Lab: ORM and Entity Framework

Problems for exercises and homework for the "Software Technologies Back End" course from the official "Applied Programmer" curriculum.

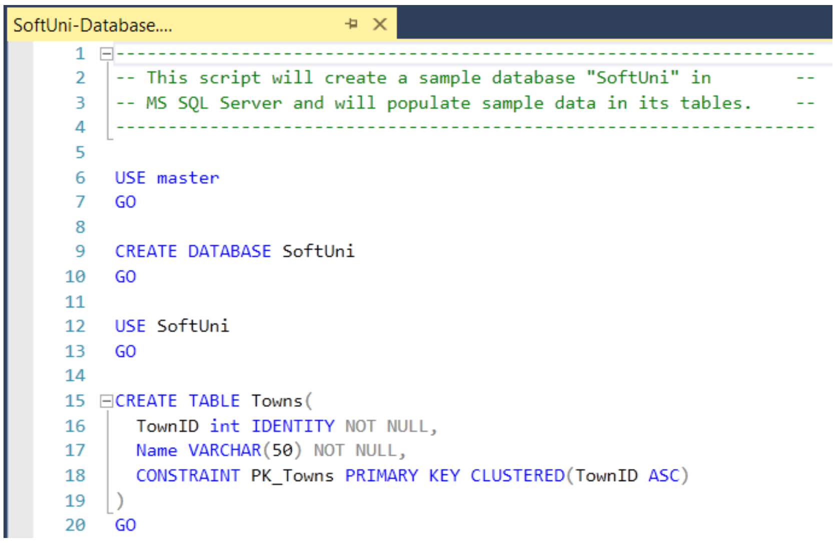
You can check your solutions here: https://judge.softuni.bg/Contests/2800/ORM-and-Entity-Framework-Lab

(delete all "**bin**"/"**obj**" folders)

Use the provided skeleton from resources! Do not change its methods, classes and namespaces!

1. Import the SoftUni Database

Import the SoftUni DB into SQL Management Studio (if not yet imported) by **executing** the provided **.sql** script.

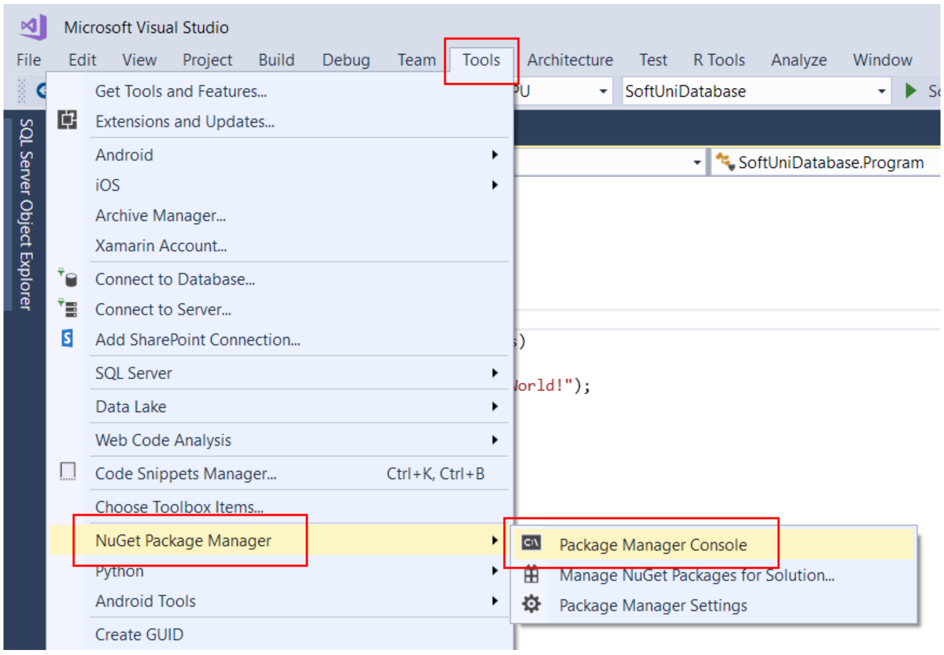


```
----------------------------------------------------------------------
-- This script will create a sample database "SoftUni" in          --
-- MS SQL Server and will populate sample data in its tables.      --
----------------------------------------------------------------------

USE master
GO

CREATE DATABASE SoftUni
GO

USE SoftUni
GO

CREATE TABLE Towns(
  TownID int IDENTITY NOT NULL,
  Name VARCHAR(50) NOT NULL,
  CONSTRAINT PK_Towns PRIMARY KEY CLUSTERED(TownID ASC)
)
GO
```

