LINQ to SQL (Part 3 - Querying our Database)

Last month I started a blog post series covering LINQ to SQL. LINQ to SQL is a built-in O/RM (object relational mapping) framework that ships in the .NET Framework 3.5 release, and which enables you to easily model relational databases using .NET classes. You can then use LINQ expressions to query the database with them, as well as update/insert/delete data from it.

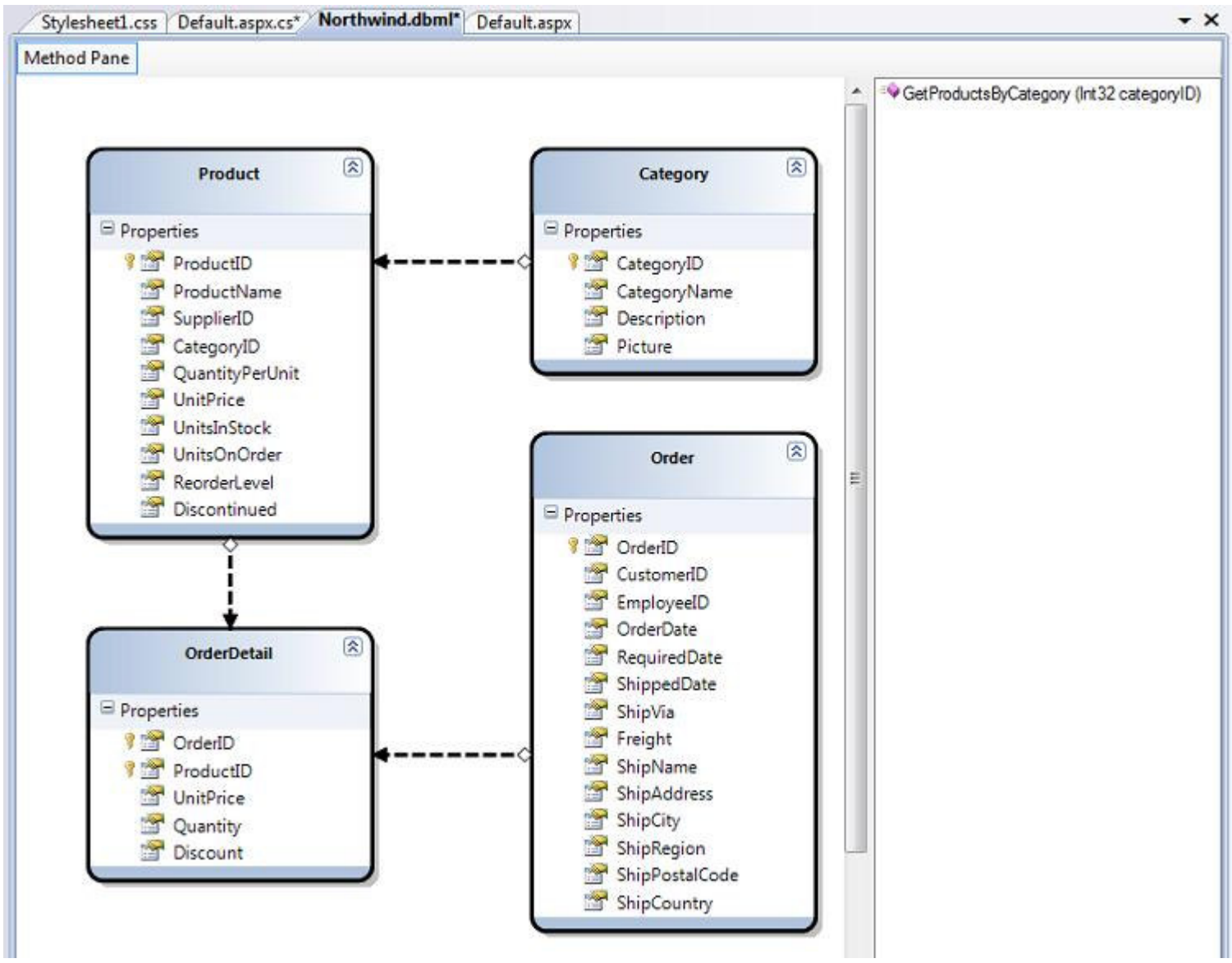
Below are the first two parts of my LINQ to SQL series:

- Part 1: Introduction to LINQ to SQL
- Part 2: Defining our Data Model Classes

In today's blog post I'll be going into more detail on how to use the data model we created in the Part 2 post, and show how to use it to query data within an ASP.NET project.

Northwind Database Modeled using LINQ to SQL

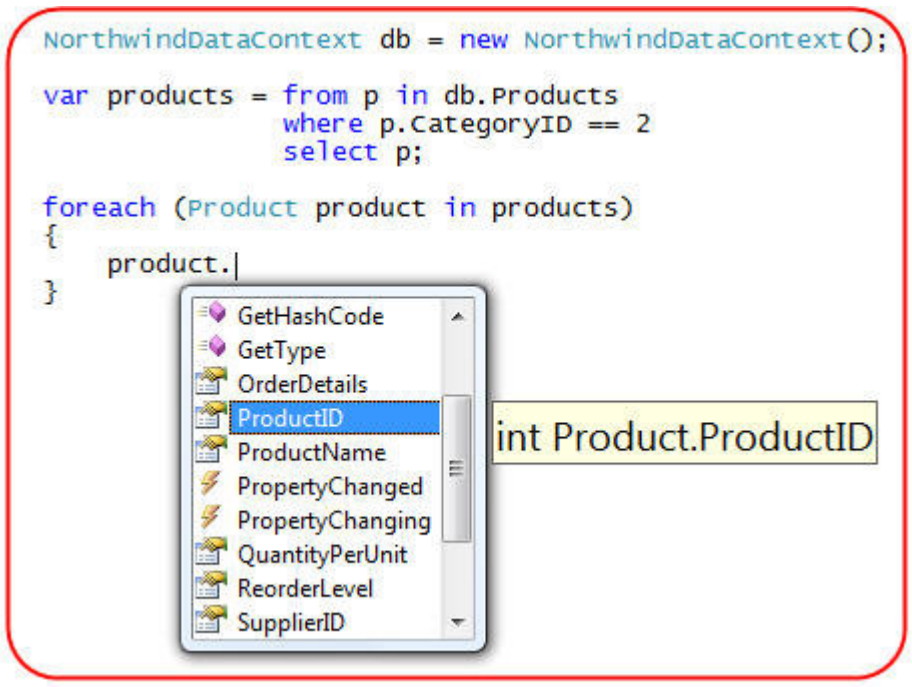
In Part 2 of this series I walked through how to create a LINQ to SQL class model using the LINQ to SQL designer that is built-into VS 2008. Below is the class model that we created for the Northwind sample database:



Retrieving Products

Once we have defined our data model classes above, we can easily query and retrieve data from our database. LINQ to SQL enables you to do this by writing LINQ syntax queries against the NorthwindDataContext class that we created using the LINQ to SQL designer above.

For example, to retrieve and iterate over a sequence of Product objects I could write code like below:



In the query above I have used a "where" clause in my LINQ syntax query to only return those products within a specific category. I am using the CategoryID of the Product to perform the filter.

One of the nice things above LINQ to SQL is that I have a lot of flexibility in how I query my data, and I can take advantage of the associations I've setup when modeling my LINQ to SQL data classes to perform richer and more natural queries against the database. For example, I could modify the query to filter by the product's CategoryName instead of its CategoryID by writing my LINQ query like so:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select p;
```

Notice above how I'm using the "Category" property that is on each of the Product objects to filter by the CategoryName of the Category that the Product belongs to. This property was automatically created for us by LINQ to SQL because we modeled the Category and Product classes as having a many to one relationship with each other in the database.

For another simple example of using our data model's association relationships within queries, we could write the below LINQ query to retrieve only those products that have had 5 or more orders placed for them:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.OrderDetails.Count > 5
               select p;
```

Notice above how we are using the "OrderDetails" collection that LINQ to SQL has created for us on each Product class (because of the 1 to many relationship we modeled in the LINQ to SQL designer).

Visualizing LINQ to SQL Queries in the Debugger

Object relational mappers like LINQ to SQL handle automatically creating and executing the appropriate SQL code for you when you perform a query or update against their object model.

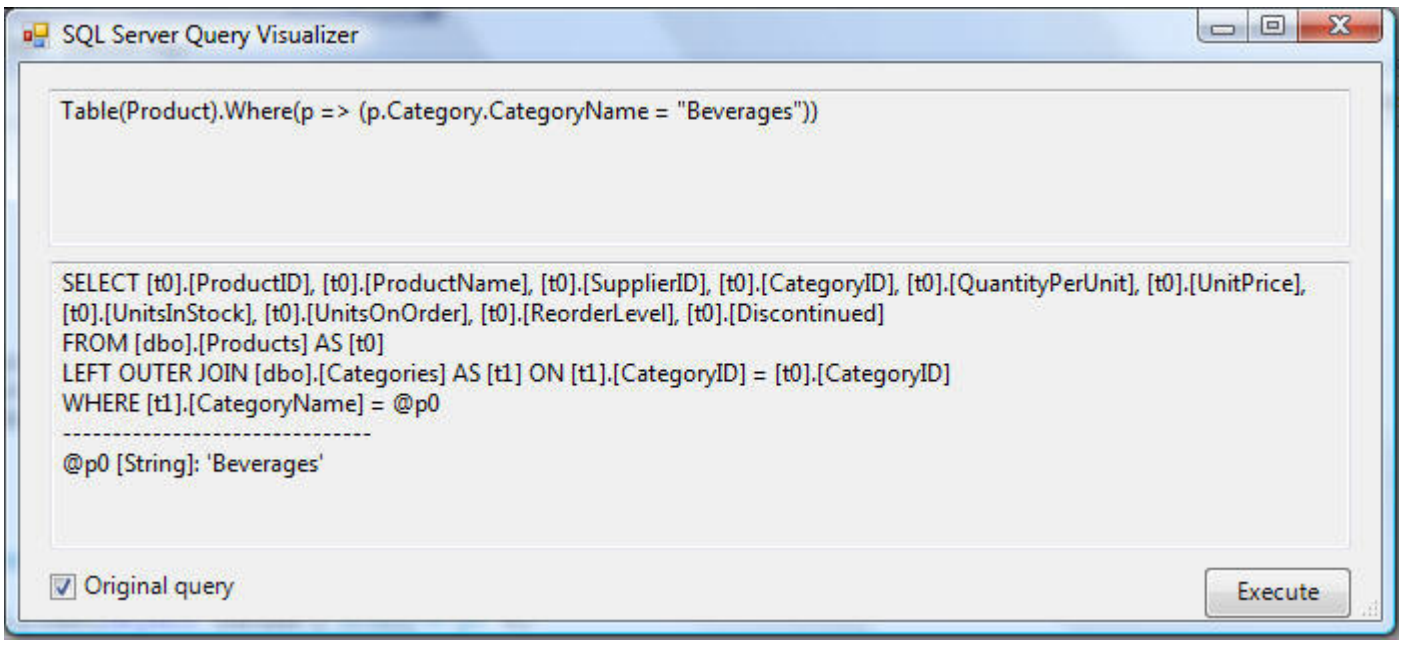
One of the biggest concerns/fears that developers new to ORMs have is "but what SQL code is it actually executing?" One of the really nice things about LINQ to SQL is that it makes it super easy to see *exactly* what SQL code it is executing when you run your application within the debugger.

