

Introduction to LINQ

Paul Litwin
Collaborative Data Services (CDS)

plitwin@fhcrc.org

Agenda

- What is LINQ?
- Language Features Behind LINQ
- LINQ Basics
- LINQ to Objects
- LINQ to SQL
- Other LINQ providers

What is LINQ?

- LINQ == Language Integrated Query
- Additions to .NET languages that enable language-based querying
- LINQ Providers
 - LINQ to Objects – in memory-collections
 - LINQ to SQL – SQL Server databases
 - LINQ to XML – XML documents
 - LINQ to DataSets – data in a ADO.NET DataSet
 - LINQ to Entities – works with ADO.NET Entity Framework
 - ...

Language Features Behind LINQ

- Automatic Properties
- Object and Collection Initializers
- Inferred and Anonymous Types
- Extension Methods
- Lambda Expressions

Automatic Properties – C# Only

- Quick way to define property Not really necessary for LINQ but nice

replace this

```
private string _firstName;  
  
public string firstName  
{  
    get { return _firstName; }  
    set { _firstName = value; }  
}
```

with this

```
public string firstName { get; set; }
```

Object Initializers – C#

- Quick way to create an instance and set a bunch of properties

replace this

```
Cust20 cust20 = new Cust20();  
cust20.firstName = "Groucho";  
cust20.lastName = "Marx";  
cust20.birthDate = new DateTime(1890, 10, 2);
```

with this

```
Cust35 cust35 = new Cust35 { firstName = "Jim",  
lastName = "Gaffigan", birthDate = new DateTime(1966, 7, 7) };
```

Object Initializers – VB

- Quick way to create an instance and set a bunch of properties

replace this

```
Dim cust20 As New Cust20()  
cust20.firstName = "Groucho"  
cust20.lastName = "Marx"  
cust20.birthDate = New DateTime(1890, 10, 2)
```

with this

```
Dim cust20 As New Cust20() With {.firstName = "Jim", _  
    .lastName = "Gaffigan", .birthDate = New DateTime(1966, 7, 7)}
```

Collection Initializers – C# Only

replace this

```
List<Cust20> pcTitans20 = new List<Cust20>();  
  
Cust20 steveJobs = new Cust20();  
steveJobs.firstName = "Steve";  
steveJobs.lastName = "Jobs";  
steveJobs.birthDate = new DateTime(1955, 2, 24);  
pcTitans20.Add(steveJobs);  
...
```

with this

```
List<Cust20> pcTitans35 = new List<Cust35>  
{  
    new Cust35{firstName="Steve",lastName="Jobs",  
              birthDate=new DateTime(1955,2,24)},  
    ...  
}
```

Type Inference

- Have compiler infer type from usage

C#

```
var serialNumber1 = 123456; // inferred as int  
var serialNumber2 = "789"; // inferred as string
```

VB

```
Dim serialNumber1 = 123456 ' inferred as int  
Dim serialNumber2 = "789" ' inferred as string
```

Anonymous Types

- Allow you create an instance of an unnamed (i.e., anonymous) class that is defined in line

C#

```
var myHouse = new {color = "green", address = "100 Elm"}
```

VB

```
Dim myHouse = New With {.color = "green", _  
    .address = "100 Elm"}
```

Extension Methods – C#

- Allow you to add a custom method to an existing type

```
public static class ScottGuExtensions
{
    public static bool IsValidEmailAddress (this string s)
    {
        Regex regex = new
            Regex(@"^[a-zA-Z0-9-]+@[a-zA-Z0-9-]{2,4}$");
        return regex.IsMatch(s);
    }
}
```

Extension Methods – VB

- Allow you to add a custom method to an existing type

```
Imports System.Runtime.CompilerServices  
Imports System.Text.RegularExpressions
```

```
Public Module ScottGuExtensions
```

```
<Extension(> _
```

```
Public Function IsValidEmailAddress(ByVal s As String) _  
As Boolean
```

```
Dim regex As Regex = _
```

```
    New Regex("^[\w-\.]+"@([\w-]+\.)+[\w-]{2,4}$")
```

```
Return regex.IsMatch(s)
```

```
End Function
```

```
End Module
```