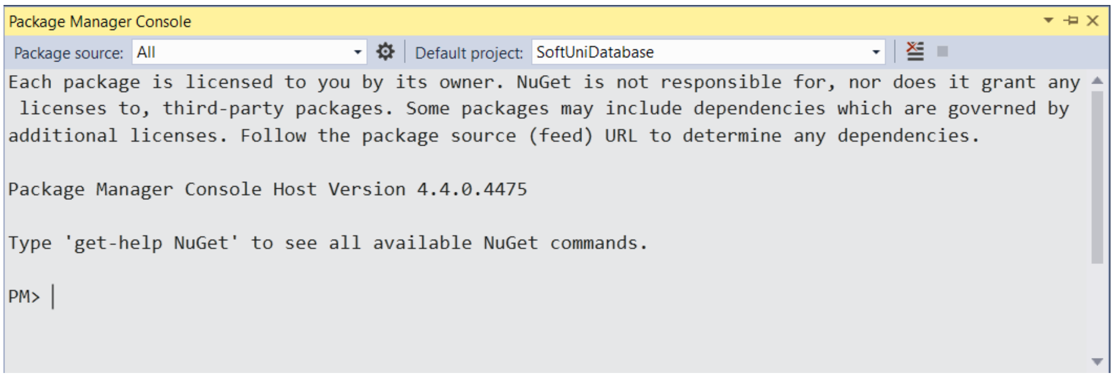
2. Generate Database First ORM Model

Model the existing database by using "**Database First**" in Entity Framework Core.

First create a new empty **.Net Core Console Application** and after it is created open the **Package Manager Console**:



It will look something like this:



Package Manager Console

Package source: All | Default project: SoftUniDatabase

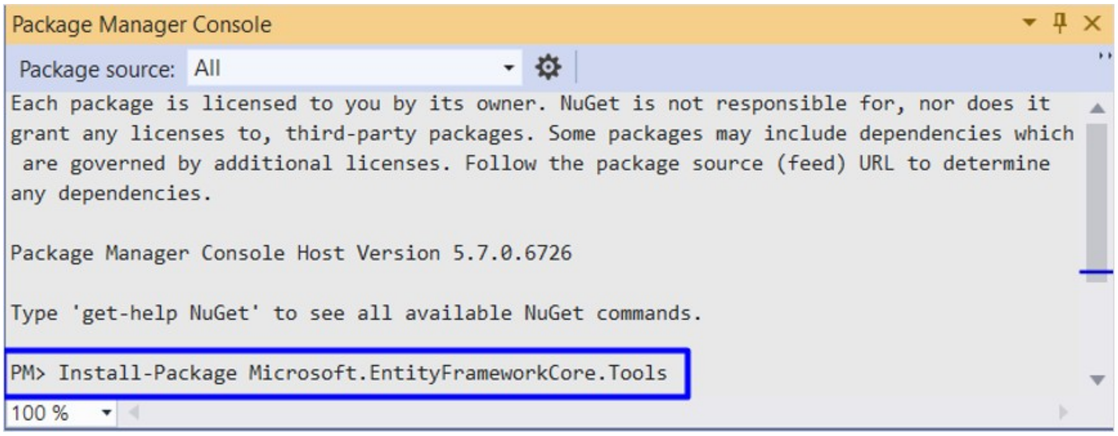
```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any
 licenses to, third-party packages. Some packages may include dependencies which are governed by
additional licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 4.4.0.4475

Type 'get-help NuGet' to see all available NuGet commands.

PM>
```

Use it to run the following commands **one by one**:

```
Install-Package Microsoft.EntityFrameworkCore.Tools
```

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design
```



Package Manager Console

Package source: All

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it
grant any licenses to, third-party packages. Some packages may include dependencies which
 are governed by additional licenses. Follow the package source (feed) URL to determine
any dependencies.

Package Manager Console Host Version 5.7.0.6726

Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package Microsoft.EntityFrameworkCore.Tools
```

These are the **packages** you will need, in order to **scaffold** our **SoftUniContext** from the **SoftUni database**.

Next, we must **execute** the **command** to **scaffold** our **context class**. It will consist of 4 things:

- First, the name of the command:

```
Scaffold-DbContext
```

- Second, the connection we will be using (our connection string):

```
-Connection "Server=<ServerName>;Database=<DatabaseName>;Integrated
Security=True;"
```

For **ServerName**, use the name of your local MS SQL Server instance or "**.**".
For **DatabaseName**, use the name of the database you want to use, in this case – **SoftUni**.