Starting with Beta2 of Visual Studio 2008 you can use a new LINQ to SQL visualizer plug-in to easily see (and test out) any LINQ to SQL query expression. Simply set a breakpoint and then hover over a LINQ to SQL query and click the magnify glass to pull up its expression visualizer within the debugger:

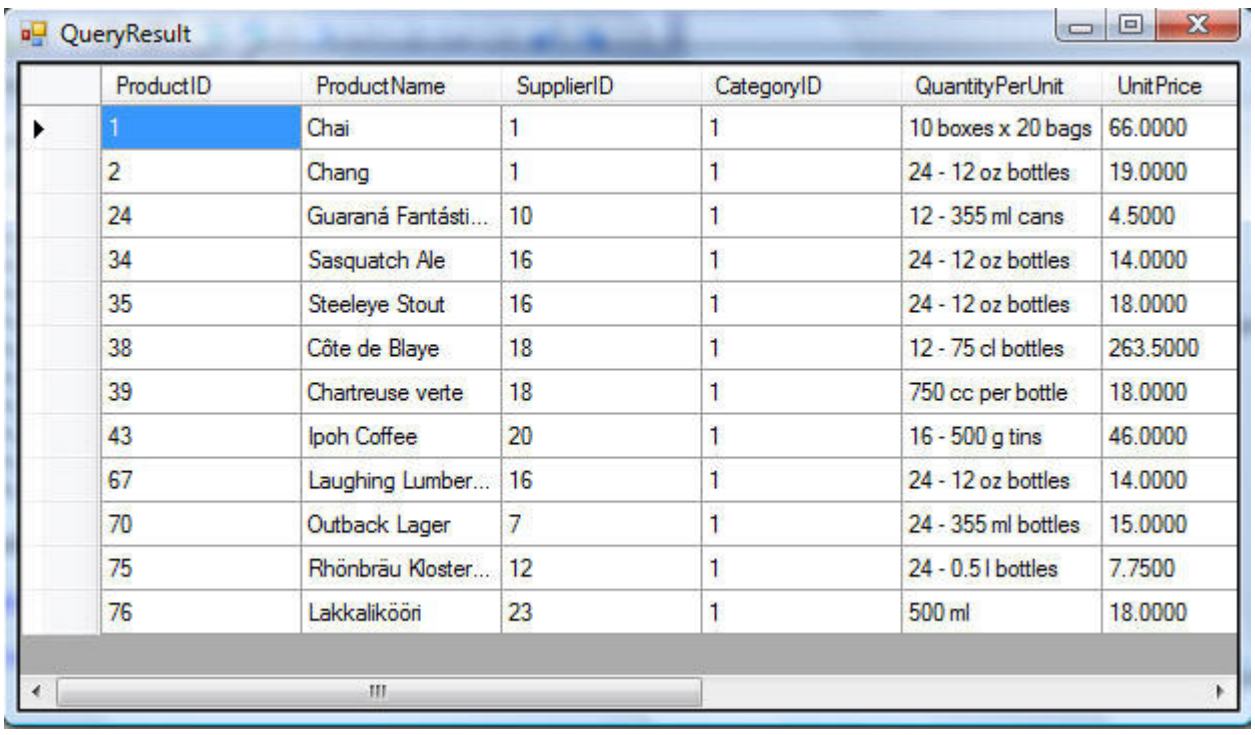
```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               select p;
```

products | {SELECT [t0].[ProductID], [t0].[ProductName], [t0].[S

This will then bring up a dialog that shows you the exact SQL that LINQ to SQL will use when executing the query to retrieve the Product objects:



If you press the "Execute" button within this dialog it will allow you to evaluate the SQL directly within the debugger and see the exact data results returned from the database:



QueryResult

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
1	Chai	1	1	10 boxes x 20 bags	66.0000
2	Chang	1	1	24 - 12 oz bottles	19.0000
24	Guaraná Fantásti...	10	1	12 - 355 ml cans	4.5000
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.0000
35	Steeleye Stout	16	1	24 - 12 oz bottles	18.0000
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5000
39	Chartreuse verte	18	1	750 cc per bottle	18.0000
43	Ipoh Coffee	20	1	16 - 500 g tins	46.0000
67	Laughing Lumber...	16	1	24 - 12 oz bottles	14.0000
70	Outback Lager	7	1	24 - 355 ml bottles	15.0000
75	Rhönbräu Kloster...	12	1	24 - 0.5 l bottles	7.7500
76	Lakkalikööri	23	1	500 ml	18.0000

This obviously makes it super easy to see precisely what SQL query logic LINQ to SQL is doing for you. Note that you can optionally override the raw SQL that LINQ to SQL executes in cases where you want to change it - although in 98% of scenarios I think you'll find that the SQL code that LINQ to SQL executes is really, really good.

Databinding LINQ to SQL Queries to ASP.NET Controls

LINQ queries return results that implement the IEnumerable interface - which is also an interface that ASP.NET server controls support to databind object. What this means is that you can databind the results of any LINQ, LINQ to SQL, or LINQ to XML query to any ASP.NET control.

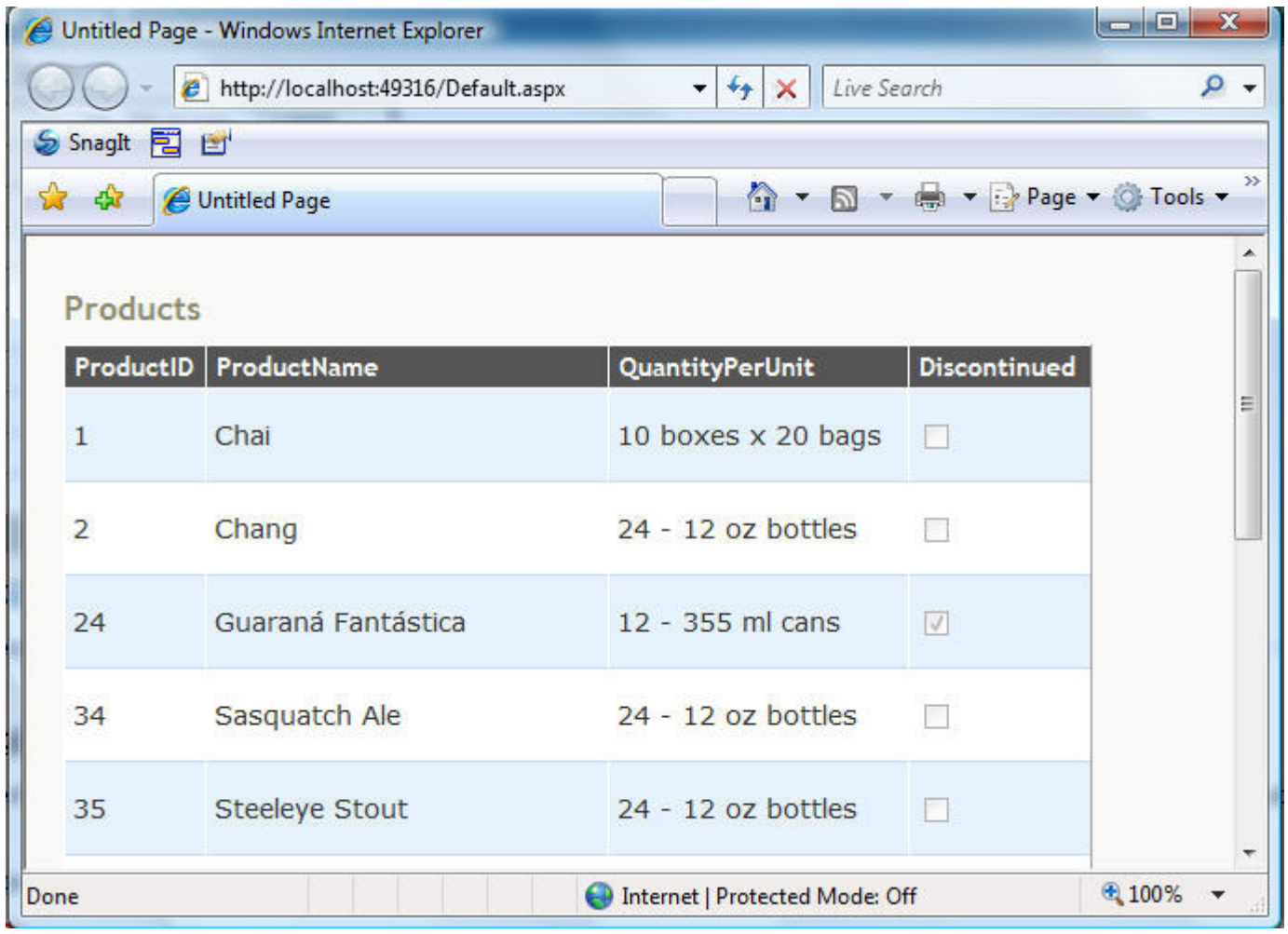
For example, we could declare an <asp:gridview> control in a .aspx page like so:

```
<h3>Products</h3>

<asp:GridView ID="GridView1"
              CssClass="gridview"
              AlternatingRowStyle-CssClass="even"
              runat="server" />
```

I could then databind the result of the LINQ to SQL query we wrote before to the GridView like so:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select p;

