General Dynamics</ManufacturerName>
    <Founded>February 7, 1899, Reston, Virginia, United States</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>Raytheon Technologies</ManufacturerName>
    <Founded>2020, Waltham, Massachusetts, United States</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>Northrop Grumman</ManufacturerName>
    <Founded>1994, 2980 Fairview Park Drive, West Falls Church, Virginia, United
States</Founded>
  </Manufacturer>
  <Manufacturer>
    <ManufacturerName>Lockheed Martin</ManufacturerName>
    <Founded>March 15, 1995, Bethesda, Maryland, United States</Founded>
  </Manufacturer>
…
</Manufacturers>
```

Output

```
Successfully import manufacturer BAE Systems founded in London, England.
Invalid data.
Successfully import manufacturer Aviation Industry Corporation of China founded in
Beijing, China.
Successfully import manufacturer General Dynamics founded in Virginia, United
States.
Invalid data.
Successfully import manufacturer Raytheon Technologies founded in Massachusetts,
United States.
Successfully import manufacturer Northrop Grumman founded in Virginia, United
States.
Successfully import manufacturer Lockheed Martin founded in Maryland, United States.
...
```

Upon **correct import logic**, you should have imported **20 unique** `manufacturers`.

Import Shells

Using the file "**`shells.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: **shell weight or caliber**, **do not** import any part of the entity and **append an error message "`Invalid data.`"** to the **method output**.

Success message

```
Successfully import shell caliber #{caliber} weight {shellWeigh} kg.
```

Example

`shells.xml`

```xml
<?xml version='1.0' encoding='UTF-8'?>
<Shells>
  <Shell>
    <ShellWeight>50</ShellWeight>
    <Caliber>155 mm (6.1 in)</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>100</ShellWeight>
    <Caliber>103 mm (8 in)</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>146</ShellWeight>
    <Caliber>203 mm (8 in)</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>0</ShellWeight>
    <Caliber>280 mm</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>300</ShellWeight>
    <Caliber>280 mm (11 in)</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>460</ShellWeight>
    <Caliber/>
  </Shell>
  <Shell>
    <ShellWeight>1500</ShellWeight>
    <Caliber>460 mm (18 in)</Caliber>
  </Shell>
  <Shell>
    <ShellWeight>53</ShellWeight>
    <Caliber>155mm</Caliber>
  </Shell>
…
</Shells>
```

Output

```
Successfully import shell caliber #155 mm (6.1 in) weight 50 kg.
Successfully import shell caliber #103 mm (8 in) weight 100 kg.
Successfully import shell caliber #203 mm (8 in) weight 146 kg.
Invalid data.
Successfully import shell caliber #280 mm (11 in) weight 300 kg.
Invalid data.
Successfully import shell caliber #460 mm (18 in) weight 1500 kg.
Successfully import shell caliber #155mm weight 53 kg.
...
```

Upon **correct import logic**, you should have imported **60 shells**.

JSON Import

Import Guns

Using the file "**guns.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 (such as invalid **gun weight, barrel length, range, gun-type**), **do not import any part of the entity** and **append an error message to the method output**.
- The **Countries** array will always contain valid ids.

Success message

```
Successfully import gun {gunType} with a total weight of {gunWeight} kg. and barrel
length of {barrelLength} m.
```

Example

guns.json

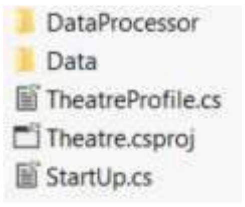
```
[
  {
    "ManufacturerId": 14,
    "GunWeight": 531616,
    "BarrelLength": 6.86,
    "NumberBuild": 287,
    "Range": 120000,
    "GunType": "Howitzer",
    "ShellId": 41,
    "Countries": [
      { "Id": 86 },
      { "Id": 57 },
      { "Id": 64 },
      { "Id": 74 },
      { "Id": 58 }
    ]
  },
  {
    "ManufacturerId": 8,
    "GunWeight": 801684,
    "BarrelLength": 31.18,
    "NumberBuild": 620,
    "Range": 19118,
    "GunType": "AntiTankGun",
    "ShellId": 38,
    "Countries": [
      { "Id": 47 },
      { "Id": 3 },
      { "Id": 85 },
      { "Id": 35 },
      { "Id": 49 },
      { "Id": 53 },
      { "Id": 30 },
      { "Id": 39 },
      { "Id": 62 },
      { "Id": 6 },
      { "Id": 76 },
      { "Id": 78 },
      { "Id": 43 },
      { "Id": 72 },
      { "Id": 23 },
      { "Id": 9 },
      { "Id": 1 },
      { "Id": 21 },
      { "Id": 8 },
      { "Id": 67 },
      { "Id": 2 },
      { "Id": 33 },
      { "Id": 28 },
      { "Id": 17 },
      { "Id": 54 },
      { "Id": 4 }
    ]
  }
…
]
```

Output