- Third, we need to declare our service provider, we'll be using **Microsoft.EntityFrameworkCore.SqlServer**:

```
-Provider Microsoft.EntityFrameworkCore.SqlServer
```

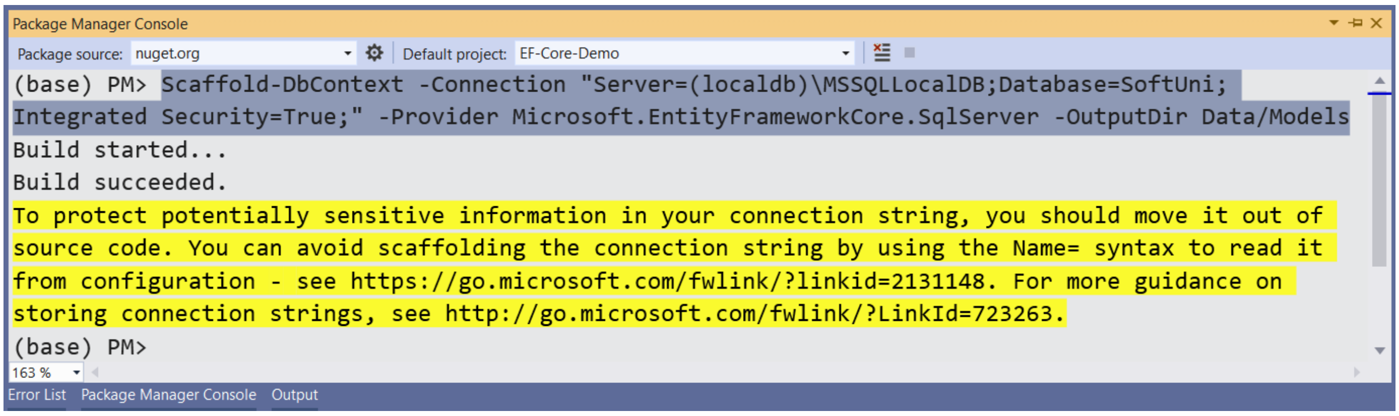
- And the fourth thing we'll do, is to give it a directory where all of our models will go (e.g. **Models**):

```
-OutputDir Data/Models
```

Our final command will look like this:

```
Scaffold-DbContext -Connection "Server=(localdb)\
MSSQLLocalDB;Database=SoftUni;Integrated Security=True;" -Provider
```

```
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Data/Models
```

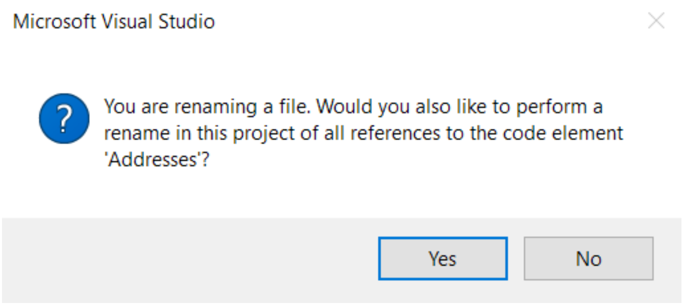


```
(base) PM> Scaffold-DbContext -Connection "Server=(localdb)\MSSQLLocalDB;Database=SoftUni;
Integrated Security=True;" -Provider Microsoft.EntityFrameworkCore.SqlServer -OutputDir Data/Models
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out of
source code. You can avoid scaffolding the connection string by using the Name= syntax to read it
from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on
storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
(base) PM>
```

Execute the **whole command** on a **single line**.

Entity Framework Core has successfully **mapped the database schema to C# classes**. However, it isn't good enough with names – all classes have been **pluralized**.

- Use the **Solution Explorer** in Visual Studio to move the **SoftUniContext** class out of **Models** into the **Data** folder and rename all of our classes properly.
- Use **right click → [Rename]** or the [**F2**] shortcut and press [**OK**] on this **pop up window** after each class:

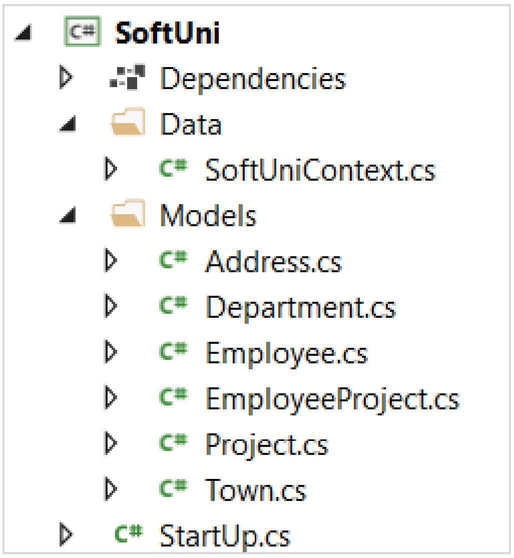


Microsoft Visual Studio

You are renaming a file. Would you also like to perform a rename in this project of all references to the code element 'Addresses'?