GridView1.DataSource = products;
GridView1.DataBind();
```

This will then generate a page that looks like below:

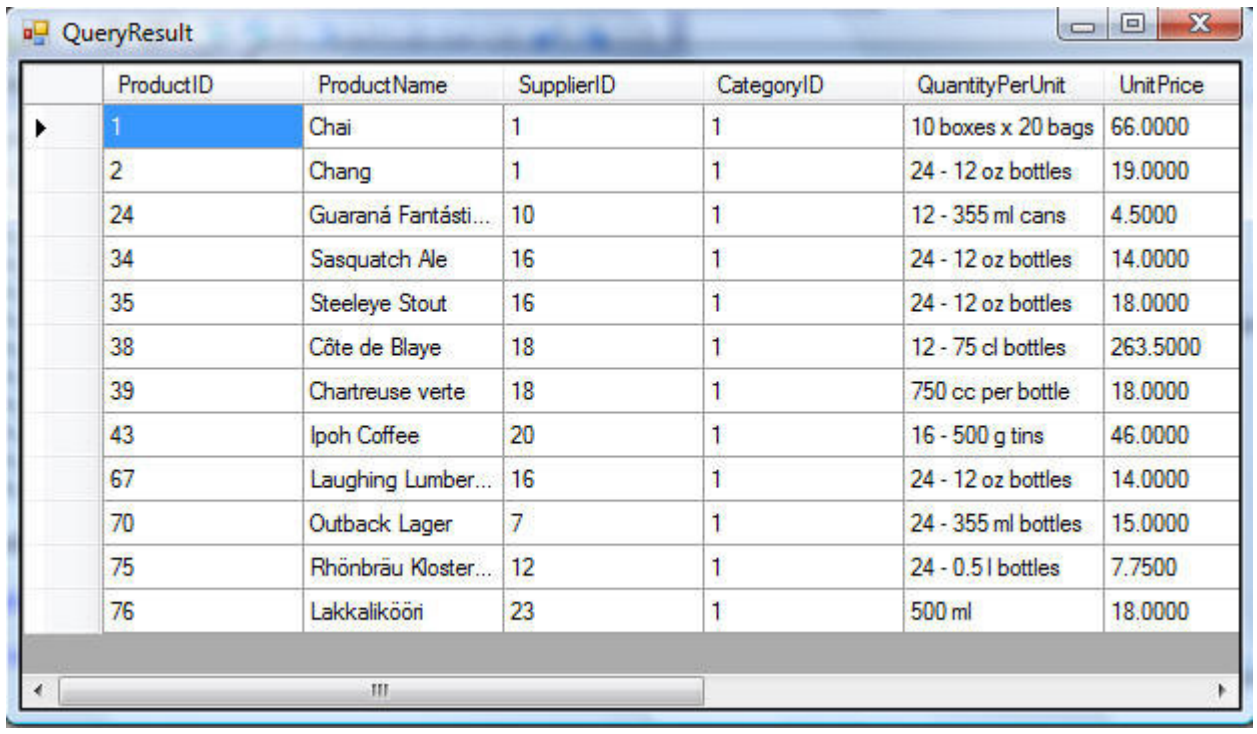


Shaping our Query Results

Right now when we are evaluating our product query, we are retrieving by default all of the column data needed to populate the Product entity classes. For example, this query to retrieve products:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select p;
```

Results in all of this data being returned:

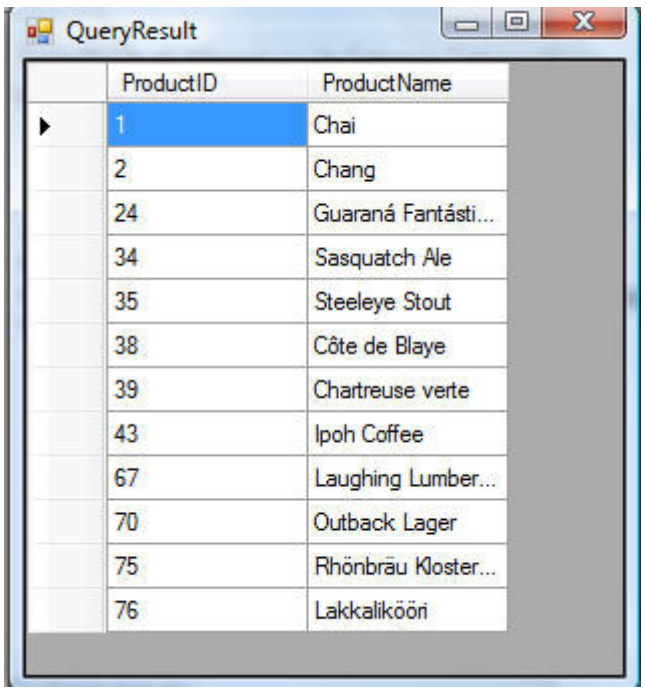


ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
1	Chai	1	1	10 boxes x 20 bags	66.0000
2	Chang	1	1	24 - 12 oz bottles	19.0000
24	Guaraná Fantásti...	10	1	12 - 355 ml cans	4.5000
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.0000
35	Steeleye Stout	16	1	24 - 12 oz bottles	18.0000
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5000
39	Chartreuse verte	18	1	750 cc per bottle	18.0000
43	Ipoh Coffee	20	1	16 - 500 g tins	46.0000
67	Laughing Lumber...	16	1	24 - 12 oz bottles	14.0000
70	Outback Lager	7	1	24 - 355 ml bottles	15.0000
75	Rhönbräu Kloster...	12	1	24 - 0.5 l bottles	7.7500
76	Lakkalikööri	23	1	500 ml	18.0000

