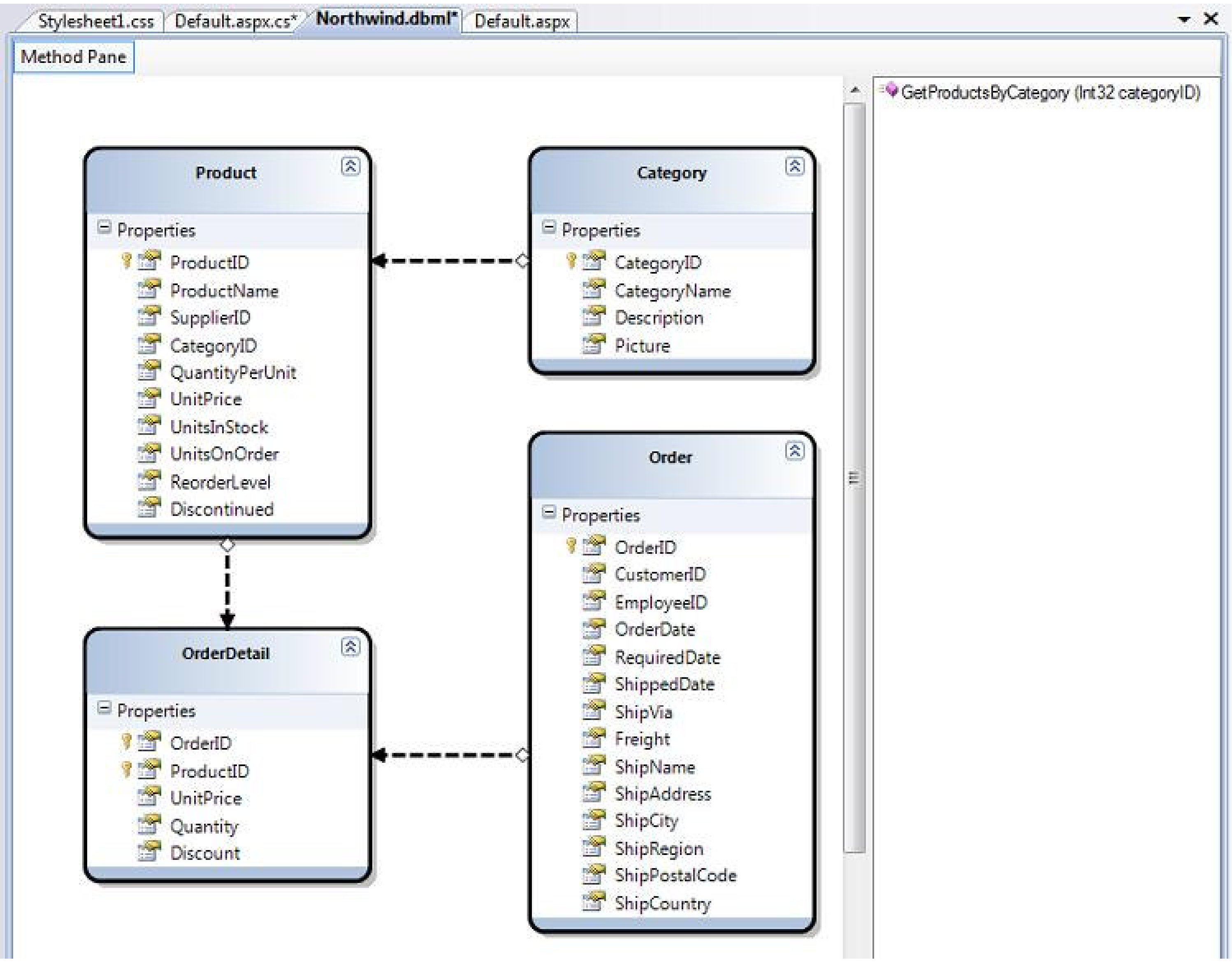
LINQ to SQL (Part 2 - Defining our Data Model Classes)

In Part 1 of my LINQ to SQL blog post series I discussed "What is LINQ to SQL" and provided a basic overview of some of the data scenarios it enables.

In my first post I provided code samples that demonstrated how to perform common data scenarios using LINQ to SQL including:

- How to query a database
- How to update rows in a database
- How to insert and relate multiple rows in a database
- How to delete rows in a database
- How to call a stored procedure
- How to retrieve data with server-side paging

I performed all of these data scenarios using a LINQ to SQL class model that looked like the one below:



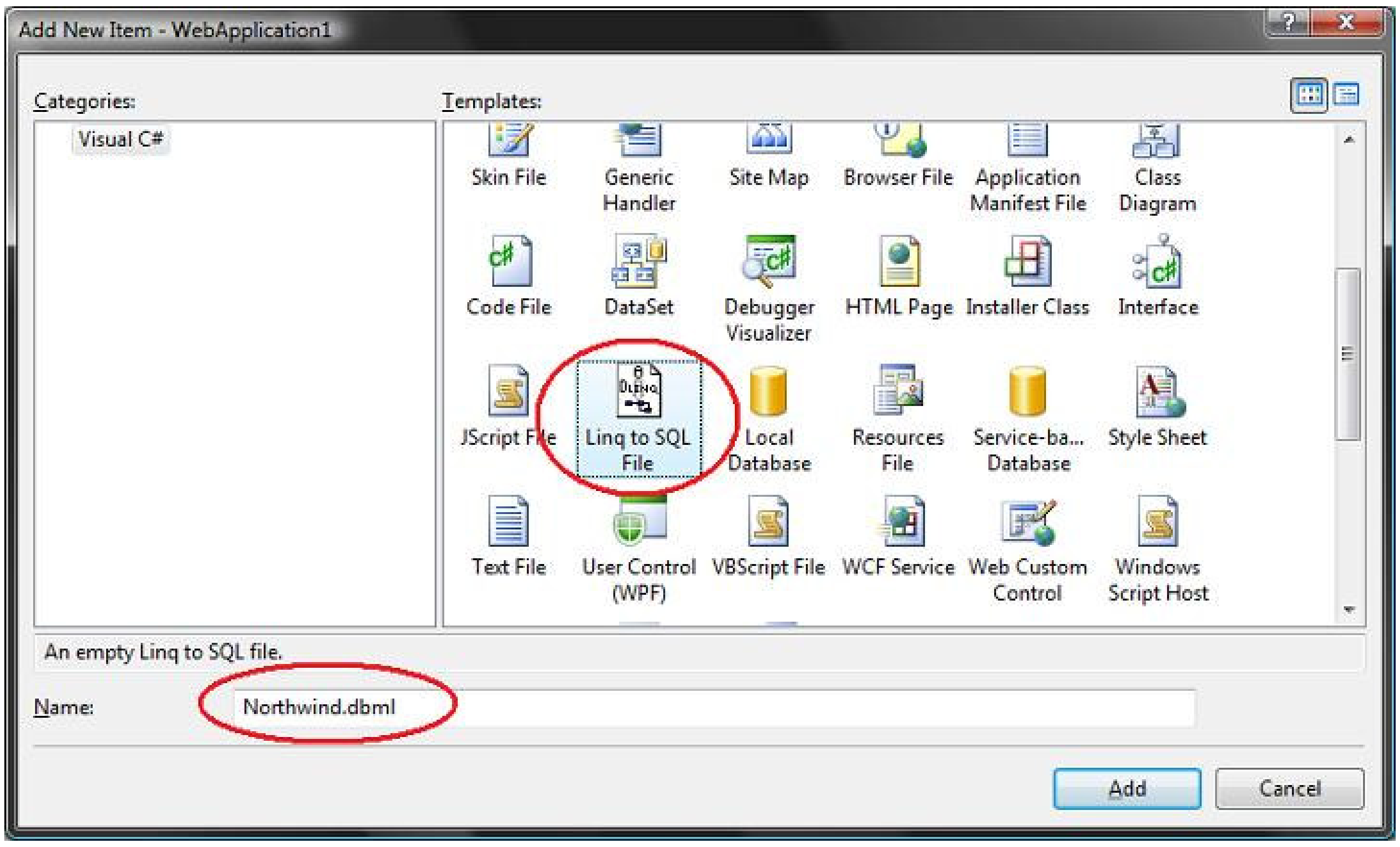
In this second blog post in the series I'm going to go into more detail on how to create the above LINQ to SQL data model.

LINQ to SQL, the LINQ to SQL Designer, and all of the features that I'm covering in this blog post series will ship as part of the .NET 3.5 and Visual Studio "Orcas" release.

You can follow all of the steps below by downloading either Visual Studio "Orcas" Beta 1 or Visual Web Developer Express "Orcas" Beta1. Both can be installed and used side-by-side with VS 2005.