This way Visual Studio will also **rename** the **classes everywhere** they’re used.

The final result should look like this:



Don’t forget to fix the **SoftUniContext’s** namespace after moving it and add a reference to the **Models** namespace:

Make sure that your namespaces are **exactly** the same as these:

```
SoftUni
```

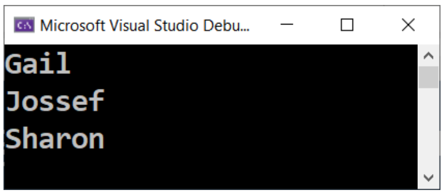
```
SoftUni.Data

SoftUni.Models
```

Finally, we can clean up the packages we won’t be using anymore from the package manager GUI or by running these commands:

```
Uninstall-Package Microsoft.EntityFrameworkCore.Tools -r

Uninstall-Package Microsoft.EntityFrameworkCore.SqlServer.Design -
RemoveDependencies
```

3. Find Employees with Job Title

Create a method `public static string FindEmployeesWithJobTitle(SoftUniContext context)` to print the **First Name** of all employees with **Job Title** equal to **“Design Engineer”**.



```
Gail
Jossef
Sharon
```

Solution

First, use the **`context`** in the method like this:

```
public static string FindEmployeesWithJobTitle(SoftUniContext context)
{

}
```

Get all employees and **filter** them using **`context.Employees`**. Then, select only the **First Name** of each employee and use **String.Join()** to return the array of names as a string to the method.

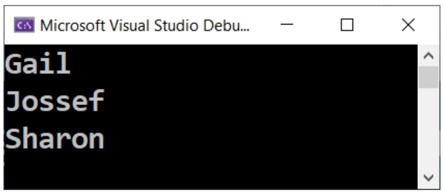
```
public static string FindEmployeesWithJobTitle(SoftUniContext context)
{
    var employees = context.Employees
        .Where(e => e.JobTitle == "Design Engineer")
        .Select(x => x.FirstName)
        .ToList();

    return string.Join(Environment.NewLine, employees);
}
```

Run Your Code in the Console

Invoke the **`FindEmployeesWithJobTitle(SoftUniContext context)`** method from the application entry point **`Main()`**:

```
static void Main()
{
    var context = new SoftUniContext();
    var result = FindEmployeesWithJobTitle(context);
    Console.WriteLine(result);
}
```

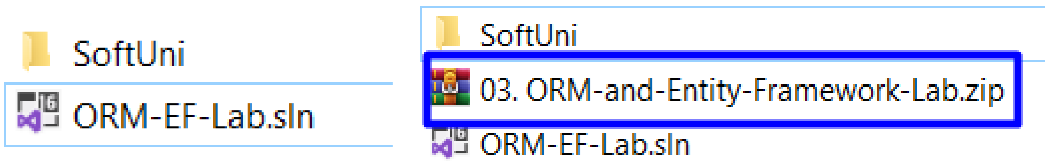
Press **[Ctrl+F5]** to run the application. Check if the result on the console is correct:



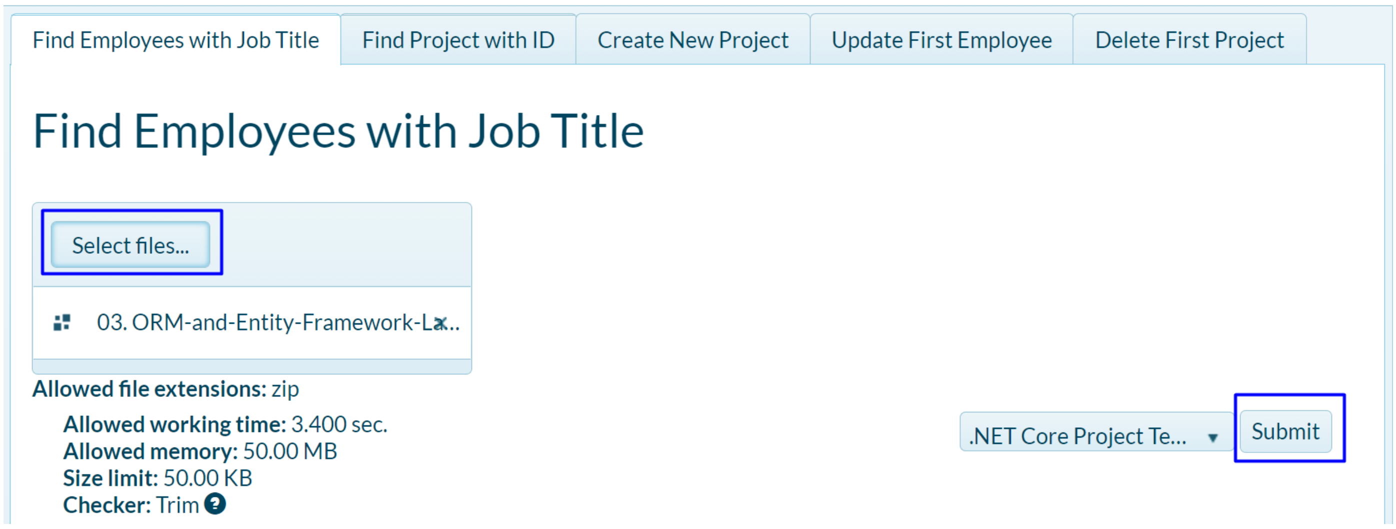
```
Gail
Jossef
Sharon
```

Submit Your Code to Judge

Save your files in Visual Studio. Delete the **"bin"** and **"obj"** folders from the **SoftUni** folder and create a **ZIP archive** of your solution:

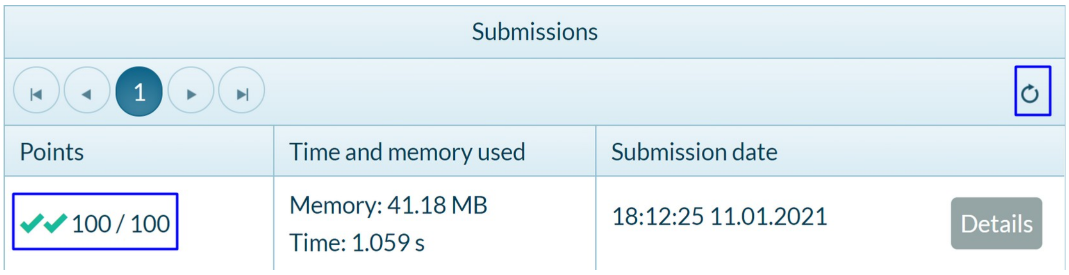


Submit the ZIP file in Judge:



Allowed file extensions: zip
Allowed working time: 3.400 sec.
Allowed memory: 50.00 MB
Size limit: 50.00 KB
Checker: Trim

You should get 100 / 100 score:

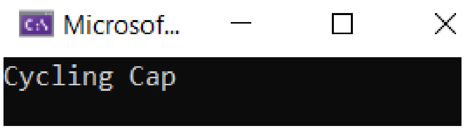


Submissions

Points	Time and memory used	Submission date
100 / 100	Memory: 41.18 MB Time: 1.059 s	18:12:25 11.01.2021

4. Find Project with ID

Again, use the **`context`** and get all **Projects** from it. Use **`.Find()`** method to find the project with **ID 2** and return the **Name** of the project.



```
Cycling Cap
```

Solution

```
public static string FindProjectWithId(SoftUniContext context)
{
    var project = context.Projects.Find(2);
    return project.Name;
}
```

5. Create New Project

Your task is to create a **new Project** in the **Projects** table.

Solution

To create a new database **row** use the **`.Add()`** method of the corresponding **`DbSet`**. First, create a new **Project object** and give values to **Name** and **StartDate** properties.

```
public static void CreateNewProject(SoftUniContext context)
{
    var project = new Project()
    {
        Name = "Our Newest Project",
        StartDate = new DateTime(2021, 1, 1),
    };
}
```

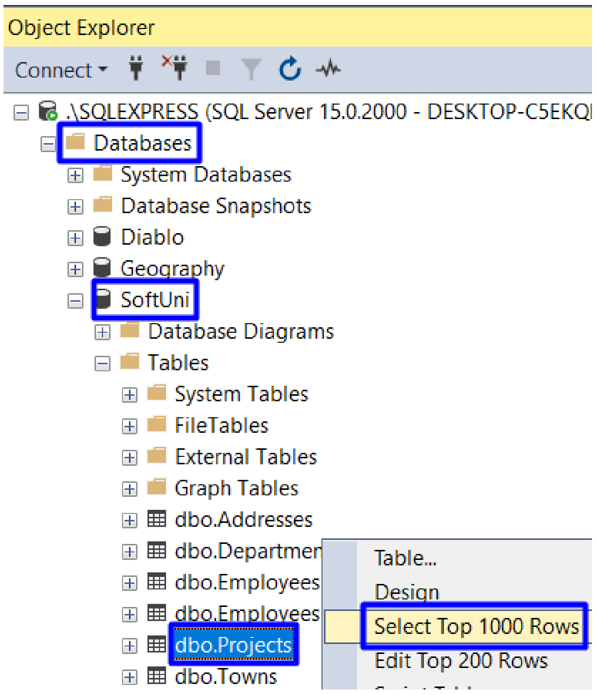
Then, add the object to the **DbSet** and do not forget to **save changes** the following way:

```
public static void CreateNewProject(SoftUniContext context)
{
    var project = new Project()
    {
        Name = "Our Newest Project",
        StartDate = new DateTime(2021, 1, 1),
    };
    context.Projects.Add(project);
    context.SaveChanges();
}
```

Run the app. There is no result displayed on the console.