Often we only want to return a subset of the data about each product. We can use the new data shaping features that LINQ and the new C# and VB compilers support to indicate that we only want a subset of the data by modifying our LINQ to SQL query like so:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select new
               {
                   ID = p.ProductID,
                   Name = p.ProductName
               };
```

This will result in only this data subset being returned from our database (as seen via our debug visualizer):

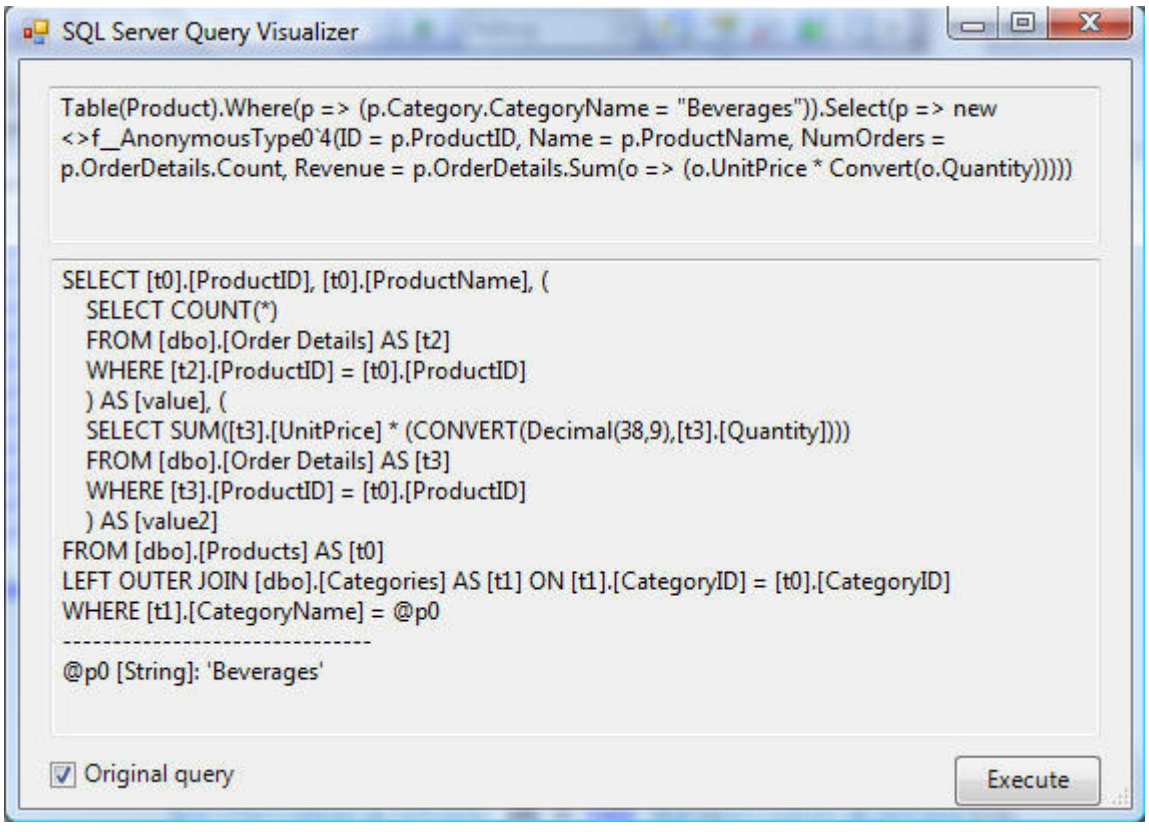


What is cool about LINQ to SQL is that I can take full advantage of my data model class associations when shaping my data. This enables me to express really useful (and very efficient) data queries. For example, the below query retrieves the ID and Name from the Product entity, the total number of orders that have been made for the Product, and then sums up the total revenue value of each of the Product's orders:

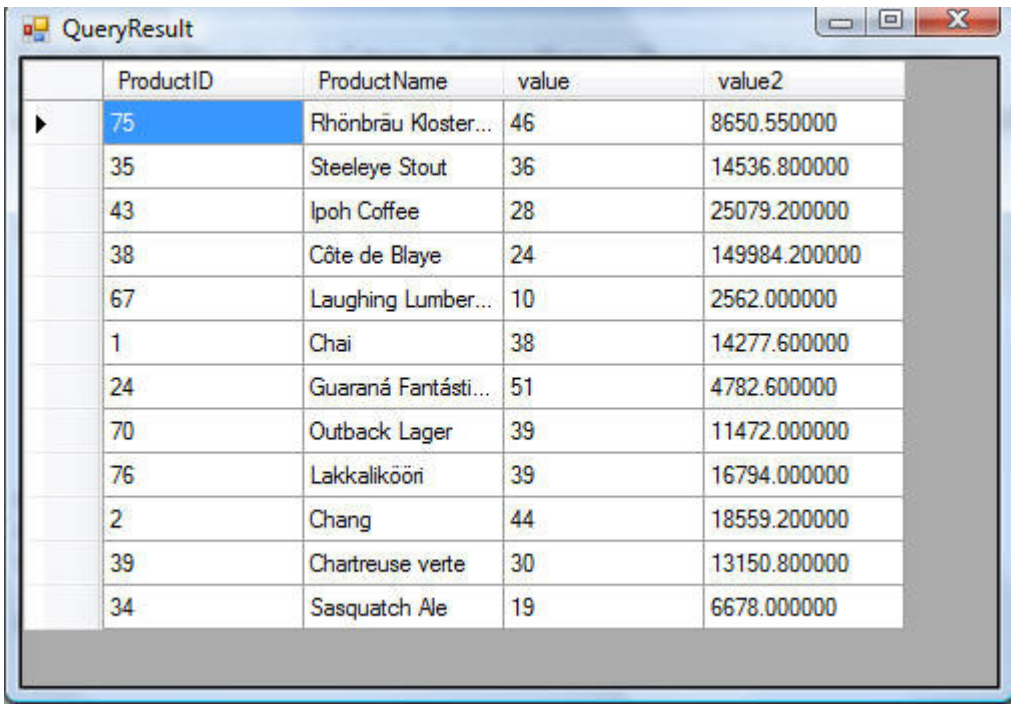
```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select new
               {
                   ID = p.ProductID,
                   Name = p.ProductName,
                   NumOrders = p.OrderDetails.Count,
                   Revenue = p.OrderDetails.Sum(o=>o.UnitPrice * o.Quantity)
               };
```

The expression to the right of the "Revenue" property above is an example of using the "Sum" extension method provided by LINQ. It takes a Lambda expression that returns the value of each product order item as an argument.

LINQ to SQL is smart and is able to convert the above LINQ expression to the below SQL when it is evaluated (as seen via our debug visualizer):

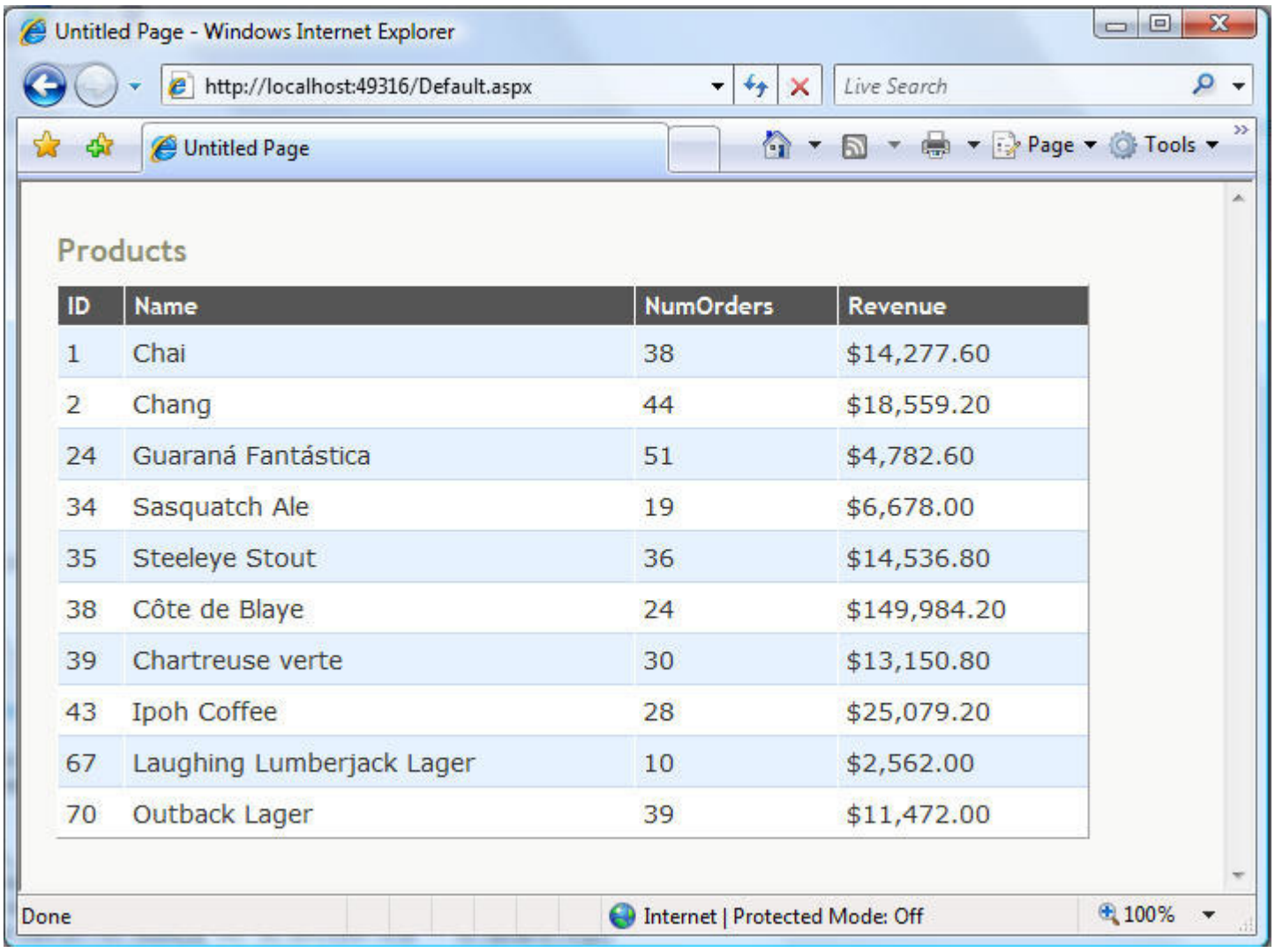


The above SQL causes all of the NumOrders and Revenue value computations to be done inside the SQL server, and results in only the below data being retrieved from the database (making it really fast):



ProductID	ProductName	value	value2
75	Rhönbräu Kloster...	46	8650.550000
35	Steeleye Stout	36	14536.800000
43	Ipoh Coffee	28	25079.200000
38	Côte de Blaye	24	149984.200000
67	Laughing Lumber...	10	2562.000000
1	Chai	38	14277.600000
24	Guaraná Fantásti...	51	4782.600000
70	Outback Lager	39	11472.000000
76	Lakkalikööri	39	16794.000000
2	Chang	44	18559.200000
39	Chartreuse verte	30	13150.800000
34	Sasquatch Ale	19	6678.000000

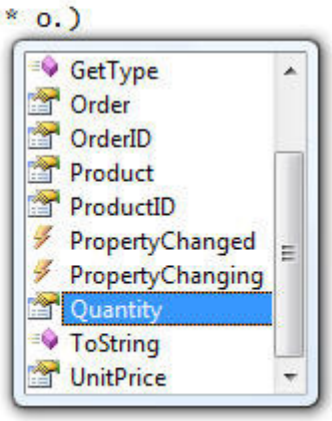
We can then databind the result sequence against our GridView control to generate pretty UI:



ID	Name	NumOrders	Revenue
1	Chai	38	$14,277.60
2	Chang	44	$18,559.20
24	Guaraná Fantástica	51	$4,782.60
34	Sasquatch Ale	19	$6,678.00
35	Steeleye Stout	36	$14,536.80
38	Côte de Blaye	24	$149,984.20
39	Chartreuse verte	30	$13,150.80
43	Ipoh Coffee	28	$25,079.20
67	Laughing Lumberjack Lager	10	$2,562.00
70	Outback Lager	39	$11,472.00

BTW - in case you were wondering, you do get full intellisense within VS 2008 when writing these types of LINQ shaping queries:

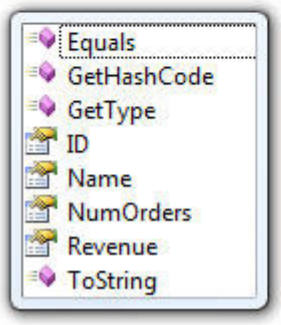
```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select new
               {
                   ID = p.ProductID,
                   Name = p.ProductName,
                   NumOrders = p.OrderDetails.Count,
                   Revenue = p.OrderDetails.Sum(o => o.UnitPrice * o.)
               };
```



In the example above I'm declaring an anonymous type that uses object initialization to shape and define the result structure. What is really cool is that VS 2008

provides full intellisense, compilation checking, and refactoring support when working against these anonymous result sequences as well:

```
NorthwindDataContext db = new NorthwindDataContext();

var products = from p in db.Products
               where p.Category.CategoryName == "Beverages"
               select new
               {
                   ID = p.ProductID,
                   Name = p.ProductName,
                   NumOrders = p.OrderDetails.Count,
                   Revenue = p.OrderDetails.Sum(o=>o.UnitPrice * o.Quantity)
               };

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



Paging our Query Results

One of the common needs in web scenarios is to be able to efficiently build data paging UI. LINQ provides built-in support for two extension methods that make this both easy and efficient - the Skip() and Take() methods.

We can use the Skip() and Take() methods below to indicate that we only want to return 10 product objects - starting at an initial product row that we specify as a parameter argument:

```
void BindProducts(int startRow)
{
    NorthwindDataContext db = new NorthwindDataContext();

    var products = from p in db.Products
                   where p.OrderDetails.Count > 2
                   select new
                   {
                       ID = p.ProductID,
                       Name = p.ProductName,
                       NumOrders = p.OrderDetails.Count,
                       Revenue = p.OrderDetails.Sum(o => o.UnitPrice * o.Quantity)
                   };

    GridView1.DataSource = products.Skip(startRow).Take(10);
    GridView1.DataBind();
}
```