Create a New LINQ to SQL Data Model

You can add a LINQ to SQL data model to an ASP.NET, Class Library or Windows client project by using the "Add New Item" option within Visual Studio and selecting the "LINQ to SQL" item within it:

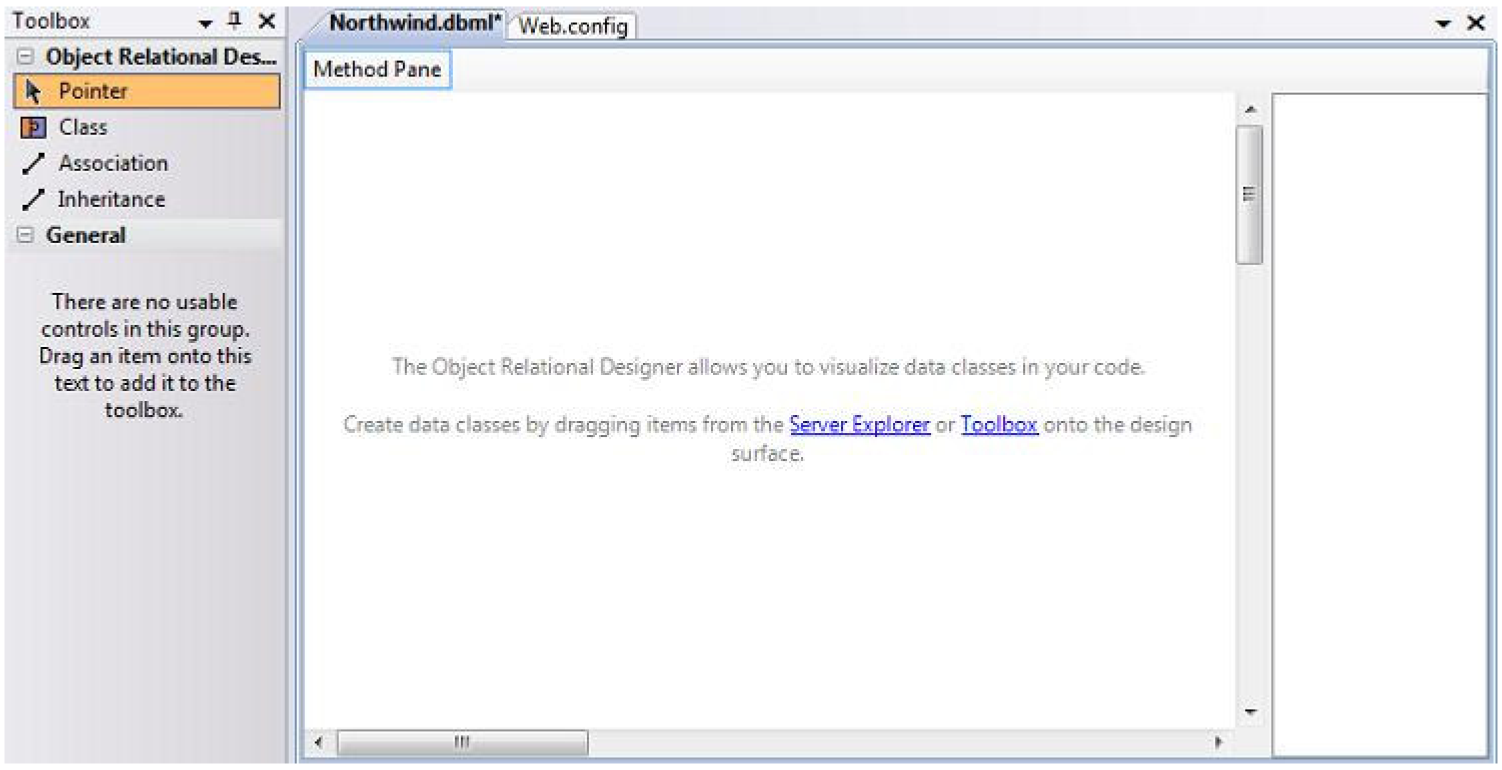


An empty Linq to SQL file.

Name: Northwind.dbml

Selecting the "LINQ to SQL" item will launch the LINQ to SQL designer, and allow you to model classes that represent a relational database. It will also create a strongly-typed "DataContext" class that will have properties that represent each Table we modeled within the database, as well as methods for each Stored Procedure we modeled. As I described in Part 1 of this blog post series, the DataContext class is the main conduit by which we'll query entities from the database as well as apply changes back to it.

Below is a screen-shot of an empty LINQ to SQL ORM designer surface, and is what you'll see immediately after creating a new LINQ to SQL data model:


Toolbox
Object Relational Des...
Pointer
Class
Association
Inheritance
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Northwind.dbml*
Web.config
Method Pane
The Object Relational Designer allows you to visualize data classes in your code.
Create data classes by dragging items from the Server Explorer or Toolbox onto the design surface.

Entity Classes

LINQ to SQL enables you to model classes that map to/from a database. These classes are typically referred to as "Entity Classes" and instances of them are called "Entities". Entity classes map to tables within a database. The properties of entity classes typically map to the table's columns. Each instance of an entity class then represents a row within the database table.

Entity classes defined with LINQ to SQL do not have to derive from a specific base class, which means that you can have them inherit from any object you want. All classes created using the LINQ to SQL designer are defined as "partial classes" - which means that you can optionally drop into code and add additional properties, methods and events to them.

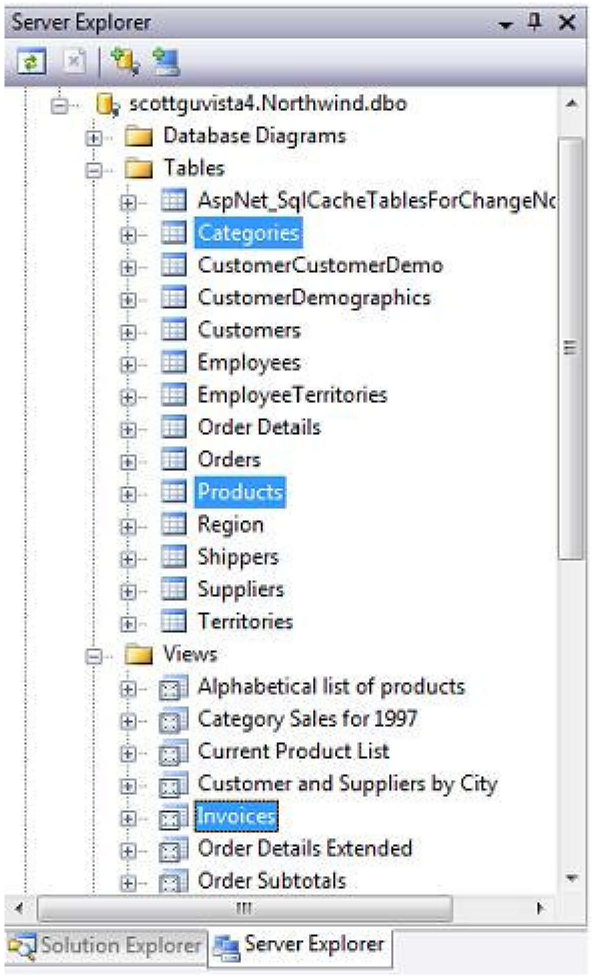
Unlike the DataSet/TableAdapter feature provided in VS 2005, when using the LINQ to SQL designer you do not have to specify the SQL queries to use when creating your data model and access layer.

Instead, you focus on defining your entity classes, how they map to/from the database, and the relationships between them. The LINQ to SQL OR/M implementation will then take care of generating the appropriate SQL execution logic for you at runtime when you interact and use the data entities. You can use LINQ query syntax to expressively indicate how to query your data model in a strongly typed way.

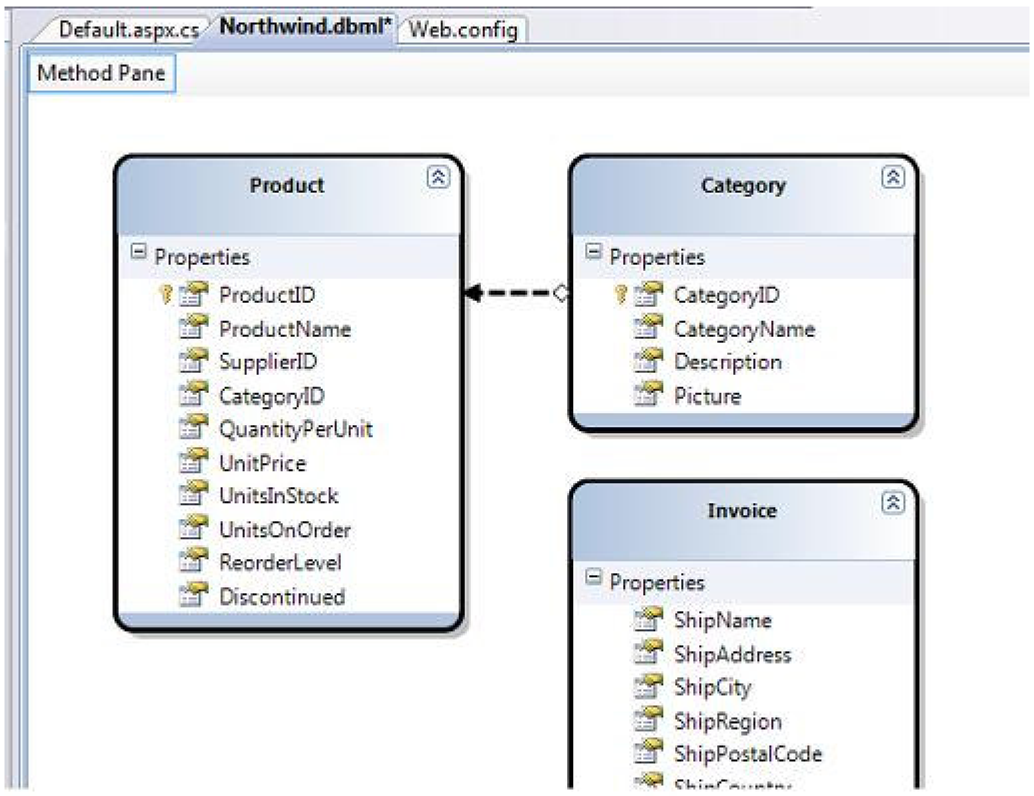
Creating Entity Classes From a Database

If you already have a database schema defined, you can use it to quickly create LINQ to SQL entity classes modeled off of it.

