Introduction to SQL Server and LINQ

This exercise introduces you to LINQ (Language Integrated Query).

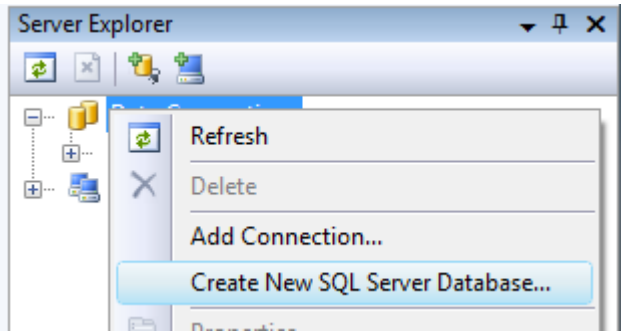
For this exercise we you will need

- Windows 2008 Server, Windows 2008 Server R2, Windows Vista or Windows 7
- Visual Studio 2008 or Visual Studio 2010
- SQL Server 2008 or newer

You can either install the software on your own computer (faster, more responsive) or use Windows 2008 R2 server `sandstorm.cs.ut.ee`. You can create similar solution using Mono Framework and Mono Developer, however, this tutorial has instructions for Visual Studio only.

It is suggested you try out both option 1 and option 2 for database creation. Due to limited permissions, you can not create a new database at SANDSTOR. Thus you need to follow option 3 on SANDSTOR.

Option 1 - Creating a new database on server



- Open Visual Studio and open Server Explorer.
- Right click on Data Connections and choose *Create New SQL Server Database…*.
- Give the database a name (prefix it with your username) and create it.
- Create a new table called **Customer**.
- Add two columns:
  - **Name** of type **nvarchar(50)** required (**not null**).
  - **Street** of type **nvarchar(100)** required (**not null**).
- Save changes and open table data view.
- Add a few entries to the table and save changes.

Option 2 – Creating a new local database

- Open Visual Studio and open Server Explorer.
- Right click on Data Connections and choose *Add Connection…* (provider: **Microsoft SQL Server Compact 3.5**).
- Choose *New database…* as database file. Choose whatever setting you like in the next dialog (just make sure you remember the password and location). Local databases usually have sdf file extension.
- Create a new table called **Customer**.
- Add two columns:
  - **Name** of type **nvarchar(50)** required (**not null**).
  - **Street** of type **nvarchar(100)** required (**not null**).
- Save changes and open table data view.

Add a few entries to the table and save changes.

Option 3 – Connecting to a database

- Open Visual Studio and open Server Explorer.
- Right click on Data Connections and choose *Add Connection….*
- Choose *SANDSTOR\ESIINSTANCE* as server name and *usernameDB* as database name where username is your UT username.
- Create a new table called **Customer**.
- Add two columns:
  - **Name** of type **nvarchar(50)** required (**not null**).
  - **Street** of type **nvarchar(100)** required (**not null**).
- Save changes and open table data view.

Add a few entries to the table and save changes.

Part A – Creating a simple console project

- Create a new *Console Application* project.
- Add a new *ADO.NET Entity Data Model* item to the project.
  - Choose to **generate model from database**.
- Edit the application main method:
  - Create new entities object.
    - The type of the object is defined in the model's property **Entity Container Name**. You can access the models properties by right-clicking on the blank space in the model editor.
    - The entities object is in your project namespace.
  - Print line to display the table header (Name, Street).
  - Loop through the Customers table and print it out.
    - The table can be accessed through database entities objects property **Customer**.
    - You can use `foreach` loop to loop through the rows (see below for an example).

➢ Run and test the application.

The resulting Main method should look similar to that in C#:

```
Database1Entities1 cdncdc = new Database1Entities1 ();
Console.WriteLine("Name \tStreet");
foreach (Customer c in cdncdc.Customers)
{
      Console.WriteLine(c.Name + "\t" + c.Street);
}
Console.ReadKey();
```

The resulting Main method should look similar to that in VB:

```
Dim de As Database1Entities1
de = New Database1Entities1()
System.Console.WriteLine("Name\tStreet")
For Each c As Customer In de.Customer
      System.Console.WriteLine("{0}\t{1}", c.Name, c.Street)
Next
System.Console.ReadKey()
```

Part B – Writing LINQ query

➢ Edit the Main method by adding code to it:

- Create a new LINQ query that returns all streets that have the string "Street" in it. This LINQ query should have the following parts (Tip! Make good use of auto-complete!):
  - `From` clause specifying the Customers table as the data source (just like we used it in the `foreach` clause).
  - `Where` clause specifying the condition of including the result. Note that you can use any C# expression on table. You can use the Contains method of String class to check for the presence on "Street" string.
  - `Select` clause specifying you only want streets to be returned.
- Write the result to the console.