Check the Result in the DB

In order to check the result, go to **`SQL Server Management Studio -> Object Explorer -> Databases -> SoftUni -> dbo.Projects`**. **Right-click** on it and choose **`Select Top 1000 Rows`**.



Scroll down to the **last entity**. It should be the one we added using a C# command in Visual Studio.

ProjectID	Name	Description	StartDate	EndDate
128	Judge System	NULL	2015-04-15 00:00:00	NULL

6. Update First Employee

Get the **first employee** using `.FirstOrDefault()` method and change their **First Name** to **“Alex”**. Do not forget to **save changes**! In case there are no employees, return **empty string** to the method, else return the changed employee’s first name.

Solution

```
public static string UpdateFirstEmployee(SoftUniContext context)
{
    Employee employee = context.Employees.FirstOrDefault();
    if (employee != null)
    {
        employee.FirstName = "Alex";
        context.SaveChanges();
        return employee.FirstName;
    }
    return "";
}
```

Check result in the DB

This is the database entity **before** the code execution:

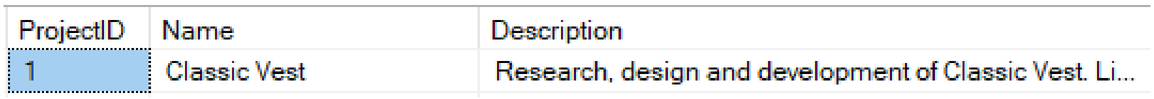
EmployeeID	FirstName	LastName	MiddleName	JobTitle
1	Guy	Gilbert	R	Production Technician

After the code execution, the entity **should be changed**:

EmployeeID	FirstName	LastName	MiddleName	JobTitle
1	Alex	Gilbert	R	Production Technician

7. Delete First Project

Get the **first project** and delete it using the **`.Remove()`** method. Do not forget to **save changes**! The entity we should remove is the following:

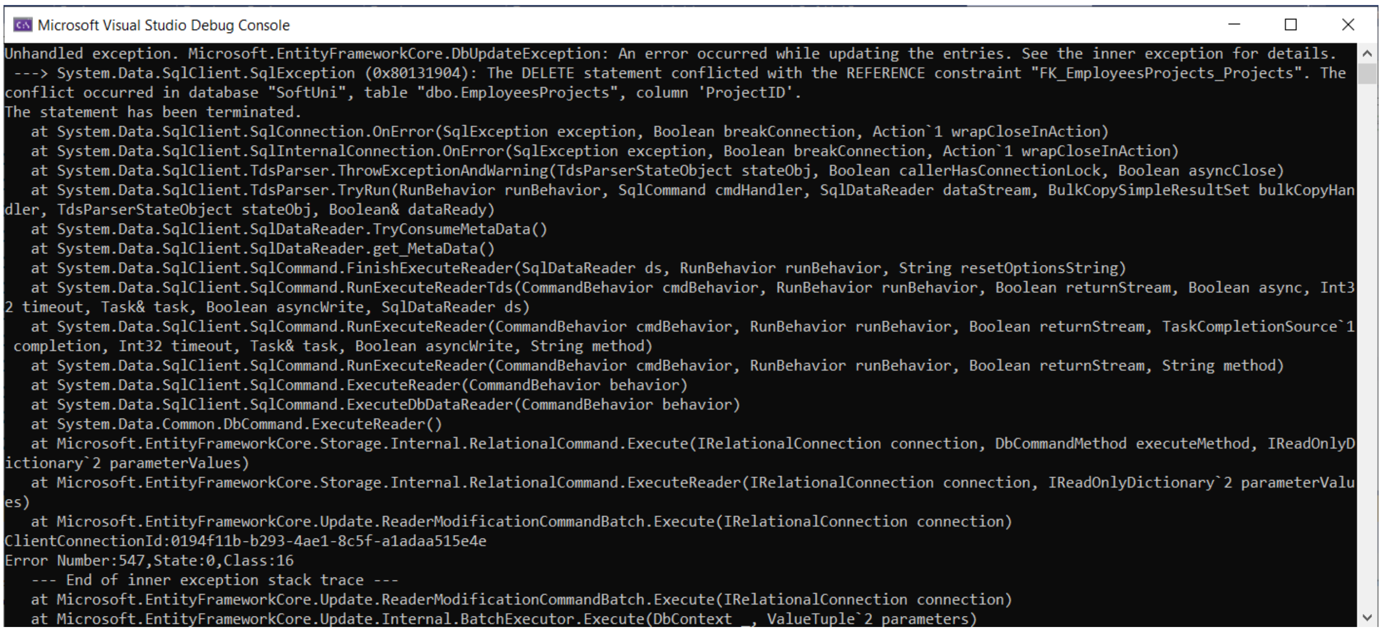


ProjectID	Name	Description
1	Classic Vest	Research, design and development of Classic Vest. Li...