The easiest way to accomplish this is to open up a database in the Server Explorer within Visual Studio, select the Tables and Views you want to model in it, and drag/drop them onto the LINQ to SQL designer surface:



When you add the above 2 tables (Categories and Products) and 1 view (Invoices) from the "Northwind" database onto the LINQ to SQL designer surface, you'll automatically have the following three entity classes created for you based on the database schema:

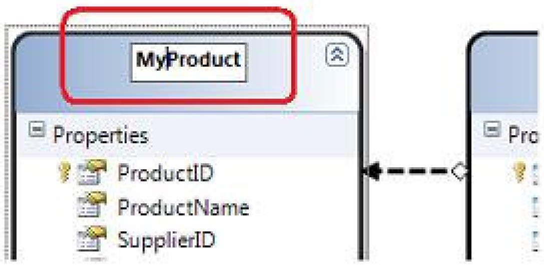


Using the data model classes defined above, I can now run all of the code samples (expect the SPROC one) described in Part 1 of this LINQ to SQL series. I don't need to add any additional code or configuration in order to enable these query, insert, update, delete, and server-side paging scenarios.

Naming and Pluralization

One of the things you'll notice when using the LINQ to SQL designer is that it automatically "pluralizes" the various table and column names when it creates entity classes based on your database schema. For example: the "Products" table in our example above resulted in a "Product" class, and the "Categories" table resulted in a "Category" class. This class naming helps make your models consistent with the .NET naming conventions, and I usually find having the designer fix these up for me really convenient (especially when adding lots of tables to your model).

If you don't like the name of a class or property that the designer generates, though, you can always override it and change it to any name you want. You can do this either by editing the entity/property name in-line within the designer or by modifying it via the property grid:



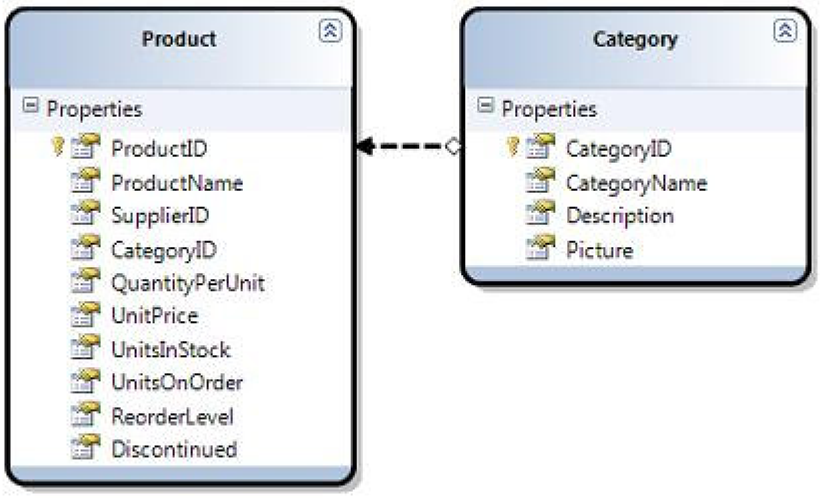
The ability to have entity/property/association names be different from your database schema ends up being very useful in a number of cases. In particular:

1) When your backend database table/column schema names change. Because your entity models can have different names from the backend schema, you can decide to just update your mapping rules and not update your application or query code to use the new table/column name.

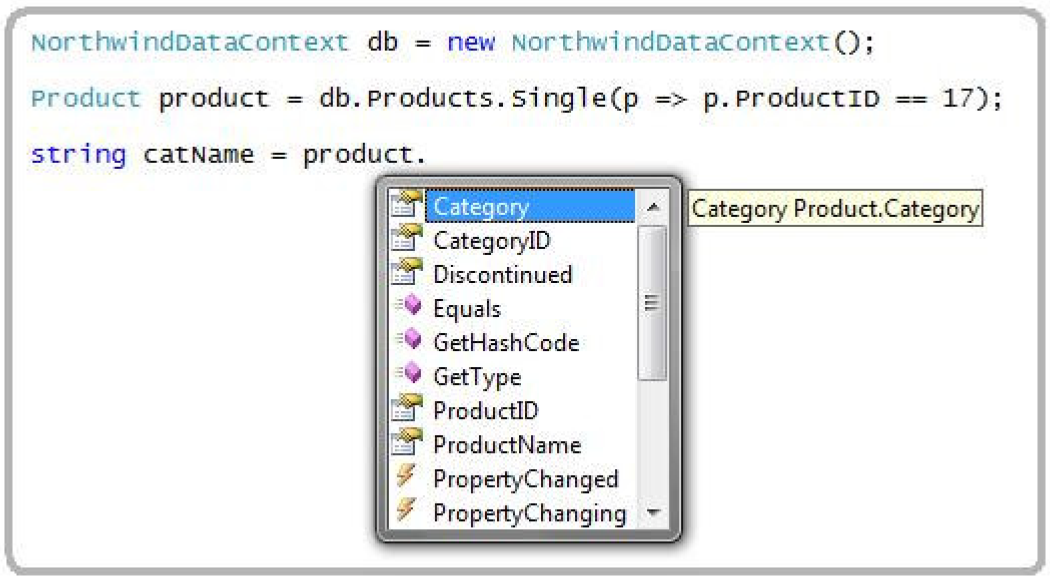
2) When you have database schema names that aren't very "clean". For example, rather than use "au_lname" and "au_fname" for the property names on an entity class, you can just name them to "LastName" and "FirstName" on your entity class and develop against that instead (without having to rename the column names in the database).

Relationship Associations

When you drag objects from the server explorer onto the LINQ to SQL designer, Visual Studio will inspect the primary key/foreign key relationships of the objects, and based on them automatically create default "relationship associations" between the different entity classes it creates. For example, when I added both the Products and Categories tables from Northwind onto my LINQ to SQL designer you can see that a one to many relationship between the two is inferred (this is denoted by the arrow in the designer):



The above association will cause cause the Product entity class to have a "Category" property that developers can use to access the Category entity for a given Product. It will also cause the Category class to have a "Products" collection that enables developers to retrieve all products within that Category.



```
NorthwindDataContext db = new NorthwindDataContext();

Product product = db.Products.Single(p => p.ProductID == 17);

string catName = product.
```

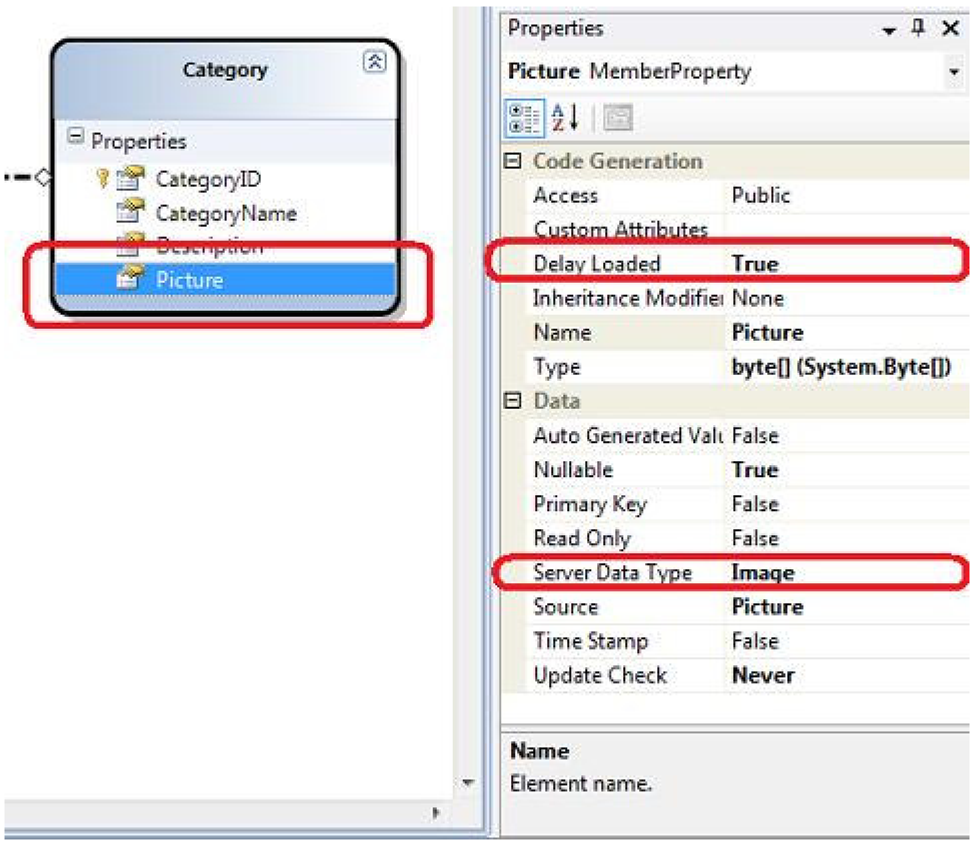
If you don't like how the designer has modeled or named an association, you can always override it. Just click on the association arrow within the designer and access its properties via the property grid to rename, delete or modify it.

Delay/Lazy Loading

LINQ to SQL enables developers to specify whether the properties on entities should be prefetched or delay/lazy-loaded on first access. You can customize the default pre-fetch/delay-load rules for entity properties by selecting any entity property or association in the designer, and then within the property-grid set the "Delay Loaded" property to true or false.

For a simple example of when I'd want to-do this, consider the "Category" entity class we modeled above. The categories table inside "Northwind" has a "Picture" column which stores a (potentially large) binary image of each category, and I only want to retrieve the binary image from the database when I'm actually using it (and not when doing a simply query just to list the category names in a list).

