

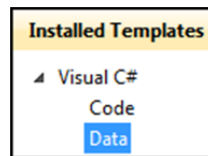
## Lesson 10-2: Add LINQ data classes to a project

LINQ, which stands for Language Integrated Query, is a relatively new addition to .NET which makes working with databases much easier.

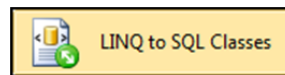
Before LINQ, programmers would usually create their own set of classes to generate the SQL needed to interact with a database. This took a lot of time and effort.

ASP.NET's LINQ classes automatically create the SQL needed to interact with your data, making it unnecessary to learn SQL. LINQ also streamlines the development process by removing the need to create your own data access classes.

- 1 Open *Spark* from your sample files folder.
- 2 Add LINQ data classes to the project.
  1. Right-click on *Spark* in the *Solution Explorer* and click Add→New Item... from the shortcut menu.
  2. Click *Data* in the left panel.

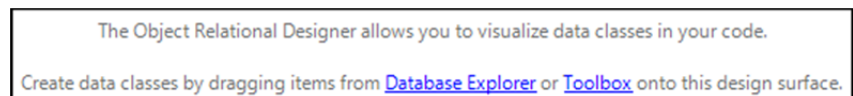


3. Click *LINQ to SQL Classes* in the center panel.



4. Type **Spark.dbml** into the *Name* text box.
5. Click *Add*.

The *dbml* file will be added and will automatically be opened, displaying what it calls the *Object Relational Designer*.



Although your LINQ data class file has been created, you need to tell it to add classes for the objects in your database using this designer.

- 3 Add a table to your LINQ classes.
  1. Open the *Database Explorer* by clicking its tab underneath the *Solution Explorer*.
  2. Expand the *Spark.mdf* database.
  3. Expand the *Tables* folder.
  4. Click and drag the *Customer* table from the *Tables* folder onto the *Object Relational Designer* in the central panel.

The table will appear in the LINQ designer panel.

## important

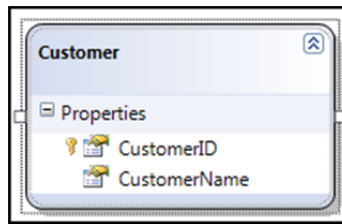
### You must rebuild the LINQ data classes if your database structure changes

Although it's very easy to create LINQ data classes, they do not automatically update when the structure (design) of a database changes.

The only way to reliably update LINQ data classes is to recreate the entire *dbml* file. You can do this by first deleting the old *dbml* file and then creating a new one.

In theory it is possible to refresh individual items but I have found that this occasionally causes problems that prevent the project from building.

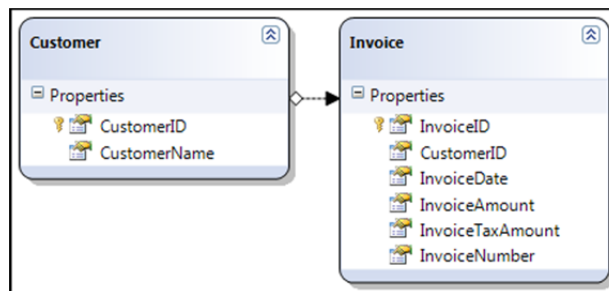
For this reason I recommend that you always recreate the entire *dbml* file if your database structure changes.



Your LINQ file now contains a class for the *Customer* table, which allows you to easily search and update the table using the C# code you'll learn later in this session.

#### 4 Add the *Invoice* table to your LINQ classes.

In the same way as you did with the *Customer* table, drag the *Invoice* table onto the LINQ designer.



The *Invoice* table is now available to your LINQ classes.

You'll notice the arrow between the two tables. This indicates the presence of a *relationship* between the tables.

The line and arrow tell you that each customer may have many invoices.

LINQ classes allow you to very easily find all of the invoices for any customer and to find the customer for any invoice. You'll see this in action in: *Lesson 10-4: Retrieve multiple rows of data using LINQ*.

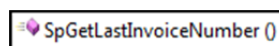
#### 5 Add a stored procedure to your LINQ classes.

Stored procedures are similar to methods, but are stored inside the database. Stored procedures contain SQL code that can manipulate data in ways that would be difficult using C#. In a team environment, stored procedures are usually written by the DBA.

The stored procedure in this example finds the highest invoice number in the *Invoice* table.

1. Expand the *Stored Procedures* folder in the *Database Explorer*.
2. Drag the *SpGetLastInvoiceNumber* procedure onto the LINQ designer.

The stored procedure will appear in the right panel of the *Linq Designer*.



Now that the stored procedure is part of the LINQ data class, you can call it just like any other C# method. You'll do this in: *Lesson 10-5: Sort results and call stored procedures using LINQ*.