```
Invalid data.
Successfully import gun AntiTankGun with a total weight of 801684 kg. and barrel
length of 31.18 m.
...
```

Upon **correct import logic**, you should have imported **138 guns and 785 countries' guns**.

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


DataProcessor
Data
TheatreProfile.cs
Theatre.csproj
StartUp.cs

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 Shells

The given method in the project's skeleton receives a double representing the shell weight. Export **all shells** which weights more than the given and **the guns which use this shell**. For each **Shell**, export its **`ShellWeight`**, **`Caliber`**, and **`Guns`**. Export only the **guns** which are **`AntiAircraftGun`** gun type. For every gun export **`GunType`**, **`GunWeight`**, **`BarrelLength`**, and **`Range`** (if the **range is bigger than `3000`**, export "**`Long-range`**", otherwise export "**`Regular range`**"). Order the guns by **`GunWeight`** (**descending**). Order the shells by **`ShellWeight`** (**ascending**).

Example

Serializer. ExportShells(context, shellWeight)

```
[
  {
    "ShellWeight": 124.0,
    "Caliber": "155 mm HE ERFB RA-BB",
    "Guns": [
      {
        "GunType": "AntiAircraftGun",
        "GunWeight": 250138,
        "BarrelLength": 6.55,
        "Range": "Long-range"
      }
    ]
  },
  {
    "ShellWeight": 146.0,
    "Caliber": "203 mm (8 in)",
    "Guns": []
  },
...
]
```

XML Export

Export Guns

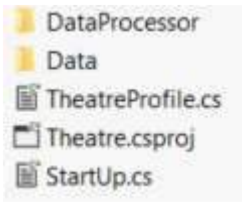
Use the method provided in the project skeleton, which receives a **manufacturer**. Export all guns with a manufacturer equal to the given. For each **gun**, export **`Manufacturer`**, **`GunType`**, **`BarrelLength`**, **`GunWeight`**, **`Range`**, and **`Countries`** that use this gun. Select only the **`Countries`** which has **`ArmySize`** `bigger than` **`4500000`**. For each country export **`CountryName`** and **`ArmySize`**. Order the countries by **army size (ascending).** Order **guns by BarrelLength (ascending).**

Example

Serializer.ExportGuns(context, manufacturer)

```xml
<?xml version="1.0" encoding="utf-16"?>
<Guns>
  <Gun Manufacturer="Krupp" GunType="Mortar" GunWeight="1291272" BarrelLength="8.31"
Range="14258">
    <Countries>
      <Country Country="Sweden" ArmySize="5437337" />
      <Country Country="Portugal" ArmySize="9523599" />
    </Countries>
  </Gun>
  <Gun Manufacturer="Krupp" GunType="AntiAircraftGun" GunWeight="1280923"
BarrelLength="10.89" Range="16530">
    <Countries>
      <Country Country="Albania" ArmySize="6296389" />
      <Country Country="United Kingdom" ArmySize="7242451" />
      <Country Country="China" ArmySize="9944746" />
    </Countries>
  </Gun>
  <Gun Manufacturer="Krupp" GunType="Howitzer" GunWeight="656499" BarrelLength="13.04"
Range="80235">
    <Countries>
      <Country Country="Malta" ArmySize="8507869" />
    </Countries>
  </Gun>
  <Gun Manufacturer="Krupp" GunType="FieldGun" GunWeight="431716" BarrelLength="15.7"
Range="28309">
    <Countries>
      <Country Country="Cape Verde" ArmySize="7704194" />
      <Country Country="Equatorial Guinea" ArmySize="9751317" />
    </Countries>
  </Gun>
  <Gun Manufacturer="Krupp" GunType="Mortar" GunWeight="388420" BarrelLength="15.87"
Range="6288">
    <Countries>
      <Country Country="Norway" ArmySize="6282380" />
      <Country Country="Myanmar" ArmySize="9883310" />
    </Countries>
…
</Guns>
```

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


DataProcessor
Data
TheatreProfile.cs
Theatre.csproj
StartUp.cs