I could configure the Picture property to be delay loaded by selecting it within the LINQ to SQL designer and by settings its Delay Loaded value in the property grid:



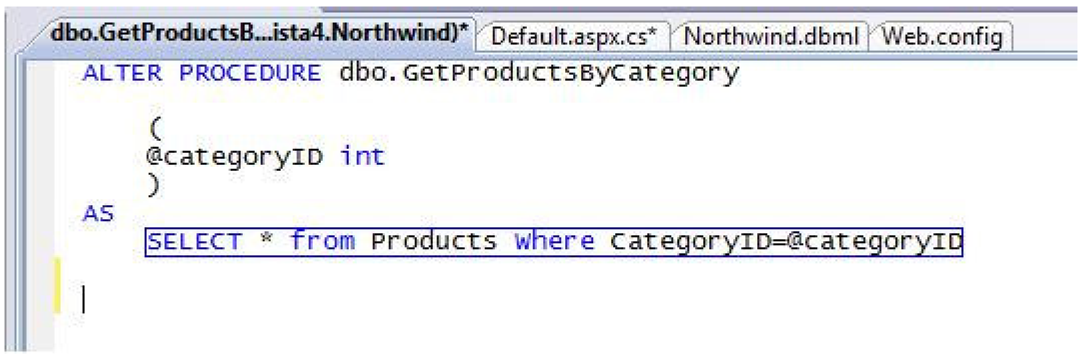
Code Generation	
Access	Public
Custom Attributes	
Delay Loaded	**True**
Inheritance Modifier	None
Name	**Picture**
Type	**byte[] (System.Byte[])**
Data	
Auto Generated Valu	False
Nullable	**True**
Primary Key	False
Read Only	False
Server Data Type	**Image**
Source	**Picture**
Time Stamp	False
Update Check	**Never**

Name
Element name.

Note: In addition to configuring the default pre-fetch/delay load semantics on entities, you can also override them via code when you perform LINQ queries on the entity class (I'll show how to-do this in the next blog post in this series).

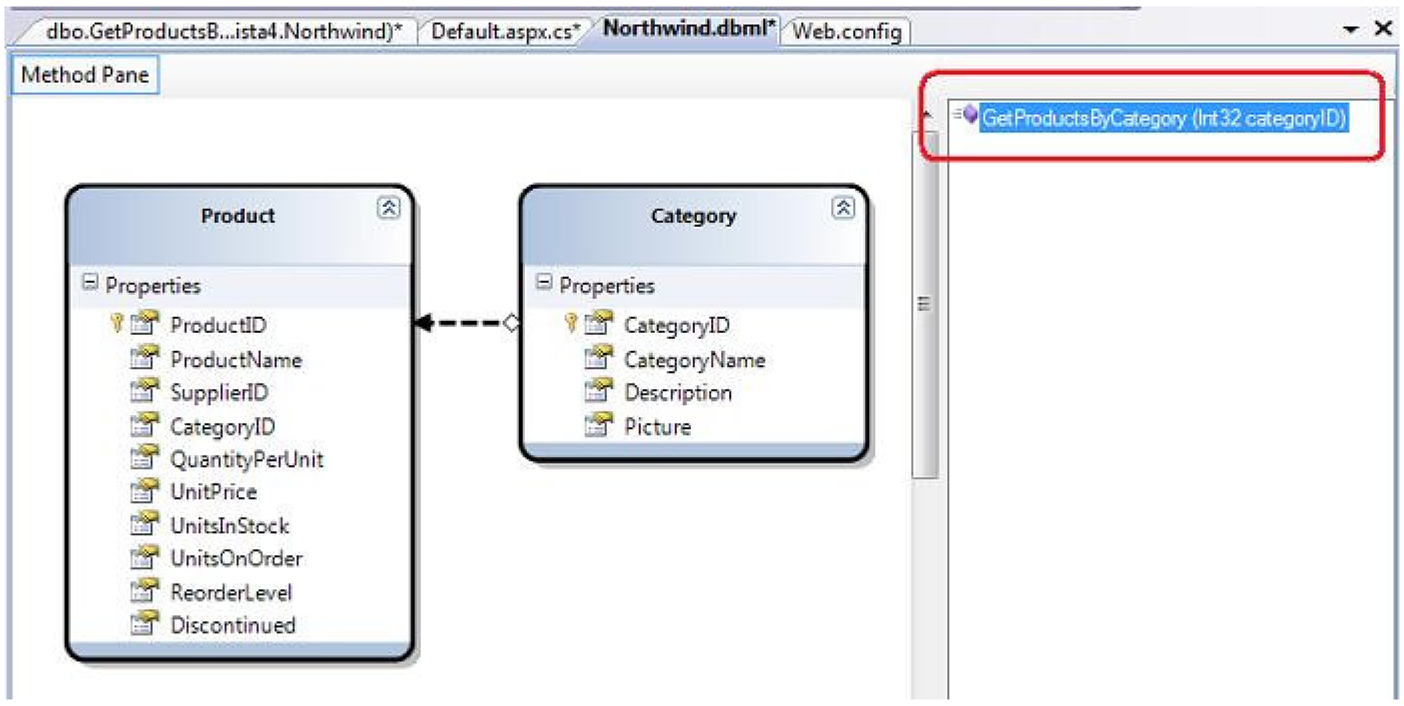
Using Stored Procedures

LINQ to SQL allows you to optionally model stored procedures as methods on your DataContext class. For example, assume we've defined the simple SPROC below to retrieve product information based on a categoryID:

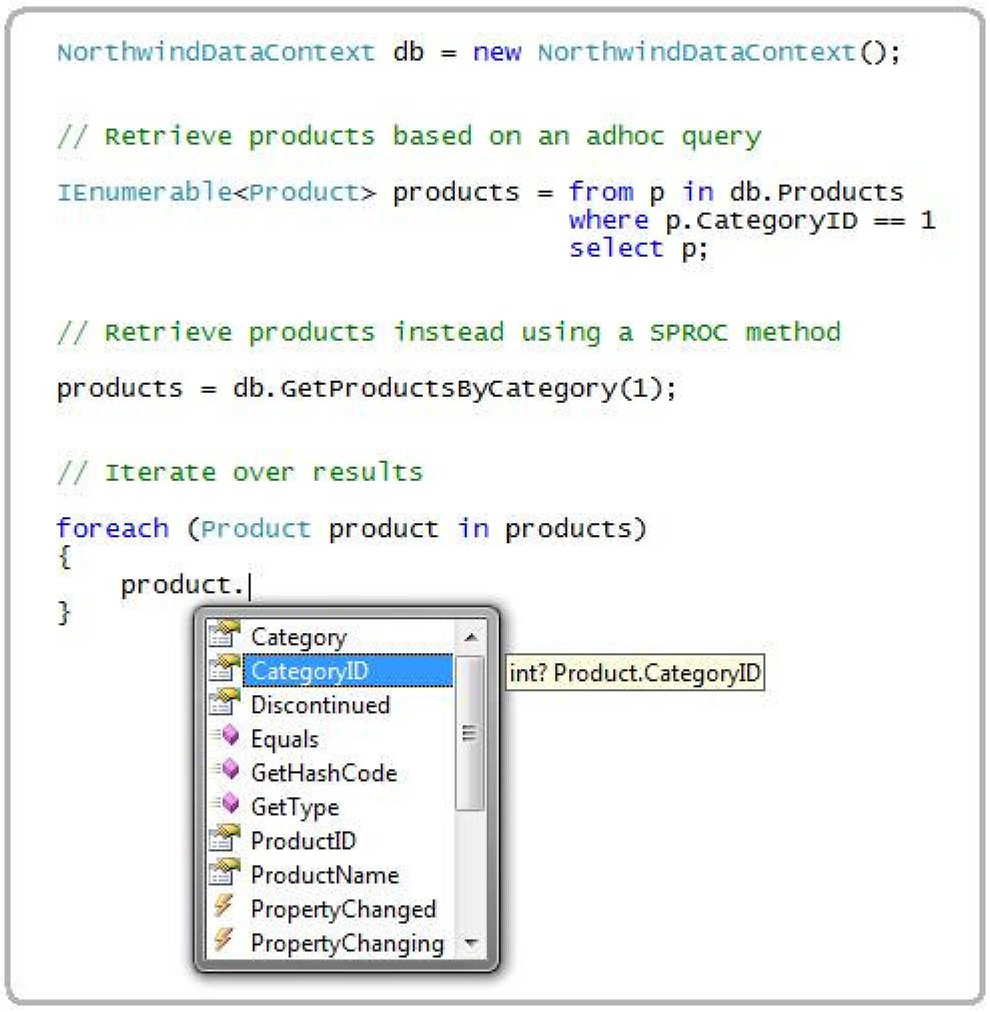


```
ALTER PROCEDURE dbo.GetProductsByCategory

    (
    @categoryID int
    )
AS
    SELECT * from Products where CategoryID=@categoryID
```

I can use the server explorer within Visual Studio to drag/drop the SPROC onto the LINQ to SQL designer surface in order to add a strongly-typed method that will invoke the SPROC. If I drop the SPROC on top of the "Product" entity in the designer, the LINQ to SQL designer will declare the SPROC to return an IEnumerable<Product> result:



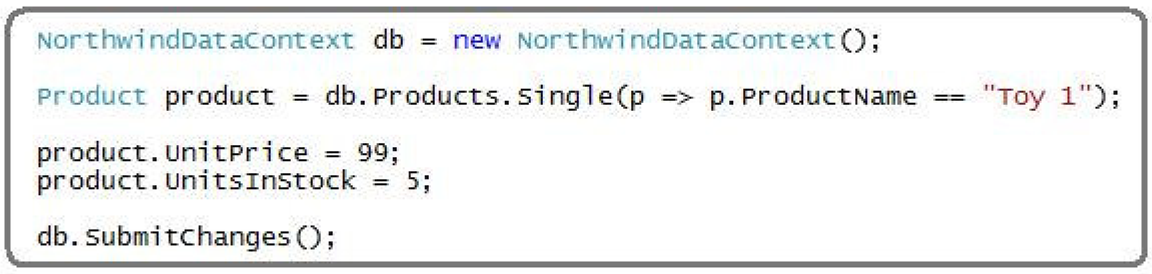
I can then use either LINQ Query Syntax (which will generate an adhoc SQL query) or alternatively invoke the SPROC method added above to retrieve product entities from the database:



```
NorthwindDataContext db = new NorthwindDataContext();

// Retrieve products based on an adhoc query

IEnumerable<Product> products = from p in db.Products
                                where p.CategoryID == 1
                                select p;

// Retrieve products instead using a SPROC method

products = db.GetProductsByCategory(1);

// Iterate over results

foreach (Product product in products)
{
    product.
}
```

Using SPROCs to Update/Delete/Insert Data

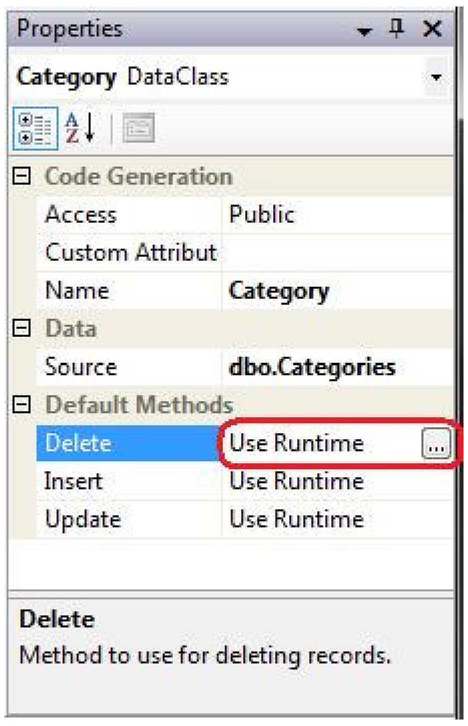
By default LINQ to SQL will automatically create the appropriate SQL expressions for you when you insert/update/delete entities. For example, if you wrote the LINQ to SQL code below to update some values on a "Product" entity instance:



```
NorthwindDataContext db = new NorthwindDataContext();

Product product = db.Products.Single(p => p.ProductName == "Toy 1");

product.UnitPrice = 99;
product.UnitsInStock = 5;

db.SubmitChanges();
```

By default LINQ to SQL would create and execute the appropriate "UPDATE" statement for you when you submitted the changes (I'll cover this more in a later blog post on updates).

You can also optionally define and use custom INSERT, UPDATE, DELETE sprocs instead. To configure these, just click on an entity class in the LINQ to SQL designer and within its property-grid click the "..." button on the Delete/Insert/Update values, and pick a particular SPROC you've defined instead:



Properties

Category DataClass

Code Generation	
Access	Public
Custom Attribut	
Name	Category
Data	
Source	dbo.Categories
Default Methods	
Delete	Use Runtime
Insert	Use Runtime
Update	Use Runtime

Delete
Method to use for deleting records.

What is nice about changing the above setting is that it is done purely at the mapping layer of LINQ to SQL - which means the update code I showed earlier continues to work with no modifications required. This avoids developers using a LINQ to SQL data model from having to change code even if they later decide to put in a custom SPROC optimization later.

Summary

LINQ to SQL provides a nice, clean way to model the data layer of your application. Once you've defined your data model you can easily and efficiently perform queries, inserts, updates and deletes against it.

Using the built-in LINQ to SQL designer within Visual Studio and Visual Web Developer Express you can create and manage your data models for LINQ to SQL extremely fast. The LINQ to SQL designer also provides a lot of flexibility that enables you to customize the default behavior and override/extend the system to meet your specific needs.

In upcoming posts I'll be using the data model we created above to drill into querying, inserts, updates and deletes further. In the update, insert and delete posts I'll also discuss how to add custom business/data validation logic to the entities we designed above to perform additional validation logic.

Mike Taulty also has a number of great LINQ to SQL videos that I recommend checking out here. These provide a great way to learn by watching someone walkthrough using LINQ to SQL in action.

Hope this helps,

Scott

Published Tuesday, May 29, 2007 1:17 AM by ScottGu
Filed under: ASP.NET, Visual Studio, .NET, LINQ, Data

Comments

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 6:57 AM by FransBouma

Two questions:

1) the proc seems to return a strongly typed element. In SqlServer there's only one way to find out the resultset of a proc, via SET FMTONLY ON; exec proc ; SET FMTONLY OFF. The thing is that with a temptable in the proc or multiple selectstatements (with if's for example) it often gives up, i.o.w.: it's not 100% reliable. (the temptable situation even throws an exception). How is linq to sql make sure this always works and also, how does it deal with renamed columns in the resultset or slightly different named columns which mean the same ? Or is it parsing the SQL from the proc? (which doesn't work in encrypted proc scenario's)

2) you say you can use a proc for insert/update of an entity. While that's cool, it has restrictions: the proc has to obey a given format otherwise updates/inserts won't work. (which field maps to which proc parameter, which parameter returns the new identity value, new NEW_SEQUENTIALID() value etc. and for updates, which parameters are for the predicates and which for the values). I haven't read anything about this yet, while it does seem to me it's very important because you're effectively mapping your entity on a proc, not on a table/view anymore.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 7:31 AM by Mike

Looks great!

When you want to use custom stored procedures for insert/update, is the method signature (name, arguments and return value) completely free, or does it need to conform to something you have defined? For instance, if we have a Product, with some properties lazy loaded, and we change only one of the properties, what arguments does the update SPROC receive? If it receives all properties of the Product as arguments, are they first loaded to their original values? How does it handle properties that are null? Hope you can clear this up for me.

Thanks for the post!

PS. How does the pluralization work for non-english table names?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 10:59 AM by Peter

1) What happens when my database schema changes? Are all those settings you mentioned (like "Delay Loaded") stored separately? Would I lose those? Can I refresh only certain tables or is it an all or nothing deal?

2) Following your example: each product has a category. When I ask for all the products in the database (let's say I have 50 products), would LINQ to SQL generate 1 query that would bring everything or 1 query for the products and 50 queries for each category. I imagine it only generates one query but then how does it distinguish between the Name field in the Products table and the Name field in the Categories table, or do I have to make sure to name them differently?

3) You talked about "pluralization" where the name of the entity class is altered. What happens if my table names are already in "singular" form? Does it just leave them like that?

Thanks.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:26 AM by ScottGu

Hi Peter,

1) The settings for different entities are stored separately, so it isn't an all or nothing thing. If you want, you can just re-create certain tables or individual settings. This makes it much easier to handle schema changes.

2) If each product has a category association, and you look over 50 products and access each product's category, the number of SQL queries performed will depend on how you've configured the entity relationship to be loaded. If you delay/lazy load it, then there will be a separate database call for each product (so potentially up to 50 calls). If you set delay load to false, or programmatically indicate that you want to retrieve the category as well when querying for the products, then you'll have 2 queries to the database - one to retrieve the products, and then another to retrieve all of the categories for the 50 products. LINQ to SQL will then associate the retrieved categies to the appropriate Products. I'll cover this more in my next post in this series which will cover querying.

3) If you already have a name that isn't plural, then it will just keep it as-is. You can also optionally change it to anything else you want as well.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 12:51 PM by ScottGu

Hi Frans,

1) LINQ to SQL does use the 'FMTONLY' approach at design-time. It is possible that this method does not produce the correct result or any result. In these cases you can describe the procedures yourself using the designer, hand editting the DBML file or by adding the appropriately mapped signatures directly on your DataContext.

2) The mapping of procedure arguments against entity fields is fully configurable in the designer, DBML or mapping file/attributes. For updates you can choose between mapping original or current state values and in which direction the information flows in/out. Note that when you click on the "..." elipse in the property grid to customize the INSERT, UPDATE, DELETE behavior it will bring up a dialog that allows you to map the entity parameters to/from the sprocs.

Thanks,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 4:48 PM by Mesan

So when is Linq to SQL going to support DB2?

# LINQ to SQL (Part 2 - Defining our Data Model Classes) - ScottGu&#39;s Blog

Tuesday, May 29, 2007 5:03 PM by LINQ to SQL (Part 2 - Defining our Data Model Classes) - ScottGu's Blog

Pingback from LINQ to SQL (Part 2 - Defining our Data Model Classes) - ScottGu&#39;s Blog

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 5:43 PM by Paul R

Is the Code Generation "Type" restricted by System types or will developers be able to use custom data types?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 8:08 PM by Juan marÃa

Great post Scott.

You can read this post in spanish here:

thinkingindotnet.wordpress.com/.../linq-to-sql-2%c2%aa-parte-definiendo-nuestras-clases-del-modelo-de-datos

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 8:38 PM by Kyle @ RPMWare

First ... Nice new look!

What about complex FK relationships? Ecommerce is an easy example ...

Brands (Id, Name), Items (Id, Name, Price), Multimedia (Id, Type, ItmId, Image)

The the above example if Type is "B" then ItmId (Multimedia) refers to a BrandId if Type is "I" then ItmId refers to an ItemId. Is there an easy way to create those relationships?