The code added to the Main method should look similar to that in C#:

```
var streetsQuery = from cust in cdncdc.Customers
      where cust.Street.Contains("Street")
      select cust.Street;
Console.WriteLine("Street");
foreach (string street in streetsQuery)
{
      Console.WriteLine(street);
}
```

The code added to the Main method should look similar to that in VB:

```
Dim ss = From c In de.Customer
       Where c.Street.Contains("Street")
       Select c.Street
System.Console.WriteLine("Streets with Street:")
For Each c As String In ss
       System.Console.WriteLine("{0}", c)
Next
```

- Run and test the application.

NB! The following exercises need SQL Server database and will not work with local database!

Part C - Creating a dynamic data web site

- Create a new Web Site project from *ASP.NET Dynamic Data Entities Web Application* template.
- Add a new *ADO.NET Entity Data Model* item to the project.
  - Choose to **generate model from database**.
- Edit the global.asax file according to the comments in that file.
- Run and test the site.

Part D – Editing the data model to use stored procedures

It is common requirement of information security standards that applications are not allowed to directly manipulate the data. All data manipulation has to be done using views and stored procedures. This kind of separation of data and application allows DBA-s to optimize the database operations more than it would be possible with direct database operations.

- Open Server Explorer and add a new stored procedure to the database you created.
- The stored procedure should take two parameters (Name and Street) and add a new row to the Customer table if it does not yet exist.

The stored procedure should look similar to that:

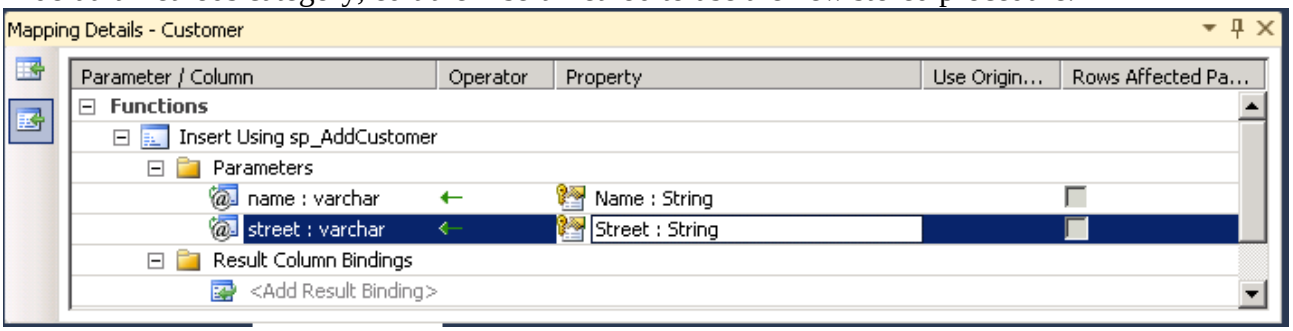
```
CREATE PROCEDURE dbo.sp_AddCustomer
      (
      @name varchar(50),
      @street varchar(100)
      )
AS
      IF (SELECT COUNT(*) FROM [Customer] WHERE [Name]=@name AND
[Street]=@street) = 0
      BEGIN
            INSERT INTO [Customer] ([Name],[Street]) VALUES (@name,
@street);
      END
      RETURN
```

You can test the procedure with following SQL command:

```
EXECUTE sp_AddCustomer 'Some Name', 'Some Street';
```

- Save changes.
- Open the Entity Data Model (edmx file) and choose to *Update model from database….*
- Choose to add the stored procedure and click *OK*.
- Click on the **Customer** object and open its properties.

- In default methods category, edit the Insert method to use the new stored procedure.



- Run and test the project.

Notes

It is possible to other data sources with LINQ as well (e.g. XML file or ODBC data connection), but this tutorial will not cover these.

Additional resources

- **eLearning**: http://learning.microsoft.com
  - Collection 6261: Developing Rich Experiences using Microsoft .NET Framework 3.5 & Visual Studio 2008
- **Free eBook „Introduction to LINQ“:** http://introducinglinq.com/
- **Microsoft IT Academy:** https://itacademy.microsoftelearning.com
  - Collection 6461: Visual Studio 2008: Windows Communication Foundation
  - Collection 6463: Visual Studio 2008 ASP.NET 3.5
  - Collection 6464: Visual Studio 2008 ADO.NET 3.5
  -