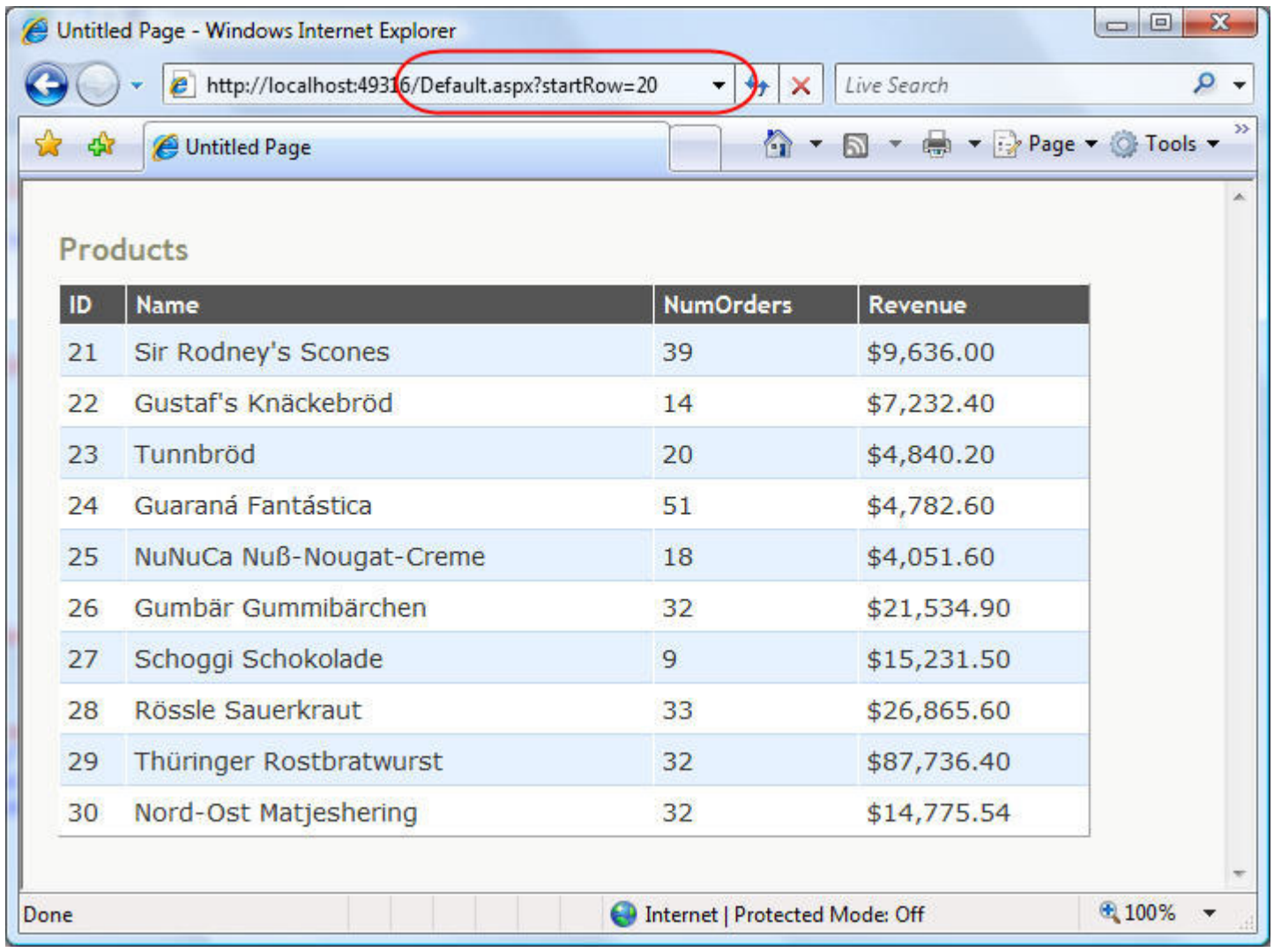
Note above how I did not add the Skip() and Take() operator on the initial products query declaration - but instead added it later to the query (when binding it to my GridView datasource). People often ask me "but doesn't this mean that the query first grabs all the data from the database and then does the paging in the middle tier (which is bad)?" No. The reason is that LINQ uses a deferred execution model - which means that the query doesn't actually execute until you try and iterate over the results.

One of the benefits of this deferred execution model is that it enables you to nicely compose queries across multiple code statements (which improves code readability). It also enables you to compose queries out of other queries - which enables some very flexible query composition and re-use scenarios.

Once I have the BindProduct() method defined above, I can write the code below in my page to retrieve the starting index from the querystring and cause the products to be paged and displayed in the gridview:

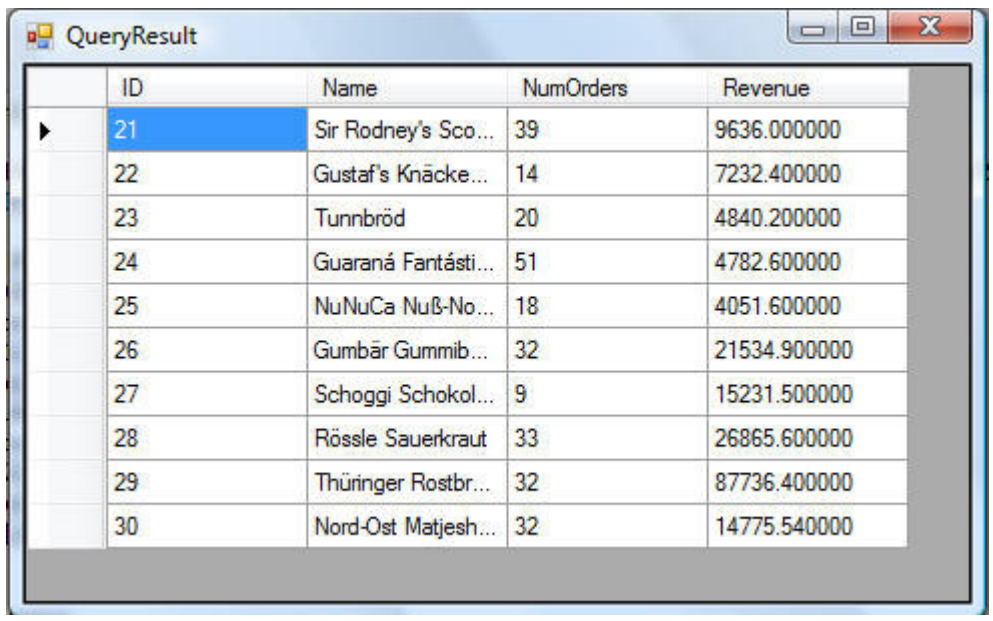
```
void Page_Load(object sender, EventArgs e)
{
    int startRow = Convert.ToInt32(Request.QueryString["startRow"]);

    BindProducts(startRow);
}
```

This will then give us a products page, filtered to list only those products with more than 5 orders, showing dynamically computed product data, and which is pageable via a querystring argument:



ID	Name	NumOrders	Revenue
21	Sir Rodney's Scones	39	$9,636.00
22	Gustaf's Knäckebröd	14	$7,232.40
23	Tunnbröd	20	$4,840.20
24	Guaraná Fantástica	51	$4,782.60
25	NuNuCa Nuß-Nougat-Creme	18	$4,051.60
26	Gumbär Gummibärchen	32	$21,534.90
27	Schoggi Schokolade	9	$15,231.50
28	Rössle Sauerkraut	33	$26,865.60
29	Thüringer Rostbratwurst	32	$87,736.40
30	Nord-Ost Matjeshering	32	$14,775.54

Note: When working against SQL 2005, LINQ to SQL will use the ROW_NUMBER() SQL function to perform all of the data paging logic in the database. This ensures that only the 10 rows of data we want in the current page view are returned from the database when we execute the above code:



ID	Name	NumOrders	Revenue
21	Sir Rodney's Sco...	39	9636.000000
22	Gustaf's Knäcke...	14	7232.400000
23	Tunnbröd	20	4840.200000
24	Guaraná Fantásti...	51	4782.600000
25	NuNuCa Nuß-No...	18	4051.600000
26	Gumbär Gummib...	32	21534.900000
27	Schoggi Schokol...	9	15231.500000
28	Rössle Sauerkraut	33	26865.600000
29	Thüringer Rostbr...	32	87736.400000
30	Nord-Ost Matjesh...	32	14775.540000

This makes it efficient and easy to page over large data sequences.

Summary

Hopefully the above walkthrough provides a good overview of some of the cool data query opportunities that LINQ to SQL provides. To learn more about LINQ expressions and the new language syntax supported by the C# and VB compilers with VS 2008, please read these earlier posts of mine:

- Automatic Properties, Object Initializer and Collection Initializers
- Extension Methods
- Lambda Expressions
- Query Syntax
- Anonymous Types

In my next post in this LINQ to SQL series I'll cover how we can cleanly add validation logic to our data model classes, and demonstrate how we can use it to encapsulate business logic that executes every time we update, insert, or delete our data. I'll then cover more advanced lazy and eager loading query scenarios, how to use the new <asp:LINQDataSource> control to support declarative databinding of ASP.NET controls, optimistic concurrency error resolution, and more.

Hope this helps,

Scott

Published Friday, June 29, 2007 1:11 AM by ScottGu
Filed under: ASP.NET, Visual Studio, .NET, LINQ, Data

Comments

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 5:13 AM by elixir

I love how you combined functional programming, query language and normal imperative programming into the whole language model and do so elegantly

now, even subsonic won't be neccessary :)

keep up the great work!

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 5:53 AM by Koen

Hi,

Will there be a LINQDataSource control also? And what if I want to have a dynamic query (where clause depends on user input, e.g. filter controls)?

Functional programming « chakrit

Friday, June 29, 2007 6:06 AM by Functional programming « chakrit

Pingback from Functional programming « chakrit

LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 6:07 AM by DotNetKicks.com

You've been kicked (a good thing) - Trackback from DotNetKicks.com

LINQ to SQL (Part 3 - Querying our Database) - ScottGu's Blog

Friday, June 29, 2007 6:08 AM by LINQ to SQL (Part 3 - Querying our Database) - ScottGu's Blog

Pingback from LINQ to SQL (Part 3 - Querying our Database) - ScottGu's Blog

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 6:13 AM by Manuel Abadia

Scott,

some time ago I wrote a DebuggerVisualizer for LINQ expressions that is available here:

www.manuelabadia.com/.../PermaLink,guid,9160035f-490f-46bd-ab55-516b5c7545af.aspx

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 6:23 AM by Juan

Hi Scott,

Great series of posts!. Very excited about this!,

Can't wait to start using it in our apps!

Cheers,

Juan

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 6:56 AM by Chris Moseley

Hi Scott,

This looks fantastic but I have a few questions:

Using your paging method, will I have to implement my own custom pager or is there a way to get the inbuilt GridView pager to behave correctly using this method of server-side paging?

Thanks, Chris

Also, If I want to use Link To Sql to bind to a GridView, how do I implement column sorting? I'll need to be able to add a dynamic field name into the select query...

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 8:03 AM by Muhammad Adnan

would need of ado.net and sql get remove and what about sql injection security

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 8:22 AM by Akaka

Scott,

I have one question about data paging on Gridview , as I know that gridview is also have feature to paging data, How can we use gridview paging feature works with LINQ.

Thank for your post.

Akaka

Akaka

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 8:37 AM by Peter Bridger

Hi Scott, apologies to post off topic, but I was wondering if you recevied the e-mail I sent to your Microsoft address yesterday?