This seems great for the simple stuff, I can't wait to play with it on some more complex things. It hard for us dataset guys to wrap our heads around the OR/M world.

Can you "go live" with Orcas code?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 9:54 PM by Vikram

wow Briliant. But I wold also like to see a post on LINQ with SQl (Stored procedures only) and alos a situation where by I can update only a certain values in the database

# Link Listing - May 29, 2007

Tuesday, May 29, 2007 10:23 PM by Christopher Steen

RESTful Web Services [Via: Jon Udell] SCSFContrib is Alive! [Via: bsimser] Expression Studio on MSDN...

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:07 PM by ScottGu

Hi Kyle,

You can support inheritance with entities, and vary the sub-class of a row based on the content values within it. So for example, you could have a "Car" entity class and then have a "SportsCar" and "SUV" sub-class. All three of these could persist into a single table. The entity instance for each row could then be driven based on a column value within the table.

Hope this helps,

Scott

P.S. "Orcas" Beta2 which will ship later this summer will support a go-live license. :-)

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:08 PM by ScottGu

Hi Mesan,

Unfortunately LINQ to SQL doesn't support DB2 yet. There is, though, a LINQ to NHibernate implementation out there, and I believe that supports DB2. The mapping layer is different than LINQ to SQL, but you can use the same LINQ queries against both.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:10 PM by ScottGu

Hi PaulR,

Can you clarify what you mean by the "code generation type"? You can create complex entities that map to rows using LINQ to SQL, so that isn't limited to only system types.

Thanks,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:27 PM by Aleksey

Scott, the only thing I don't like about LINQ is that I cannot return unnamed type var from a function. Why can't we name it?

Instead of:

var myCustomers = from ... in ... select ...;

why not to use something like:

IEnumerable<public class Customer> myCustomers = from ... in ... select ...;

letting compiler to define new public class Customer making it possible to return IEnumerable<Customer> from the function.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, May 29, 2007 11:51 PM by Steve

Thanks for the writeup - good stuff Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 12:07 AM by ScottGu

Hi Aleksey,

Anonymous types in C# and VB unfortunately don't allow you to specify return arguments like that. What you can do, though, is have the return type be IEnumerable<Customer> and then pass things back that way.

My samples in this Query Syntax blog post: weblogs.asp.net/.../new-orcas-language-feature-query-syntax.aspx show how to-do this.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 3:16 AM by Keith Patton

Hi,

The following article talks about inheritance support only being available in Linq for Entities/Entity Framework which i understand is delayed till next year.

dotnetaddict.dotnetdevelopersjournal.com/adoef_vs_linqsql.htm

When you say Linq for Sql suppors inheritance, is this "table per class" inheritance only? Does the designer support the definition of a discriminator value to distinguish between types in the hierarchy? An example of table per class inheritance would be very useful!

Finally, i'm wondering what the "upgrade" path to Linq for Entities will be like. Presumably, as long as we keep the entity class structure the same we can swap in the mapping to the conceptual model and remove the direct to db mappings without affecting the application.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 3:35 AM by ScottGu

Hi Keith,

LINQ to SQL supports single table inheritance, and the LINQ to SQL designer supports this modeling as well (you can use the "inheritance" element from the toolbox to setup relationships between entities on the designer surface, and there is a discriminator property you can set to distinguish them).

For a good overview of how LINQ to SQL entity inheritance works, I'd recommend downloading these LINQ whitepapers: download.microsoft.com/.../orcasmarchctpwhitepapers.zip

If you open the "LINQ to SQL Overview" document and go to page 38 you can see an example of how inheritance works.

In terms of what the "upgrade" path to LINQ to Entities will be, it will probably depend a little on your application. But if you keep the entity class structure the same, you should be able to use the same LINQ queries against both, which should make swapping out the mapping layer much easier.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 4:44 AM by Mikael Östberg

Hello Scott!

Excellent post!

I'm really convinced that LINQ really helps developing a nice and clean data layer. It's really what we all have been waiting for.

However, my concern is performance. I have an upcoming project which is a funds trading system that has extremely demanding requirements regarding performance. I really would like to choose LINQ to SQL, but do I dare to? This design decision comes with a heavy load of responsibility, so I really want to be certain that it'll hold.

Have you made any performance tests so far?

Thanks!

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 7:58 AM by Stephen Woolhead

Was wondering if there is going to be any support for creating these models off of DB schemas defined in a Team Edition for Database Professionals project? Or if there is going to be any other form of intergration between DBPro and the Linq technology?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 9:04 AM by Dan

Hey Scott,

Is there any way to wire up LinqToSql to SQL Server 2005 query notifications?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 10:23 AM by Walter

Hi Scott,

1) is there support in LINQ to SQL for recognising (for example) a new column added to a table in the database on the fly? Or does this always necessitate revisiting the designer and a recompile?

2) how easy is it to fetch from a database using LINQ and then serialize this to an in-memory XML document? (And then do something like process the XML using an external XSLT file)

It would be nice to be able to add a new column into the database and then refer to it in the XSLT without having to recompile the classes in between.

# links for 2007-05-30 &raquo; mhinze.com

Wednesday, May 30, 2007 11:30 AM by links for 2007-05-30 » mhinze.com

Pingback from links for 2007-05-30 &raquo; mhinze.com

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 12:05 PM by Josh

Hi Scott,

What happened to the <RECOMMENDED></RECOMMENDED> link in the nav bar? It had the tips, tricks, and gotchas. That was an excellent reference, is it still available?

Thanks :)

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 12:37 PM by Morten

How will the new spatial features of Katmai fit into all of this? Will I get spatial keywords (intersects, within etc) that I can use as part of my LINQ expression?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 12:53 PM by ScottGu

Hi Mikael,

The good news is that the performance of LINQ to SQL is really, really good. I believe the numbers I've seen has the Beta2 numbers within ~10% of raw datareader performance for common scenaros.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 12:56 PM by ScottGu

Hi Stephen,

>>> "Was wondering if there is going to be any support for creating these models off of DB schemas defined in a Team Edition for Database Professionals project? Or if there is going to be any other form of intergration between DBPro and the Linq technology?"

That is a good question, and to be honest with you I'm not 100% sure. LINQ to SQL does ship with a command-line tool for creating models, so it is possible to automate creating your data layer from the command/build line. I don't know if you can create them directly from a .SQL file though (I don't think you can but am not 100% sure).

Hope this helps,

Scott

P.S. Note that you can definitely use DBPro with LINQ to SQL, and LINQ to SQL makes it much easier to refactor your database. I'm just not sure about the model creation part of your question.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 1:00 PM by ScottGu

Hi Walter,

>> 1) is there support in LINQ to SQL for recognising (for example) a new column added to a table in the database on the fly? Or does this always necessitate revisiting the designer and a recompile?

If you add a new column to the database your LINQ to SQL entity classes should continue to work just fine, unless you add some constraint or restriction on the column that prevents the table from being used. If you want to actually access and manipulate the new column, then you'll need to add the property that maps to it to your entitiy class.

>> 2) how easy is it to fetch from a database using LINQ and then serialize this to an in-memory XML document? (And then do something like process the XML using an external XSLT file)

XML datatypes in SQL are represented as strings with LINQ to SQL Entities. You could use XLINQ to query on an XML column within your LINQ to SQL entitiy - but this querying would happen in your middle-tier (within ASP.NET). You can't execute a remote XQuery against the database and filter returned results based on that in the first release.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 1:05 PM by ScottGu

Hi Josh,

I'm working on getting my tips/tricks link back on the site (I seemed to lose it when I updated the skin of my blog).

Here is a pointer to access it in the meantime: weblogs.asp.net/.../ASP.NET-2.0-Tips_2C00_-Tricks_2C00_-Recipes-and-Gotchas.aspx

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 1:59 PM by Ben Hayat

Hi Scott;

Two questions:

a) When you create a model class of your table, can you then create instances of that class within a form? i.e. Having two instances of customer object on my form that points to two different customers?

b) Can you have the model to create SP on server side when Inserting/Updating/deleting records, instead doing it from he client side?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 4:21 PM by Walter

Hi Scott, thanks for your earlier reply.

Regarding point 2, I wasn't thinking about XML-type columns in the database, but about taking the entire entity as fetched by LINQ to SQL and mapping that to some form of in-memory XML document.

I suppose manually iterating over the IEnumerable<T> collection and creating XElement, XAttribute, etc, objects is one possibility but can you suggest a nice way of automating this?

Perhaps a solution that makes use of the System.Xml.Serialization namespace...could the LINQ to SQL entity classes be decorated with attributes such as XmlRoot to enable this? I am interested to hear your thoughts on this.

Thanks

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 8:04 PM by Will

How would you insert a record and return the primary key of the new record using LINQ?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, May 30, 2007 9:15 PM by Rob

How does LINQ to SQL support data concurrency? Must I maintain a list of changed fields (and original values) in my data objects and craft sprocs to get concurrency?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 3:19 AM by ScottGu

Hi Rob,

LINQ to SQL supports concurrency, and allows you to detect changes in a couple of ways.

One way is to compare origional and changed values when doing updates. LINQ to SQL entities support this automatically.

Another option is to add a timestamp column to your table. LINQ to SQL will then use this to detect if the underlying row has changed and automatically handle concurrency for you this way.

I'll cover both approaches in an upcoming blog post when I talk about updates.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 3:21 AM by ScottGu

Hi Ben,

You can definitely have multiple instances of entity/model classes within a form. Each instance can point to different customer objects.

In my sproc section above I talked a little about how you can define stored procedures to handle insert/update/delete scenarios for your entities. This is definitely supported.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 3:23 AM by ScottGu

Hi Will,

In my LINQ to SQL Part 1 post: weblogs.asp.net/.../using-linq-to-sql-part-1.aspx I showed an insert code sample.

