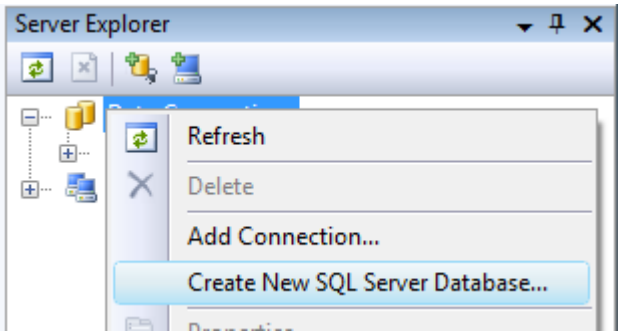
Introduction to LINQ

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

Preparations

You need to have SQL Server (2005 or newer), Visual Studio 2008 SP1 and .NET Framework 3.5 to complete this tutorial. 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. Mono Developer and Mono Framework can be used to create an similar solution as it supports LINQ as well.

Creating a new database



- 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 varchar(50) required (not null).
  - Street of type varchar(100) required (not null).
- Save changes and open table data view.
- Add a few entries to the table and save changes.

Creating a simple console project

- Create a new Console Application project.
- Add a new item (LINQ to SQL Classes) to the project.
- Drag the table from Server Explorer to the LINQ to SQL view.
- Edit the application main method:
  - Create new DataContext object (The type of the object will be the name of the LINQ to SQL mapping with DataContext appended to it. The default would be DataClassesDataContext) .
  - Print line to display the table header (Name, Street).
  - Loop through the Customers table and print it out.

The resulting Main method should look similar to that:

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

- Run and test the application.

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.

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

```
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);
}
Console.ReadKey();
```

  - Write the result to the console.
- Run and test the application.

Creating a dynamic data web site

- Create a new Web Site project from Dynamic Data Web Site template.
- Add a new item (LINQ to SQL Classes) to the project.
- Drag the table from Server Explorer to the LINQ to SQL view.
- Edit the global.asax file according to the comments in that file.
- Run and test the site.

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.
- Save changes.

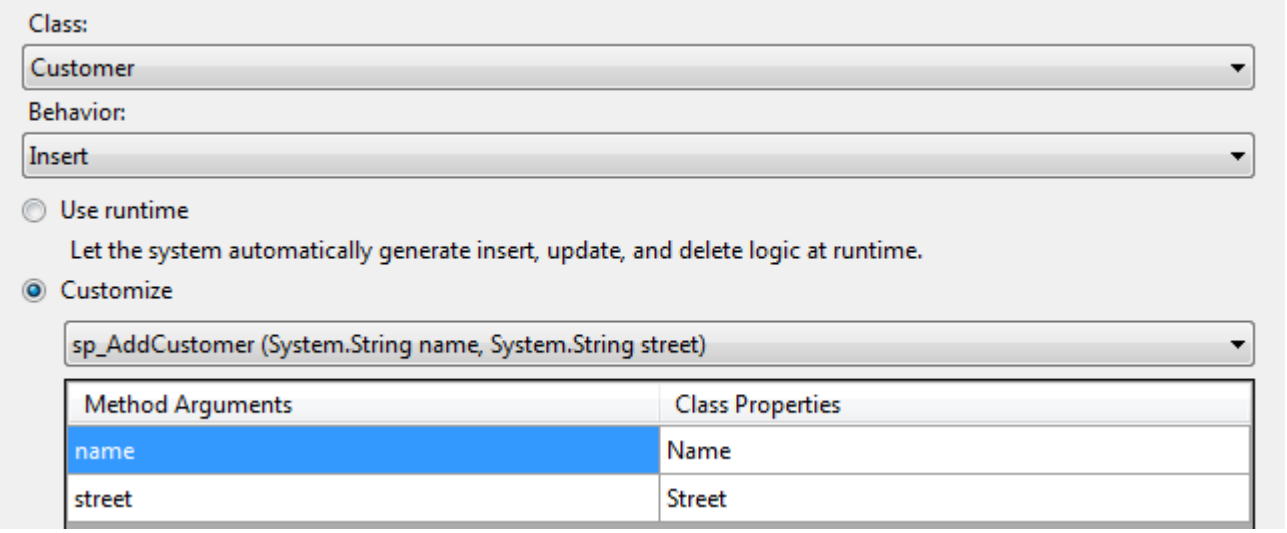
The stored procedure should look similar to that:

```
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';
```

- Open LINQ to SQL mapping (dbml file) and drag the stored procedure to the view.
- 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.