Basic LINQ Examples

```
Comedians = new List<String> { "Groucho Marx", "Eric Idle", ... }
```

```
var results = from c in Comedians  
              where c.Contains("Marx")  
              select c;
```

```
var results = from c in Comedians  
              orderby c  
              select c;
```

- [LINQ2Objects.aspx](#) – C#
- [LINQ2ObjectsVB.aspx](#) – VB

Query Basics – C#

- Query expression consists of set of clauses written in a declarative syntax similar to SQL or XQuery
- Query must
 - begin with from clause, and
 - end with select or group clause
- Between 1st from clause and last select/group clause, it can contain one or more of following clauses
 - where
 - orderby
 - join
 - let
 - from
 - into

Query Basics – VB

- Query expression consists of set of clauses written in a declarative syntax similar to SQL or XQuery
- Query must
 - begin with From or Aggregate clause
- After From/Aggregate clause, you may include any of following clauses
 - Select
 - Where
 - Order By
 - Join
 - Group By
 - Aggregate
 - Let
 - Distinct
 - Take
 - Take While
 - Skip
 - Skip While
 - Into

Contrasting Query Keywords

C# LINQ	VB LINQ	ANSI SQL
from	From	FROM
select	Select	SELECT
where	Where	WHERE
orderby	Order By	ORDER BY
join	Join	JOIN
group	Group By	GROUP BY
Distinct()	Distinct	DISTINCT
into	Into	INTO
let	Let	AS
Count(), Sum(),...	Aggregate	COUNT, SUM,... w/ no group
Skip() SkipWhile()	Skip Skip While	n/a
Take() TakeWhile()	Take Take While	n/a

LINQ Basics

- There are two ways to write LINQ queries
 - Query syntax
 - The previous examples used this syntax
 - Method syntax
 - This syntax uses Lambda expressions

Anonymous Methods Review

- C# 2.0 introduced anonymous methods

C# 1.0

```
btn.Click += new EventHandler(btn_Click):
```

```
void btn_Click(object sender, EventArgs e)  
{  
    lblResult.txt = "Hello world";  
}
```

C 2.0

```
btn.Click += delegate(object sender, EventArgs e)  
{  
    lblResult.Text = "Hello World";  
}
```

Lambda Expressions

- Lambda expressions take the C# 2.0 anonymous method one step further

C 2.0

```
btn.Click += delegate(object sender, EventArgs e)
{
    lblResult.Text = "Hello!";
}
```

C 3.0

```
btn.Click += (object sender, EventArgs e) => lblResult.Text = "Hello!";
```

-or-

```
btn.Click += (sender, e) => lblResult.Text = "Hello!";
```

Lambda Expressions

- `=>` goes into
- separates parameters from method body
- can drop parentheses when there is only one parameter

LINQ – C#

- Equivalent LINQ queries
 - query syntax

```
var results = from c in Comedians  
               where c.Contains("Marx")  
               select c;
```

- method (lambda expression) syntax

```
var results = Comedians.Where(c => c.Contains("Marx"));
```

LINQ – VB

- Equivalent LINQ queries
 - query syntax

```
Dim results = From c In Comedians _  
                Where (c.Contains("Marx")) _  
                Select c
```

- method (lambda expression) syntax

```
Dim results = Comedians.Where(Function(c) c.Contains("Marx"))
```

LINQ – Hybrid Syntax

- You can also use a hybrid of query and method syntax
 - often this is necessary because using query syntax alone is not expressive enough, especially when using C#

```
var results = (from c in Comedians  
                select c).Count();
```