To retrieve the new primary key of an added object, you can access its primary key column immediately after the SubmitChanges() call. The entity will at that point have been updated to reflect the new PK from the database.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 3:24 AM by ScottGu

Hi Walter,

LINQ ships with XLINQ - and the projection features of LINQ enable you to-do some pretty nice transformations of objects into XML with it. All you'd need to-do is modify the select clause of your LINQ to SQL query and specify a XLINQ projection in that to convert it to XML.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 5:11 AM by Skup

Hi,

the DataContext object is IDisposable. I did not try, but I guess that enumarating elements from a query after disposing the DataContext will result in an exception.

The question is, in ASP.Net, where is the best place to dispose the DataContext ?

For now, I do it in a OnDispose() override, is there a better place ?

Thanks

Skup

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 6:56 AM by Boris Yeltsin's Zombie

Hey Scott,

Is there a button in the designer to automatically refresh all the database schema changes into the DAL? (this is a nice feature in LLBLGen)

Also - do you have any links for writing my own LINQ to XXXX - where XXXX is the database du jour. I imagine this might be a royal pain-in-the-ass, but I'd love to see how possible it is to build a LINQ to MySQL and LINQ to PostgreSQL.

Is the source to LINQ to SQL going to be available?

- BY's Zombie

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, May 31, 2007 11:47 AM by Ben Hayat

Hi Scott;

Thank you for answering my questions. I do want to tell you what a great job you're doing by breaking this new technology into "Understandable" blogs. Your time and effort is very much appreciated and your style of writing brings excitement to read about LINQ.

I have not downloaded the Orcas yet to get the SDK docs, but do you have any links to good docs on C# 3.0 and LINQ that are "Up to date"? Most of the stuff I find, are old and things have already changed with LINQ!

Thanks!

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, June 01, 2007 6:44 AM by Damien Guard

Would still love to see some guidance on the scopen and usage of the DataContext especially in shared web environments taking the caching into consideration.

[)amien

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, June 01, 2007 1:03 PM by Matt

Hi Scott,

is it possible to design a model in Linq and then let link create the Database with all table structures?

And, assuming I have imported e.g. the northwind databases table customers. Now I add a new property to this entity. Can Linq add this property to the customers table, if I set all necessary properties?

Regards,

Matt

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Saturday, June 02, 2007 12:26 PM by Mehfuz Hossain

Great post! in addtion i have downloaded the session of Mix07 = > Dev 04 , where Anders gives a hand on of working with dbml. Must check out!

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Monday, June 04, 2007 5:44 PM by Claudia

Hello Scott,

I'd like to know how to represent one to many relationship to the same table. Eg: Category table has many categories, and I have a parentCategoryID in my Category table. Is there a way to return a Category object within the Category class using this graphic tool ? If not, can I do it using this:

[Table(Name = "Category")]

class Category

{

...

[Association(Storage = "someCategoryVariable",OtherKey = "parentCategoryID")]

public Category CategoryParent{ ... }

}

# Linq - Way too easy!

Tuesday, June 05, 2007 7:32 AM by TheCoder

Linq - Way too easy!

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, June 06, 2007 12:07 PM by El Guapo

Hello,

Thanks for the post. Is it possible to create many-to-one relationships? (i.e., just an entity reference, no collections). I don't see any way to do it. But it must be possible?

Thanks

El Guapo

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, June 07, 2007 2:07 PM by Fduch

Thanks for the tutorials.

Is there any way to update values in database without fetching any data from it?

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Saturday, June 09, 2007 9:51 AM by James

Hi Scott,

How does the generated entity classes behave when serialized? Specifically, I'm thinking of a tree-like object graph scenario. What happens when you deserialize the entities? Is there a manual detach/attach process involved or can you immediately start using it as if it were a "live" object graph?

Very interested to know the answers to this.

Thanks

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, June 14, 2007 1:12 PM by Scott Roberts

I too am eager to see more. Specifically:

1. How do I update my dbml when the DB schema changes? I don't see a "refresh" button on the designer.

2. What is the recommended "best practice" for using this in an ASP.NET environment? In Part 1 you mentioned releasing the DataContext after each page and just storing the Entity Objects themselves then using Attach() to attach them to a new DataContext on postback. However, it appears that changes made to the object prior to Attach() are "lost". That means I can't pass an Entity Object from page to page (updating the object along the way, in a wizard for example) then save all changes at the end.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, June 15, 2007 2:34 AM by Stuart

Hi,

Ever since I installed the Visual Web Developer beta, I've been getting the following error notice when I interact with a connection string (such as opening a file containing table adapters or clicking on an adapter's properties):

Following error occurred while getting connection string information from configuration.

"Object reference not set to an instance of an object"

This happens now in both VS 2005 and VWD. I've tried restoring an old web.config, but no dice. The program still compiles and runs, but it's a pain clicking through 20 or 30 warnings when I want to open a file

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, June 15, 2007 4:13 PM by Todd

Is it possible to use many-to-many relationships with LINQ to SQL? For example, a Person table and a Group table could be related with a mapping table so that one Person can belong to multiple Groups and a single Group can 'contain' multiple Persons.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Sunday, June 17, 2007 4:23 PM by ScottGu

Hi Stuart,

If you can send me an email with more details about the error, I'll have someone help figure out what is going wrong.

Thanks,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Sunday, June 17, 2007 4:24 PM by ScottGu

Hi Scott,

In general what I do when the schema changes is to delete the entity and then drag/drop it onto the surface again.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Sunday, June 17, 2007 4:26 PM by ScottGu

Hi Ed/Todd,

With LINQ to SQL in .NET 3.5 you need to model M:M relationships using an intermediate class. I'll cover how to-do this in a future blog post.

LINQ to Entities supports M:M without needing an intermediate class - so that is another option to use.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Monday, June 25, 2007 7:56 PM by mathmax

Hi ScottGu,

you said : "You can support inheritance with entities, and vary the sub-class of a row based on the content values within it. So for example, you could have a "Car" entity class and then have a "SportsCar" and "SUV" sub-class. All three of these could persist into a single table. The entity instance for each row could then be driven based on a column value within the table."

Could you give me an example. I tried writting this :

```
  public class Item
  {
      [Column(Id = true)]
      private int Id;
      [Column()]
```

```
    private int Category_Id;
  }
  [Table()]
  public class Product : Item
  {
    private Iso _Origin;
    [Column(DBType = "nchar(3)")]
    public Iso Origin
    {
      get { return _Origin; }
      set { _Origin = value; }
    }
  }
```

but it doesn't work. It however works well if I write :

```
  [Table()]
  public class Product : Item
  {
    [Column(Id = true)]
    private int Id;
    [Column()]
    private int Category_Id;
    private Iso _Origin;
    [Column(DBType = "nchar(3)")]
    public Iso Origin
    {
      get { return _Origin; }
      set { _Origin = value; }
    }
  }
```

An other point that is not clear for me :

what is the difference between linq to sql and linq to Entities ? Is the code above linq to Entities ? If not, is there currently a beta from linq to Entities ?

Thank you in advance.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, June 28, 2007 2:56 AM by ScottGu

Hi Mathmax,

Here is a pointer to a recent LINQ to SQL whitepaper that discusses how to handle single table inheritance: download.microsoft.com/.../orcasmarchctpwhitepapers.zip

I am going to try and put together a blog post in the next few weeks that shows a good example of it as well.

Thanks,

Scott

# LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 4:27 AM by Community Blogs

Last month I started a blog post series covering LINQ to SQL. LINQ to SQL is a built-in O/RM (object

# LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 5:16 AM by ASP.NET

Last month I started a blog post series covering LINQ to SQL. LINQ to SQL is a built-in O/RM (object

# Quick LINQ link list

Wednesday, July 04, 2007 5:48 PM by Fabrice's weblog

Some quick links about LINQ: Articles about extension methods by the Visual Basic team Third-party LINQ

# Quick LINQ link list

Friday, July 06, 2007 8:12 AM by Linq in Action News

Some quick links about LINQ: Articles about extension methods by the Visual Basic team Third-party LINQ

# Linq to SQL之查询和添加。

Sunday, July 08, 2007 8:35 PM by 勤勤同学

本文以Northwind数据库中的Customers和Orders表为例说明Linq to SQL的查询和添加是怎样操作的。

# LINQ to SQL (Part 4 - Updating our Database)

Wednesday, July 11, 2007 1:20 PM by Blogs

Over the last few weeks I&#39;ve been writing a series of blog posts that cover LINQ to SQL. LINQ to

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, July 13, 2007 12:12 AM by Arnaud

Adding attributes to Associations doesn't work.

I have a Member entity that has ChildMembers and a ParentMember. In the table, Member rows have a parent_id column that points to their parent Member in the same table. To represent this as a WCF DataContract, I have added the DataMember attribute to the Associations so that when I pass a Member object to a client, the client has access to it's childMembers. The DataMember attribute is never added to the .designer.cs file.

I tried adding it manually, that works fine.

Must be a bug with the designer.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, July 17, 2007 10:55 AM by Daniel Plomp

Hello Scott,

This are great articles. I have a question about the pluralization (is that how you write this!??)

If you have in dutch a table called 'Persoon' (which means 'Person'), the plural would become 'Persoons' (which should mean 'Persons'). Only in dutch it should be 'Personen'.

Is there any way to change this behaviour? I couldn't find a property i.e. to change this. Only in code, but then if you change your .dbml file, it would regenerate the classes.

Hope there is or will be a solution for this.