Solution

```
public static string DeleteFirstProject(SoftUniContext context)
{
    Project project = context.Projects.FirstOrDefault();
    context.Projects.Remove(project);
    context.SaveChanges();
    return project.Name;
}
```

However, when the program is executed an **error message** appears.



Microsoft Visual Studio Debug Console

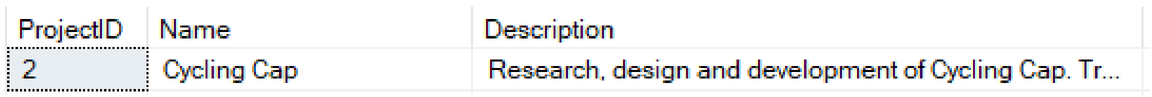
```
Unhandled exception. Microsoft.EntityFrameworkCore.DbUpdateException: An error occurred while updating the entries. See the inner exception for details.
 ---> System.Data.SqlClient.SqlException (0x80131904): The DELETE statement conflicted with the REFERENCE constraint "FK_EmployeesProjects_Projects". The
conflict occurred in database "SoftUni", table "dbo.EmployeesProjects", column 'ProjectID'.
The statement has been terminated.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHan
dler, TdsParserStateObject stateObj, Boolean& dataReady)
   at System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
   at System.Data.SqlClient.SqlDataReader.get_MetaData()
   at System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString)
   at System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int3
2 timeout, Task& task, Boolean asyncWrite, SqlDataReader ds)
   at System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, TaskCompletionSource`1
 completion, Int32 timeout, Task& task, Boolean asyncWrite, String method)
   at System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
   at System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior)
   at System.Data.SqlClient.SqlCommand.ExecuteDbDataReader(CommandBehavior behavior)
   at System.Data.Common.DbCommand.ExecuteReader()
   at Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand.Execute(IRelationalConnection connection, DbCommandMethod executeMethod, IReadOnlyD
ictionary`2 parameterValues)
   at Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand.ExecuteReader(IRelationalConnection connection, IReadOnlyDictionary`2 parameterValu
es)
   at Microsoft.EntityFrameworkCore.Update.ReaderModificationCommandBatch.Execute(IRelationalConnection connection)
ClientConnectionId:0194f11b-b293-4ae1-8c5f-a1adaa515e4e
Error Number:547,State:0,Class:16
   --- End of inner exception stack trace ---
   at Microsoft.EntityFrameworkCore.Update.ReaderModificationCommandBatch.Execute(IRelationalConnection connection)
   at Microsoft.EntityFrameworkCore.Update.Internal.BatchExecutor.Execute(DbContext _, ValueTuple`2 parameters)
```

The reason for the error is that the **EmployeesProjects table** in the SoftUni DB contains a **ProjectID column**. So, entities from the Projects table **cannot be deleted** that way because some entities in the EmployeesProjects table contain the id of the project entity we want to delete. To solve that issue we may **first delete all enities** from the **EmployeesProjects** table, which contain our **ProjectId** (in our case with ProjectId=**1**). The command is the following:

```
public static string DeleteFirstProject(SoftUniContext context)
{
    Project project = context.Projects.FirstOrDefault();
    var entitiesWithProject = context.EmployeesProjects
        .Where(x => x.ProjectId == project.ProjectId).ToList();
    context.EmployeesProjects.RemoveRange(entitiesWithProject);
    context.Projects.Remove(project);
    context.SaveChanges();
    return project.Name;
}
```

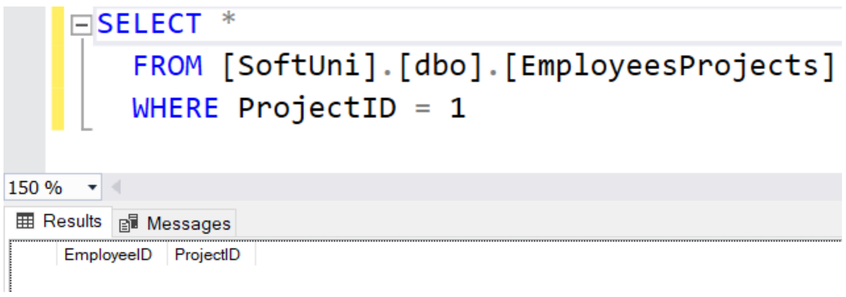
Check result in the DB

Execute the program and see the result in the **Projects** table in the SoftUni DB.