LINQ to Objects

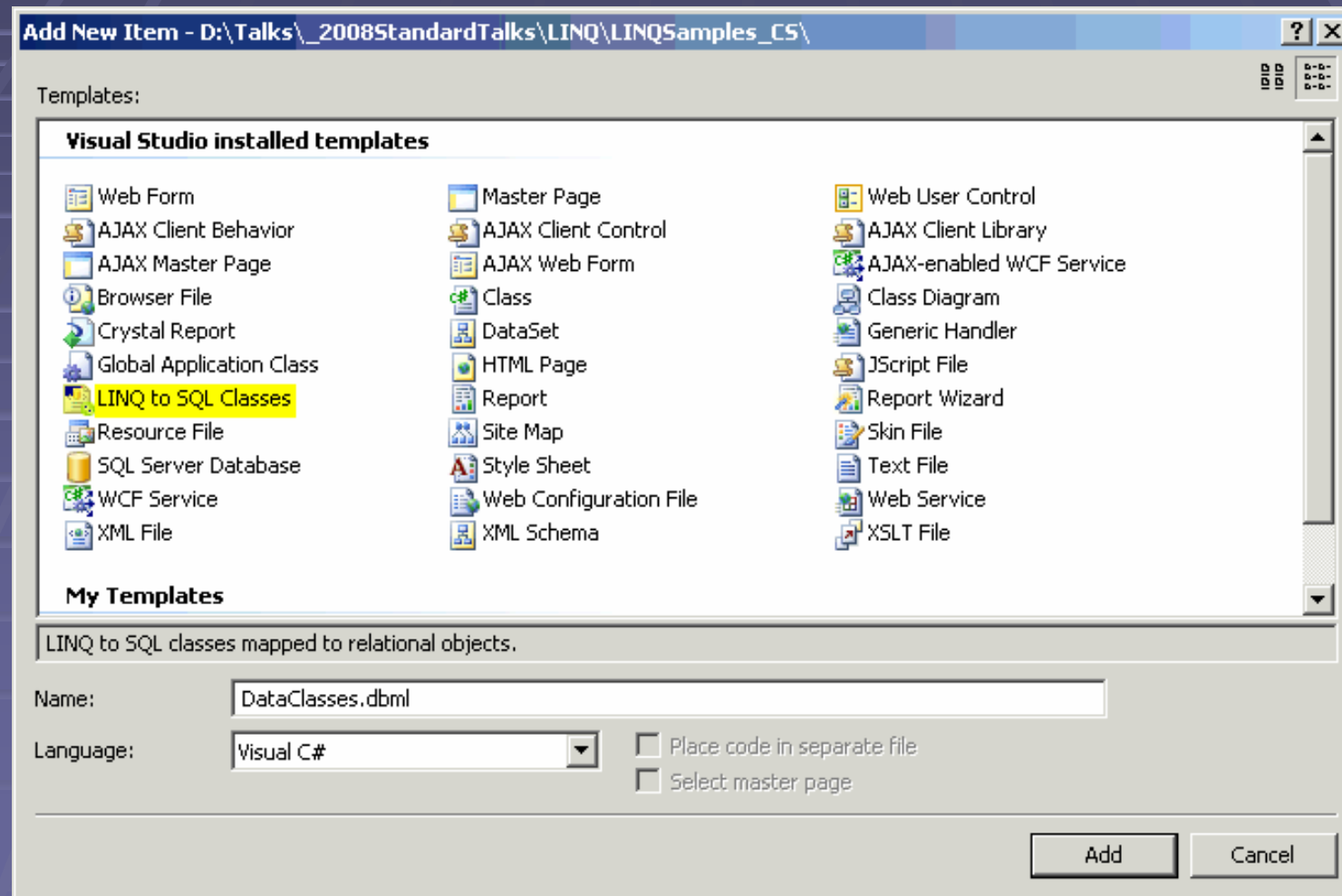
- If an object supports either `IEnumerable` or `IEnumerable<T>` interface, LINQ to Objects provider enables you to query it
- `Array`, `ArrayList`, `collections`, `generic collections`, ...
- `System.Linq` namespace
- All examples shown so far have used LINQ to Objects