Thanks,

Daniel Plomp

The Netherlands

# LINQ to SQL

Thursday, July 19, 2007 10:18 PM by OpsanBlog

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Friday, July 20, 2007 5:26 PM by AzamSharp

Hi Scott,

I got one question about lazy loading. I know that by default the lazy loading is disabled. Let's say I am using the Categories table which off course has Products. Now, I don't want to load Products when I am loading Categories. How can I disable that. I think the LINQ to SQL diagram created using drag and drop lacks the foreign key relationships. It simply mirrors the database schema. It should show all the properties associated with Categories which includes Products.

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Tuesday, July 24, 2007 2:27 PM by ScottGu

Hi Amam,

>>>>>> I got one question about lazy loading. I know that by default the lazy loading is disabled. Let's say I am using the Categories table which off course has Products. Now, I don't want to load Products when I am loading Categories. How can I disable that. I think the LINQ to SQL diagram created using drag and drop lacks the foreign key relationships. It simply mirrors the database schema. It should show all the properties associated with Categories which includes Products.

By default lazy loading is enabled for association properties. That means when you retrieve a category it does not actively load the products for each category. It only does this the first you access a category's products property. This avoids you running into performance issues with large graphs of objects.

You can optionally then choose to tell the DataContext to pre-load particular associations so that you grab the data all in one go.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Wednesday, July 25, 2007 2:41 AM by Daniel Plomp

Hi Scott,

I still like to know about the plural(s) inside LINQ. What can I do? See my previous post...

Thanks,

Daniel

# Valer BOCAN&#8217;s Web Log &raquo; Blog Archive &raquo; Visual Studio 2008 Beta 2 is here!

Thursday, July 26, 2007 5:02 PM by Valer BOCAN’s Web Log » Blog Archive » Visual Studio 2008 Beta 2 is here!

Pingback from Valer BOCAN&#8217;s Web Log &raquo; Blog Archive &raquo; Visual Studio 2008 Beta 2 is here!

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Thursday, July 26, 2007 7:28 PM by Clayton Powell

Hi Scott,

I was wondering if the the LINQ to SQL will support SQL Server 2000. If yes, then are there any features that will not work (or are perhaps less efficient - like paging with TAKE etc).

Also can you put the ORM in a class library and use it from both a web application and an EXE? Will it automatically use a connection string from web.config/app.config depending on the calling environment (like the dataset designer does)?

Clayton

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Saturday, July 28, 2007 2:01 AM by ScottGu

Hi Clayton,

LINQ to SQL will support SQL 2000. One feature that requires SQL 2005 is the server-side paging support (where you only do the paging in the database). This uses the SQL 2005 ROW_NUMBER() feature which is only in SQL 2005.

Hope this helps,

Scott

# re: LINQ to SQL (Part 2 - Defining our Data Model Classes)

Saturday, July 28, 2007 2:03 AM by ScottGu

Hi Daniel,

>>>>> I still like to know about the plural(s) inside LINQ. What can I do? See my previous post...

You can change the pluralization in the LINQ to SQL ORM designer by selecting class entity and then clicking on the name. It will then give you an edit box that allows you to name it whatever you want.

Hope this helps,

Scott

# All you can LINQ &raquo; Quick LINQ link list

Tuesday, July 31, 2007 10:38 PM by All you can LINQ » Quick LINQ link list

Pingback from All you can LINQ &raquo; Quick LINQ link list

# LINQ to SQL Debug Visualizer

Wednesday, August 01, 2007 1:40 AM by Community Blogs

Probably the biggest programming model improvement being made in .NET 3.5 is the work being done to make

# The asp:ListView control (Part 1 - Building a Product Listing Page with Clean CSS UI)

Friday, August 10, 2007 4:21 AM by Community Blogs

One of the new controls in ASP.NET 3.5 that I think will be very popular is the &lt;asp:ListView&gt;

# Common Data Patterns with LINQ &laquo; blog.jemm.net

Friday, August 10, 2007 4:32 PM by Common Data Patterns with LINQ « blog.jemm.net

Pingback from Common Data Patterns with LINQ &laquo; blog.jemm.net

# LINQ to SQL (Part 6 - Retrieving Data Using Stored Procedures)

Thursday, August 16, 2007 5:47 AM by Community Blogs

Over the last few weeks I've been writing a series of blog posts that cover LINQ to SQL. LINQ to SQL

# Creating a Web Portal with CRM 3 and LINQ

Monday, August 20, 2007 12:40 AM by rosslotharius.com

I have been spending a decent amount of time reading up and playing around with the new features that

# Creating a Web Portal with CRM 3 and LINQ

Monday, August 20, 2007 12:51 AM by rosslotharius.com

I have been spending a decent amount of time reading up and playing around with the new features that

# Creating a Web Portal with CRM 3 and LINQ

Wednesday, August 22, 2007 1:46 AM by Invoke Systems Microsoft Dynamics CRM 3.0 Developer's Weblog

I have been spending a decent amount of time reading up and playing around with the new features that...

# LINQ to SQL (Part 7 - Updating our Database using Stored Procedures)

Thursday, August 23, 2007 4:41 AM by Blogs

Over the last few weeks I&#39;ve been writing a series of blog posts that cover LINQ to SQL. LINQ to

# LINQ to SQL (Part 8 - Executing Custom SQL BLOCKED EXPRESSION

Monday, August 27, 2007 8:10 AM by Blogs

Over the last few weeks I&#39;ve been writing a series of blog posts that cover LINQ to SQL. LINQ to

# LINQ to SQL Tutorials von Scott Guthrie

Monday, September 10, 2007 2:35 AM by Jürgen Gutsch

Wer das Blog von Scott Guthrie (1) kennt, kennt sicher auch seine Tutorials über LINQ to SQL (2). Auf

# My Slides/Samples from MIX:UK - Building Silverlight Apps with .NET and Building ASP.NET Apps with VS 2008

Wednesday, September 12, 2007 10:19 PM by Community Blogs

The last two days I've been speaking at the the MIX:UK conference that was held this week in London.

# Inserting Data with LINQ | Andy Brudtkuhl

Monday, September 24, 2007 2:13 PM by Inserting Data with LINQ | Andy Brudtkuhl

Pingback from Inserting Data with LINQ | Andy Brudtkuhl

# Community Convergence XXXI

Tuesday, September 25, 2007 8:43 PM by Charlie Calvert's Community Blog

Welcome to the thirty-first edition of Community Convergence. This issue features links to seven very

# LINQ to SQL (Part 3 - 查询数据库)

Monday, October 01, 2007 9:21 PM by exp

LastmonthIstartedablogpostseries

coveringLINQtoSQL.

# ScottGu's 博客之－－LINQ to SQL 第二部分－定义数据模型类－－LINQ to SQL (Part 2 - Defining our Data Model Classes)

Sunday, January 06, 2008 12:40 AM by 韩现龙

更为简洁的版本将会发布在博客堂（除去了英文的版本），希望本文能对您有所帮助！ 允许转载，麻烦注明出处：www.cnblogs.com/hanxianlong

# More Linq to SQL &laquo; Jaspers&#8217; Weblog

Friday, January 18, 2008 3:20 PM by More Linq to SQL « Jaspers' Weblog

Pingback from More Linq to SQL &laquo; Jaspers&#8217; Weblog

# &nbsp; Visual Studio 2008, C# 3.0, LINQ, ASP.Net 3.5 Getting Started&nbsp;by&nbsp;Aneef.Net

Monday, February 18, 2008 3:40 AM by Visual Studio 2008, C# 3.0, LINQ, ASP.Net 3.5 Getting Started by Aneef.Net

Pingback from &nbsp; Visual Studio 2008, C# 3.0, LINQ, ASP.Net 3.5 Getting Started&nbsp;by&nbsp;Aneef.Net

# Detailed Look: Key components in LINQ to SQL and their Key Roles

Wednesday, February 27, 2008 4:02 PM by Corey Gaudin

As part of this blog, I plan to have an on-going set of articles that takes a detailed look into some

# Greg Hochard&#8217;s Blog &raquo; LINQ and ScottGu

Monday, March 17, 2008 12:49 PM by Greg Hochard's Blog » LINQ and ScottGu

Pingback from Greg Hochard&#8217;s Blog &raquo; LINQ and ScottGu

# [转自Scott]ASP.NET MVC框架(第四部分): 处理表单编辑和提交场景

Sunday, April 06, 2008 11:52 PM by 菩提树下的杨过

英文原文地址:weblogs.asp.net/.../asp-net-mvc-framework-part-4-handling-form-...

# [转自Scott]ASP.NET MVC框架(第四部分): 处理表单编辑和提交场景

Tuesday, April 08, 2008 12:37 AM by yjmyzz

ASP.NET MVC框架(第四部分): 处理表单编辑和提交场景

??????????????? &raquo; Blog Archive &raquo; LINQ to SQL??????Part 2 - Defining our Data Model Classes

Sunday, July 06, 2008 5:01 AM by ??????????????? » Blog Archive » LINQ to SQL??????Part 2 - Defining our Data Model Classes

Pingback from ??????????????? &raquo; Blog Archive &raquo; LINQ to SQL??????Part 2 - Defining our Data Model Classes

# ASP.NET MVC Archived Buzz, Page 1

Sunday, January 11, 2009 1:56 AM by ASP.NET MVC Archived Buzz, Page 1

Pingback from ASP.NET MVC Archived Buzz, Page 1

# LINQ Resources &laquo; Vakul Kumar More

Saturday, January 17, 2009 10:20 PM by LINQ Resources « Vakul Kumar More

Pingback from LINQ Resources &laquo; Vakul Kumar More