ProjectID	Name	Description
2	Cycling Cap	Research, design and development of Cycling Cap. Tr...

You can also check the **EmployeesProjects** table. Now it does not contain entities with **ProjectId = 1**.

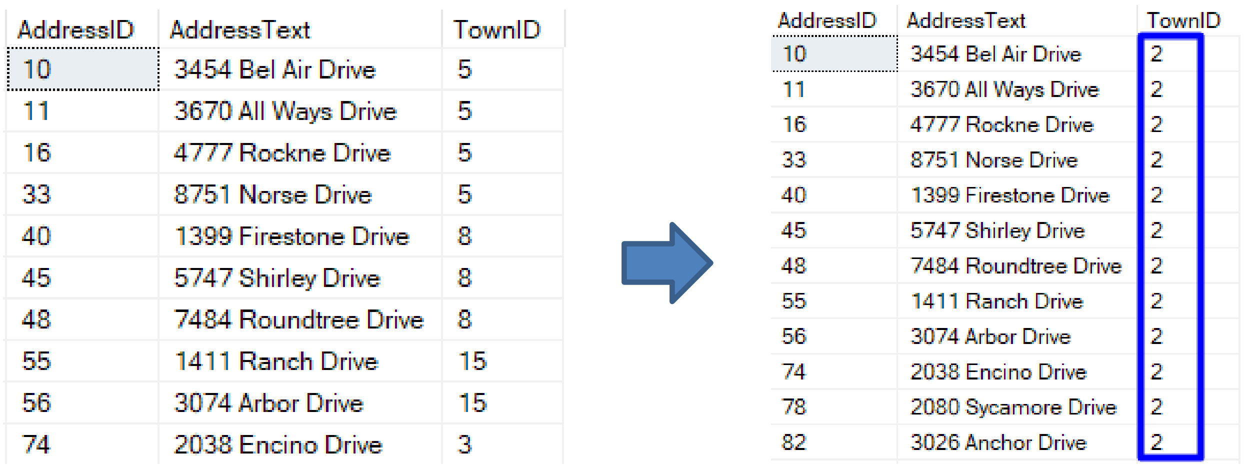


```
SELECT *
  FROM [SoftUni].[dbo].[EmployeesProjects]
  WHERE ProjectID = 1
```

EmployeeID	ProjectID

8. Update Addresses

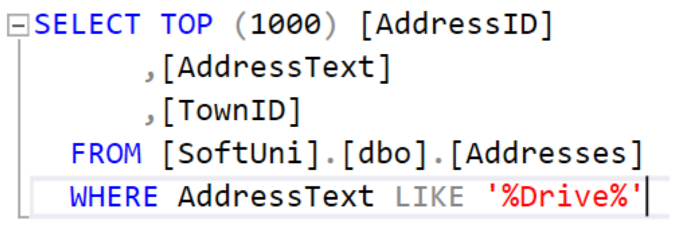
Write a **method** to update **TownId** to **2** for all **Addresses** with **AddressText**, containing the word **"Drive"**.



AddressID	AddressText	TownID
10	3454 Bel Air Drive	5
11	3670 All Ways Drive	5
16	4777 Rockne Drive	5
33	8751 Norse Drive	5
40	1399 Firestone Drive	8
45	5747 Shirley Drive	8
48	7484 Roundtree Drive	8
55	1411 Ranch Drive	15
56	3074 Arbor Drive	15
74	2038 Encino Drive	3

AddressID	AddressText	TownID
10	3454 Bel Air Drive	2
11	3670 All Ways Drive	2
16	4777 Rockne Drive	2
33	8751 Norse Drive	2
40	1399 Firestone Drive	2
45	5747 Shirley Drive	2
48	7484 Roundtree Drive	2
55	1411 Ranch Drive	2
56	3074 Arbor Drive	2
74	2038 Encino Drive	2
78	2080 Sycamore Drive	2
82	3026 Anchor Drive	2

You can check the result in the **SoftUni DB** with this command:



```
SELECT TOP (1000) [AddressID]
      ,[AddressText]
      ,[TownID]
  FROM [SoftUni].[dbo].[Addresses]
  WHERE AddressText LIKE '%Drive%'
```

The method `UpdateAddresses(SoftUniContext context)` should return the **count** of **changed** addresses, converted to **string**.

```
public static string UpdateAddresses(SoftUniContext context)
{
    return addresses.Count.ToString();
}
```