LINQ to SQL

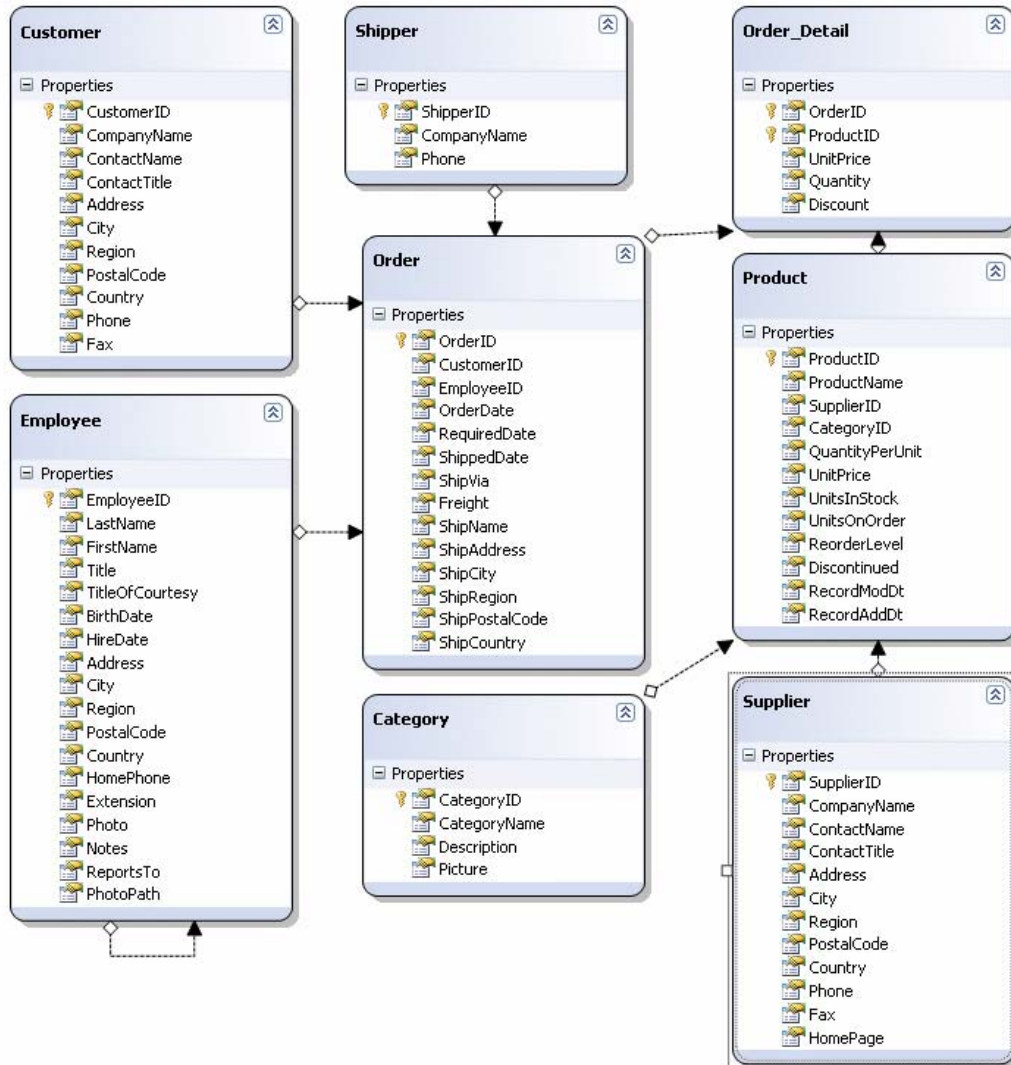
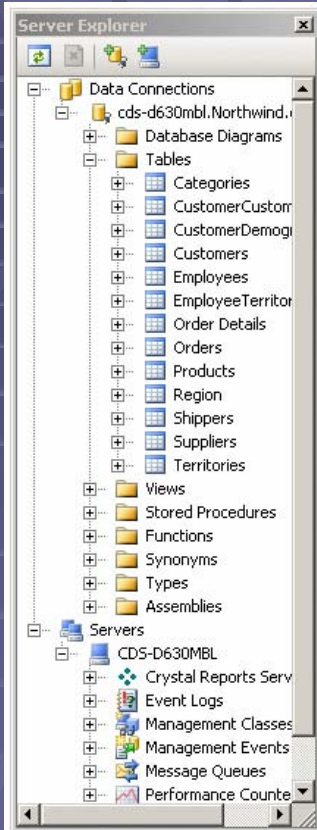
- System.Data.Linq namespace
- Works with SQL Server 2000/2005
- Requires entity classes which you can build using the Object Relational Designer (ORD)
- Entity Framework will let you use LINQ with other databases using E-R data models