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 9:24 AM by Rob Mathieson

Hi Scott

When submitting an object to the database for updating, is it possible to manipulate the query to include logic?

For example, I may wish to insert or update, based on whether or not that item's id is already in a database table.

Also, just want to say that the stuff you guys are doing is fantastic. We are starting to get to a point where we can produce applications at a speed that has never been possible before. Well done!

Regards

Rob

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 10:41 AM by Mike Loll

Scott - I am tinkering with Orcas and the 3.5 beta. I love linq, but what I want to do is use linq to sql to get my data, but I use interfaces in my business layer. So I might have an IProduct which has various properties and methods. Using an architecture like this, would I be better of using linq to sql to populate some new domain object that implements my interface, or is there some way to marry my interface with what linq to sql will give me?

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 11:08 AM by Josh Stodola

Hi Scott, thanks for posting this. I have been quite intrigued by LINQ, but unfortunately the provided examples somewhat leave me at a loss. Please show some love to the millions of VB programmers by providing examples in both languages! C# should never considered superior!

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:20 PM by ScottGu

Hi Koen,

There will be a <asp:linqdatasource> control, and I'll be blogging about it more over the next few weeks.

You can also use dynamic queries with LINQ - this allows end-users to dynamically specify what columns/filters they want and then build it via code. I'll try and put a sample together that shows this in the future.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:25 PM by ScottGu

Hi Chris,

The good news is that you can point the GridView at the <asp:linqdatasource> control, and get automatic server-side paging with it using the GridView UI. We also now have a <asp:datapager> control that you can use with other controls (like the new <asp:listview>) to enable server-side paging UI for them as well.

I'll show more how to use the linqdatasource control in the future. I wanted to show the code-approach first so that people wouldn't get confused about LINQ only being a databinding scenario - the great news is that you can use it with both code and declarative binding and have it work great everywhere.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:27 PM by Bryan

Scott:

I just love this stuff! I can't wait for VS 2008 Beta 2 to be made available!

I attended sessions at Tech Ed that went through all of these LINQ to SQL features, but it is good to go over it all again at a pace I can keep up with!

And the opposite of Lazy Loading is Eager Loading...makes sense.

Thanks to your team for all of their hard work!

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:28 PM by John

Scott,

I noticed that one of the lines is

WHERE p.OrderDetails.Count > 5

This is a little confusing. LINQ is structured to be like SQL, but it seems to me, that if we wanted it to be similar to SQL, it would have been written:

HAVING p.OrderDetails.Count > 5

Does LINQ contain the "HAVING" sql keyword?

Thanks.

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:30 PM by ScottGu

Hi Muhammad,

The good news is that LINQ to SQL protects you against SQL injection attacks. All parameters are automatically SQL encoded for you so that you don't have to-do anything to

make them safe.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:33 PM by ScottGu

Hi Rob,

I'll be posting about the validation logic shortly. You should be able to perform the conditional logic you are after to check whether and order (or its order items) are already in the database prior to completing the transaction.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 12:35 PM by ScottGu

Hi Mike,

There are a couple of approaches you can use if you want to use your own interfaces:

1) Add the interfaces to your LINQ to SQL data model classes. The LINQ to SQL classes are partial classes - which means you could add the interface directly to them.

2) Modify your LINQ queries to return not the origional model types, but rather separate objects that implement your interfaces. You can control this via the select clause in your LINQ to SQL query. This blog post shows a little how to-do this: weblogs.asp.net/.../new-orcas-language-feature-query-syntax.aspx

Hope this helps,

Scott

New and Notable 176

Friday, June 29, 2007 1:23 PM by Sam Gentile

TGIF!! I am super busy right now designing a multi-CPU/multi-threaded Parallel Calculation Engine and

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 1:37 PM by Sean Chase

Hi Scott,

How do you envision a traditional "n-tier" / "layered" application approach using these technologies. For example, *today* many of us don't want to see SQL outside of a DAL or DBMS, calls from thee UI layer go through a business interface (or facade layer) and then possibly 1 or more calls from the business interface layer into a business logic layer then DAL. You get the jist. :-) Perhaps I'm missing something. Do you think it will be "best practice" to use LINQ query syntax at the UI layer such as an ASP.NET application or would that be more reserved for a DAL layer.

Appreciate the input.

Sean

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 1:54 PM by MIke

What about Stored Procedures? Are we expected to give read/write rights to all of our tables? What about query performance? This is also the problem I have with (n) Hibernate or Rails. I can't believe people aren't demanding more control over the sql statements generated. It looks awesome, but my company would never give users to update tables directly. We currently use iBatis.Net which gives the OR mapping, but lets us still have complete control over the sql and permissions (at the cost of development time).

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 4:41 PM by ScottGu

Hi Mike,

The good news is that you can use Stored Procedures with LINQ to SQL, and can design your data model so that updates, inserts, deletes all go through SPROCs. This ends up being transparent to anyone consumer of a data model.

You can also use SPROCs for queries, so you can integrate those with your data models as well. I have a small sample of this in my blog post here: weblogs.asp.net/.../using-linq-to-sql-part-1.aspx

I'll be doing a dedicated blog post on using SPROCs with LINQ to SQL in the next few weeks where I'll go into more detail on this.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 4:49 PM by ScottGu

Hi Sean,

There are a couple of ways you can use LINQ to SQL. One is to use it to effectively replace your DAL layer, and encapsulate a separate business layer between your UI and it.

One of the nice things about LINQ to SQL, though, is that it enables you to build a pretty flexible object oriented layer for your data/business logic. Specifically, it provides a very clean way to add business logic and rules in your entity classes. The entity classes for LINQ to SQL also support persistance ignorance and flexible inheritance (no base class required). I'll cover some of these in blog posts over the next few weeks. I expect a lot of developers will use these features to merge their data and business layers together for many projects (which for 90% of projects is just fine).

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 4:58 PM by Alan

Scott,

Regarding paging, you say that if we're using SQL2k5, that ROW_NUMBER() will be used to facilitate paging. Can LINQ be used with SQL2k and still have built-in paging support? If so, what type of SQL pattern is it using?

Also, for the ROW_NUMBER() approach, I've read things which say that in large volumes, ROW_NUMBER() is quite slow. Is ROW_NUMBER() still appropriate when paging sets of millions of records? What performant alternatives are there?

I like that LINQ appears to know the right type of JOINs to perform-- this is good because most developers don't ;) It's usually left to our DBAs to optimize the mess. Does LINQ allow you to programatically specify join hints in the cases where they're useful? Or would we need to go the custom sproc route and bypass LINQ fanciness?

Can you describe the domain of cases where LINQ-genereated SQL might not be the most efficient? In general, LINQ appears to be a great way to prototype at the very least.. but in high volume production scenarios, we'd like to make sure that we're not going to have to go through and tweak (beyond propery indexing) the LINQ-generated SQL everywhere due to performance reasons.

-Alan

BTW: Assuming the performance holds up in production scenarios, this approach really beats the hell out of J2EE's EJB/CMP approach ;)

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 8:39 PM by Ben Hayat

Hi Scott;

As always, great article. Can't tell you how valuable you articles are. This is well spent time educating us the way you write.

One question: I know LINQ and all the related parts have been changing and getting ready for production and it's hard to write our production code against a moving target. I like to know if starting Beta 2 of VS2008, the feature list have been locked and we can begin to write code against LINQ and ASP.Net 3.5?

Thanks again!

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 9:23 PM by Ben Hayat

Hi Scott;

>>I expect a lot of developers will use these features to merge their data and business layers together for many projects (which for 90% of projects is just fine)<<

This was a comment you made to Sean, and I had been thinking about the exact thing you said prior to reading your post. Seems like the DAL is already done for you when VS creates the ORM DataContext, we just do the rest in the Business layer using LINQ.

..Ben

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, June 29, 2007 11:38 PM by Kangnoz

Scott,

You said,"When working against SQL 2005, LINQ to SQL will use the ROW_NUMBER() SQL function to perform all of the data paging logic in the database. This ensures that only the 10 rows of data we want in the current page view are returned from the database when we execute the above code."

so current page view has 10 rows at most,if we will render >10 rows,Linq To SQL can't do it? The function of Linq To SQL is limited.

Sorry ,I am Chinese. My English is not good.

Thank you.

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 12:08 AM by ScottGu

Hi Kangnoz,

You can have any sized page of data that you want. In the code above I am passing 10 to the Take() method - which is why it is returning 10 rows of data. You could change this to be any number you want.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 12:09 AM by Vikram

Hi scott,

This was really good example. Can we also have an example on how to use LINQ with stored procedures

Thanks

http://www.vikramlakhotia.com

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 12:09 AM by ScottGu

Hi Ben,

The Beta2 release of .NET 3.5 and VS 2008 should be fairly baked in terms of feaure set - I wouldn't expect LINQ and ASP.NET to change much after that (a few minor tweaks - but nothing substantial). So I thjink you could start then to write applications with it and not be worried about having to change much for the final release.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 9:05 AM by Juan María

Hi!

Great post,

you can read it in spanish here:

thinkingindotnet.wordpress.com/.../linq-to-sql-3%c2%aa-parte-consultando-la-base-de-datos

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 12:13 PM by Eric

Hi Scott,

I have the same question with Chris, about sorting the data through LINQ dynamically.

The paging feature is very convenient for us web developers, and what should we do if we want sort the data in a gridview depending on which column header the user has click?

Thanks

re: LINQ to SQL (Part 3 - Querying our Database)

Saturday, June 30, 2007 8:09 PM by Matt

Is Microsoft striving to support other databases besides SQL Server? How about Oracle?

re: LINQ to SQL (Part 3 - Querying our Database)

Sunday, July 01, 2007 2:54 AM by Microogle

Hi Scott,

I have the same question with Chris, about sorting the data through LINQ dynamically.

The paging feature is very convenient for us web developers, and what should we do if we want sort the data in a gridview depending on which column header the user has click?

re: LINQ to SQL (Part 3 - Querying our Database)

Sunday, July 01, 2007 10:32 AM by JasonBSteele

Hi Scott,

Loving the series, and I am looking forward to the next part which I condsider to be the most useful.

I share some of the other posters concerns about how this can be used in an n-tier project and have read your responses with interest.