Using ORD (1 of 3)

- Add to your App_Code folder



Using ORD (2 of 3)



Using ORD (3 of 3)

- Generated designer.cs file

```
210
211 public Customer()
212 {
213     this._orders = new EntitySet<Order>(new Action<Order>(this.attach_
214         onCreated());
215 }
216
217 [Column(Storage="_CustomerID", DbType="NChar(5) NOT NULL", CanBeNull=false)]
218 public string CustomerID
219 {
220     get
221     {
222         return this._CustomerID;
223     }
224     set
225     {
226         if ((this._CustomerID != value))
227         {
228             this.OnCustomerIDChanging(value);
229             this.SendPropertyChanging();
230             this._CustomerID = value;
231             this.SendPropertyChanged("CustomerID");
232             this.OnCustomerIDChanged();
233         }
234     }
235 }
236
237 [Column(Storage="_CompanyName", DbType="NVarChar(40) NOT NULL", CanBeNull=false)]
238 public string CompanyName
239 {
240     get
241     {
242         return this._CompanyName;
243     }
244     set
245     {
246         if ((this._CompanyName != value))
247         {
248             this.OnCompanyNameChanging(value);
249             this.SendPropertyChanging();
250             this._CompanyName = value;
251             this.SendPropertyChanged("CompanyName");
```

Binding to a LINQ Query

- Once the ORD created, you can reference its DataContext

```
NorthwindDataContext db = new  
NorthwindDataContext();
```