For obvious reasons examples of implementing business logic can be pretty trivial (e.g. is ToDate > FromDate). I think it would be of real value if something a bit more substantial could be shown that involves more than one enitity. For example: Is this a new Order, and if so deduct quantities from the Stock table for each Order Detail checking that they do not fall below 0.

Just a suggestion ;)

Thanks,

Jason

Enlaces .NET 01-07-2007

Sunday, July 01, 2007 3:07 PM by OberData

TreeView con Ajax .NETValidaciones con Enterprise Library 3.xLINQ to SQL (Part 3 - Querying our Database)Busqueda...

re: LINQ to SQL (Part 3 - Querying our Database)

Sunday, July 01, 2007 11:33 PM by Sambo

Hi Scott,

This is a great post of LINQ. Could you tell me, when the VS 2008 release?

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 2:54 AM by Softlion

Hi Scott,

99% of the projects I've worked on use 90% of simple sproc using a NOLOCK isolation level, and 10% of advanced sproc which uses mixed isolation levels to prevent dead locks. An advanced sproc may be written using C# and calls to smaller sprocs. But LOCK controls (NOLOCK, HOLDLOCK, ROWLOCK, ...) are finely tuned.

Will LINQ be smart enough or soft enough to let us retain control over these ?

Also what about search facilities ? (indexed searches ?).

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 5:31 AM by Ilias Hossain

But where is the sorting facilities. Suppose i want to short the above grid with Product name of Product table

and

Quentity, price fields of the OrderDetail table. All these field are in on the same grid. Is there any way.

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 11:28 AM by ScottGu

Hi Sambo,

VS 2008 Beta2 will ship in a few weeks. The final release of VS 2008 and .NET 3.5 will then ship in a few months.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 11:30 AM by ScottGu

Hi Microogle and Eric,

I'll be covering sorting (including dynamic scenarios involving the GridView) in a future blog post. There is an orderby keyword that LINQ supports that makes this easy to support. When you point a GridView control at a LinqDataSource the paging and sorting support works automatically.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 11:32 AM by ScottGu

Hi Matt,

LINQ itself is a datasource agnostic - so you can use it against in-memory objects, XML files, and any relational database.

On the database front, what you need is a data provider that performs the appropriate mappings between LINQ queries and the database. Both LINQ to Entities and LINQ to NHibernate will support Oracle.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 11:33 AM by ScottGu

Hi Jason,

Thanks for the suggestion - I'll try and incorporate that scenario into my validation logic post.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 02, 2007 5:42 PM by Zack Owens

oooo... .NET Framework 3.5 June CTP is out! (www.microsoft.com/.../details.aspx)

I'm thinking that Visual Studio 2008 CTP is not far behind. Is this a fair assumption?

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 03, 2007 12:07 PM by Boris Yeltsin's Zombie

Hey Scott,

You didn't get back to my previous comment on your blog.

Will you guys release the source to LINQ to SQL?

You guys did a fabulous job releasing the source for the SQL Server Membership Provider which let people quickly modify the code and connect to whatever flavour of database they were using in-house.

The source for LINQ to SQL would let people quickly go through and replace out all the SQL statements with the equivalents for other DBs.

I realise those in the hierarchy above you might feel that it reduces the MS lock-in, but there are plenty of us out there that love .NET and Visual Studio and are happy to buy Windows servers all day long, but where SQL Server is not the right database for the task, or just not an option because of other application stacks that are in use at the organization.

I badly want to use LINQ to SQL exclusively and drop all the other DALs I'm using. Ease my pain and release the source. Pretty please?

- BY's Zombie

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 03, 2007 3:14 PM by Alan

Scott, just a quick ping to see if you had any comments on my reply above regarding ROW_NUMBER() and performance issues.

-Alan

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 03, 2007 3:55 PM by shashi

How to do Insert, Update, Delete. Do I need individual table adapters like vs2005 DataSets?

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 03, 2007 4:12 PM by Judah

Scott,

First, excellent post. Functional concepts are like deferred execution are really making this all look very elegant.

I've a question: does LINQ to SQL do anything for updating the data? Say I bind the data to a WinForms DataGridView. I make a change to the data in the UI - what happens then? Does LINQ to SQL play any role there?

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 04, 2007 9:03 PM by ScottGu

Hi Zack,

Yep - Beta2 isn't that far away. I'd probably recommend holding off for that - since the quality will be much higher than the CTP, and it shouldn't be more than about 3 weeks away.... :-)

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 04, 2007 9:07 PM by ScottGu

Hi Boris,

Sorry for the delay in getting back to you - for some reason your first comment didn't go through (the spam filter might have nuked it!).

To answer your question - I'm hoping we'll be able to release the source for LINQ to SQL - I definitely think that is something we should do.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 04, 2007 9:09 PM by ScottGu

Hi Judah,

Yep - LINQ to SQL supports not just querying data, but also applying updates back to the database. It is really easy to-do this both with WinForms (using say a DataGridView) as well as with ASP.NET. I'll be covering this more in the next few weeks.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 04, 2007 9:14 PM by ScottGu

Hi Alan,

Sorry for the delay in getting back to you!

I've heard relatively good things about ROW_NUMBER() performance, but in general if you have extremely large result sets (millions of rows) then you might want to look hard at how you page it. Note that by result-set I don't mean millions of rows in the table - but rather millions of rows in the table that match the where filter and are to be paged in the UI.

LINQ to SQL allows you to call SPROCs (or embed custom SQL) for totally custom queries - so that is an option if you find ROW_NUMBER() causes problems.

I don't know the query optmizer in LINQ to SQL well enough to know the exact domain models that are best with LINQ to SQL. But Matt Warren from the LINQ to SQL team has a great blog here: blogs.msdn.com/.../default.aspx and would be a great person to ask for more suggestions on this.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 04, 2007 9:16 PM by ScottGu

Hi Shashi,

My next set of posts will start to cover insert, update and delete scenarios for LINQ to SQL. Here is a post I did earlier that shows a few syntax examples on how to-do them: weblogs.asp.net/.../using-linq-to-sql-part-1.aspx

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Thursday, July 05, 2007 10:10 AM by Brendan Enrick

Scott,

Great post. It is very helpful.

When do you think we may look forward to the next post in this series?

Thanks again,

Brendan

re: LINQ to SQL (Part 3 - Querying our Database)

Thursday, July 05, 2007 4:50 PM by ScottGu

Hi Brendan,

I'm hoping to have a newer post on LINQ to SQL up this weekend or early next week.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Thursday, July 05, 2007 10:13 PM by Brian Anderson

Thanks for another great post. LINQ looks great and is a long awaited addition to the stack. After doing some testing with LINQ and the SQL mapping tool I realized that I can't change the namespaces of my generated classes. Are there any plans to allow this before RTM?

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, July 06, 2007 10:08 AM by ScottGu

Hi Brian,

I don't have the older builds of LINQ to SQL installed so can't check if this behavior is new, but with Beta2 builds there are two properties on the root designer called "Entity Namespace" and "Context Namespace". You can set these in order to change the namespaces for the data entitity classes generated as well as the root DataContext class generated.

If you don't set these, then the designer will default to using the project namespace.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, July 06, 2007 10:15 AM by Boris Yeltsin's Zombie

Thanks for the reply Scott. Hope you can release that LINQ to SQL source.

p.s. You work too hard. I talked to your boss - he said you can have next week off.

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 10, 2007 4:05 PM by Steve

Scott,

You previously mentioned that you will be discussing many to many relationships and how to work around them. Can you point me to something that might get me started or give me confidence that I can migrate my application from nHibernate to LINQ to SQL? I have an existing ASP.net 2.0 app that has around 10 tables but 2 of them have a m-m relationship. I don't want to start learning this tool if it will be a dead end due to lack of m-m support.

Thanks,

Steve

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 11, 2007 5:20 AM by ScottGu

Hi Steve,

Here is a good blog post that talks about how you can implement M:M relationships using LINQ to SQL: blogs.msdn.com/.../how-to-implement-a-many-to-many-relationship-using-linq-to-sql.aspx

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 11, 2007 10:00 AM by Enrock

I wonder if linq can link any database without changing it,how can it deal with different SQLs,you know ,the SQL translater of every database is improving

LINQ to SQL (Part 4 - Updating our Database)

Wednesday, July 11, 2007 1:33 PM 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

UPDATE: Table Joins in LINQ - Simplify it! « Ramblings of the Sleepy…

Friday, July 13, 2007 1:22 PM by UPDATE: Table Joins in LINQ - Simplify it! « Ramblings of the Sleepy…

Pingback from UPDATE: Table Joins in LINQ - Simplify it! « Ramblings of the Sleepy…

re: LINQ to SQL (Part 3 - Querying our Database)

Sunday, July 15, 2007 4:10 AM by Jarle Nygård

As far as I can see LINQ is based on mapping tables and classes more or less 1:1, but how about mapping classes and table patterns 1:1? In our database we have a lot of "codetable" tables that follows the same patterns; ie. they have the same columns and "sub-tables" (connected). So the classes would basically be the same, within a specific pattern. We have 4 different "codetable" patterns (hierarchial etc.).

Is this a scenario you already support or are planning to support?

re: LINQ to SQL (Part 3 - Querying our Database)

Sunday, July 15, 2007 6:20 PM by ScottGu

Hi Jarle,

>>>>>> As far as I can see LINQ is based on mapping tables and classes more or less 1:1, but how about mapping classes and table patterns 1:1? In our database we have a lot of "codetable" tables that follows the same patterns; ie. they have the same columns and "sub-tables" (connected). So the classes would basically be the same, within a specific pattern. We have 4 different "codetable" patterns (hierarchial etc.).

Can you provide more details on the schema model you are using with these table patterns?

LINQ to SQL does support the ability to use the single-table inheritance pattern, which allows you to model sub-classes based on a discriminator key in a table. That sounds loosely like what you are looking for above - but I'm not entirely sure.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 17, 2007 9:08 AM by Amr Elsehemy

Hello Scott,

Where can I find the code samples of this article.

Thanks,

Amr

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 17, 2007 3:18 PM by Lucas Krause

I have beta 2 and i can't see the linq to sql visualizer, i took the c# samples folder build the express viewer, then drop it in both users\document\VS 2008\Visualizers, and program files\vs 9.0\common7\debugger\visualizers\ and it doesn't show up, is there a setting inside VS or a tweak needed for this?

i am running vista, have VS 2005SP1, and VS 2008 beta 2, with Team explorer as well.

thanks!

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 17, 2007 4:12 PM by ScottGu

Hi Lucas,

To install the debug visualizer, you'll want to follow these following steps:

1. Unzip <VSOrcas Install Dir>\Samples\CShaprLinqSamples.zip
2. Build the query visualizer project
3. Copy the DLL to <VSOrcas Install Dir>\Common7\Packages\Debugger\Visualizers\

You'll then want to shutdown and restart VS 2008. It should then work.

Thanks,

Scott

LINQ to SQL (パート 3 - データベースの検索)

Friday, July 20, 2007 8:19 AM by Chica's Blog

LINQ to SQL (パート 3 - データベースの検索)

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 23, 2007 10:51 AM by discovia

Whats the best way to return the results you get through LINQ as xml, or JSON?

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 24, 2007 1:59 PM by ScottGu

Hi discovia,

There are a couple of ways to transfer the result of a LINQ query into XML. The best way is to probably use LINQ to XQL - which is built-in with .NET 3.5. I'll try and do a few samples that show this off in the near future.

Thanks,

Scott

» Functional programming « chakrit-My 1983

Wednesday, July 25, 2007 2:24 AM by » Functional programming « chakrit-My 1983

Pingback from » Functional programming « chakrit-My 1983

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, July 25, 2007 2:41 PM by Discovia

Thanks Scott,

You do great posts. It'll be real nice to see a write up on using LINQ for a web service as consumer and provider.

It's be great to see it being used from the client, with nothing to do with asp.net controls. The controls are great for quick impressive demos, but personally I'm not a fan. It'd be great to see how it would be used with AJAX controls like extjs grid, etc.

re: LINQ to SQL (Part 3 - Querying our Database)

Monday, July 30, 2007 8:06 PM by Mark

Say I want to cache some lookup data from SQL Server and I want to use LINQ to do the query. Is there a way to use SQL Notification Services to be notified when data retrieved with LINQ to SQL has been updated in the database?

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 31, 2007 4:36 AM by ScottGu

Hi Mark,

>>>>>>> Say I want to cache some lookup data from SQL Server and I want to use LINQ to do the query. Is there a way to use SQL Notification Services to be notified when data retrieved with LINQ to SQL has been updated in the database?

There isn't automatic support for this scenario today, but you could manually handle it. I'll put it on my list of things to post about in the future.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 31, 2007 8:10 AM by Chad

Scott,

When writing static methods to access data, does the deferred query "feature" work? See the code below, is this safe to run? or will it be extremely slow in large query situations?

```
public static IQueryable<Lesson> SelectAllLessons()
{
eigoDB db = new eigoDB();
var lessons = from l in db.Lessons
 orderby l.DateAdded descending
 select l;
return lessons;
}
public static IQueryable<Lesson> SelectAllLessons(int skip, int take)
{
return SelectAllLessons().Skip(skip).Take(take);
}
```

Notice how I'm simply reusing the static method, and adding .Skip/.Take to it? It works, but I'd like to know if I should not be doing it this way.

Thanks!

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 31, 2007 10:54 AM by ScottGu

Hi Chad,

The deferred query exection still works fine in your sample above. It won't access the database until someone starts trying to access the database - which allows you to cleanly compose Skip() and Take() operators like you are above.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Tuesday, July 31, 2007 11:53 AM by Mark

Thanks for the quick response Scott. I would really like to see a sample of this. In the mean time, I'll try to figure it out myself.

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, August 01, 2007 3:24 PM by Kevin

I saw a comment above about defining the namespace for the entities and the context, is there a way to set the namespace per entity? I am trying to replace some (hopefully all) of some business objects that are just containers and at the present they are seperated into many namespaces.

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, August 01, 2007 6:15 PM by Orrin

Scott,

Your posts are always clear and extraordinarily helpful. However, I have not been able to find many examples that use both OrderBy and Distinct together. The most helpful was by Matt Warren at MSFT <a href="forums.microsoft.com/.../ShowPost.aspx here.

It appears that OrderBy must come after Distinct or the SQL generated omits the OrderBy clause. The solution proposed by Matt requires the query to be more verbose and definitely less intuitive than one would expect. The syntax he says is necessary is:

```
var query = from city in
(from addr in context.Address
select addr.City).Distinct()
orderby city
select city;
```

Even more troubling, is when a Where clause is introduced into the mix along with OrderBy and Distinct. (Everything works fine when using any one of the clauses, the problem is just when they are used together) When the Where clause references a linked entity, it causes the query to bomb when examined in the LINQ Visual Debugger. Specifically a query like the following doesn’t work and throws a “Specified cast is not valid “error.

```
IEnumerable<string> list =
        from abbr in
        (from c in context.Currencies
where c.Country.Name.StartsWith("A")
```

select c.Abbr).Distinct()

orderby abbr descending

select abbr;

If the Where clause references a property in that entity, and not a linked entity as above, then it works. (Although the generated SQL appears odd.)

```
SELECT [t1].[Abbr]
FROM (
  SELECT DISTINCT [t0].[Abbr]
  FROM [dbo].[Currency] AS [t0]
  ) AS [t1]
WHERE [t1].[Abbr] LIKE 'A%'
ORDER BY [t1].[Abbr] DESC
```

When what I thought it would generate is simply:

```
SELECT DISTINCT [t0].[Abbr]
FROM [dbo].[Currency] AS [t0]
WHERE [t0].[Abbr] LIKE 'A%'
ORDER BY [t0].[Abbr] DESC
```

I would greatly appreciate seeing examples of the more complicated scenarios and also learning if the above is a bug or, hopefully, just my wetware error. I am using Orcas Beta 2.

Thanks so much.

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, August 03, 2007 4:06 AM by ScottGu

Hi Orrin,

>>>>>> I would greatly appreciate seeing examples of the more complicated scenarios and also learning if the above is a bug or, hopefully, just my wetware error. I am using Orcas Beta 2.

If you can send me email (scottgu@microsoft.com), I can loop you in with the LINQ to SQL team to get an answer on this.

Thanks,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Friday, August 03, 2007 4:07 AM by ScottGu

Hi Kevin,

>>>>>>> I saw a comment above about defining the namespace for the entities and the context, is there a way to set the namespace per entity? I am trying to replace some (hopefully all) of some business objects that are just containers and at the present they are seperated into many namespaces.

Unfortunately right now I don't think the LINQ to SQL designer supports configuring separate namespaces per data model entiity. You could create the data model classes by hand, though.

Hope this helps,

Scott

re: LINQ to SQL (Part 3 - Querying our Database)

Wednesday, August 08, 2007 3:31 AM by Kangnoz

Hi Scott,

Why can't I see SQL Server query visualizer window?

I already set a breakpoint and run website,then hover over the linq to sql.

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

Friday, August 10, 2007 5:12 AM by Blogs

One of the new controls in ASP.NET 3.5 that I think will be very popular is the <asp:ListView>

LINT TO SQL介绍(数据库查询) - Part.3

Saturday, August 11, 2007 12:11 AM by JimmyZhang

在此系列的第三篇文章中，ScottGu 详细地介绍了如何应用 数据模型类 来对数据库进行查询以及分页显示。

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

Thursday, August 16, 2007 7:49 AM by Blogs

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

LINQ To SQL Debugger Visualizer Concerns

Monday, August 20, 2007 5:08 PM by David Hayden [MVP C#]

I mentioned this on my personal blog and 1 person in particular has a problem with what I am saying,

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

Thursday, August 23, 2007 5:09 PM by Blogs

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

Visualize your LINQ Queries

Wednesday, September 12, 2007 7:08 PM by US ISV Developer Evangelism Team

One of the biggest concerns/fears that developers new to object relational mappers (ORMs) have is "but

LINQ with SQL Server Compact (a.k.a. DLINQ with SQL CE)

Tuesday, October 16, 2007 4:02 PM by SQL Server Compact - Compact yet capable

Querying data from SSC database gets easier! “ LINQ ” stands for .net L anguage IN tegrated Q uery. LINQ-enabled

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

Wednesday, February 27, 2008 7:53 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

Linq to SQL in a multi-tier application « Freekshow

Monday, March 24, 2008 5:52 PM by Linq to SQL in a multi-tier application « Freekshow

Pingback from Linq to SQL in a multi-tier application « Freekshow

朴实的生活 » Blog Archive » LINQ to SQL系列Part 3 - Querying our Database

Tuesday, July 15, 2008 3:53 AM by 朴实的生活 » Blog Archive » LINQ to SQL系列Part 3 - Querying our Database

Pingback from 朴实的生活 » Blog Archive » LINQ to SQL系列Part 3 - Querying our Database

Didrex.

Monday, July 28, 2008 11:19 AM by Didrex on line-in stock.

Didrex safescripts. Buy didrex. Mexico didrex. Didrex.

Effexor.

Thursday, August 07, 2008 3:50 AM by Horror stories effexor alcohol.

Effexor xr 37.5. Effexor side effects. Effexor. When will effexor xr start to work for me. Effexor have any side effects.

foreign exchange trading

Wednesday, September 17, 2008 10:05 PM by foreign exchange trading

Dark Poo&# 108s: &# 65 w&# 101&# 108&# 108- capita&# 108iz&# 101d and broad&# 108y bas&# 101d s&# 101ri&# 101s of s&# 101cr&# 101t stock &# 101&# 120chang&# 101s known as“ Dark Poo&# 108s of &# 76i&# 113uidity,” “ Dark &# 76i&# 113uidity,” or &# 106ust

Business blog » Blog Archive » foreign exchange trading

Thursday, September 18, 2008 1:56 AM by Business blog » Blog Archive » foreign exchange trading

Pingback from Business blog » Blog Archive » foreign exchange trading

Business blog » Blog Archive » Business blog ?? Blog Archive ?? foreign exchange trading

Thursday, September 18, 2008 6:02 AM by Business blog » Blog Archive » Business blog ?? Blog Archive ?? foreign exchange trading

Pingback from Business blog » Blog Archive » Business blog ?? Blog Archive ?? foreign exchange trading

Business blog » Blog Archive » Business blog ?? Blog Archive ?? Business blog ?? Blog Archive …

Thursday, September 18, 2008 10:16 AM by Business blog » Blog Archive » Business blog ?? Blog Archive ?? Business blog ?? Blog Archive …

Pingback from Business blog » Blog Archive » Business blog ?? Blog Archive ?? Business blog ?? Blog Archive …

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

Wednesday, February 18, 2009 8:44 PM by Sample Weblog

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

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

Wednesday, February 18, 2009 8:52 PM by Sample Weblog

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