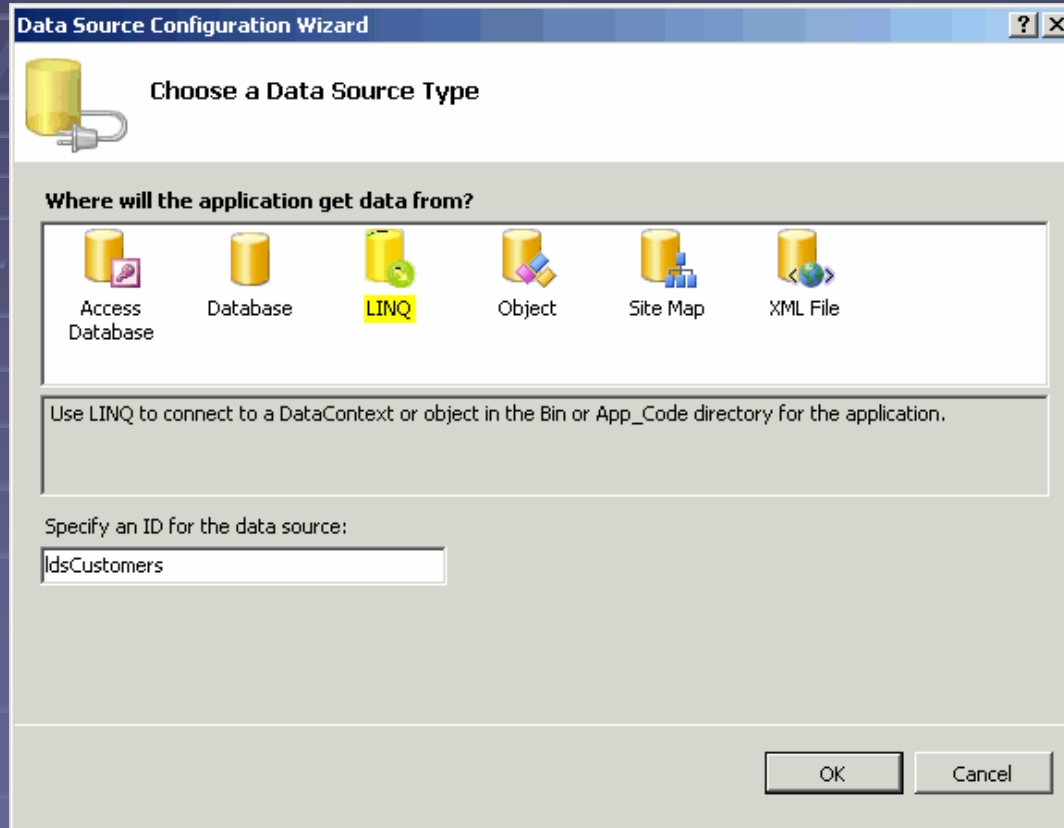
```
var results = from cust in db.Customers  
               where cust.Country == "USA"  
               orderby cust.CompanyName  
               select cust;
```

```
gvCustomers.DataSource = results;  
gvCustomers.DataBind();
```

- Example: CustomersGrid1.aspx

Alternately....

- You can use LINQ data source



- Example: CustomersGrid2.aspx

LINQ Tools

- LINQPad
 - free tool for testing LINQ queries
 - from Joseph Albahari
 - <http://www.linqpad.net/>
- LINQ Debug Visualizer
 - free download from Microsoft
 - <http://www.scottgu.com/blogposts/linqquery/sqlserverqueryvisualizer.zip>

Selecting Fields

- All fields (CustomersGrid1.aspx)

```
var results = from cust in db.Customers  
select cust;
```

- Subset of fields –notice use of "new"
to create anonymous type (CustomersGrid3.aspx)

```
var results = from cust in db.Customers  
select new  
{  
    cust.CompanyName,  
    cust.ContactName,  
    cust.City  
};
```

Filtering Data

```
var results = from cust in db.Customers
               where (cust.Country == "USA" &&
                     (cust.CompanyName.ToLower().StartsWith("t") ||
                      cust.CustomerID.ToLower().StartsWith("t")))
               orderby cust.CompanyName
               select cust;
```

- Filter1.aspx

Using ORD Associations

- ORD automatically adds associations for database relationships

```
var results = from c in db.Customers
               where c.Country == "USA"
               select new
               {
                   c.CompanyName,
                   c.City,
                   TotalOrders = c.Orders.Count,
                   LastOrder = c.Orders.Max(o => o.OrderDate)
               };
```

- Association1.aspx

Associations

- You can always get to 1st order associations
- Getting to 2nd and 3rd order associations requires you to compose your query so that you traverse 1—m relationships from the many side to the one side
- Contrast
 - Association1.aspx, and
 - Association2.aspx

Ordering Data

```
// ascending is default order
var results = from cust in db.Customers
              orderby cust.Country,
                      cust.Orders.Count descending
              select new
              {
                cust.Country,
                cust.City,
                cust.CompanyName,
                Orders = cust.Orders.Count
              };
```

- Order1.aspx

Joining Tables

```
var results = from order in db.Orders
              join detail in db.Order_Details
                on order.OrderID equals detail.OrderID
              join product in db.Products
                on detail.ProductID equals product.ProductID
              join customer in db.Customers
                on order.CustomerID equals customer.CustomerID
              select new
              {
                customer.CompanyName,
                order.OrderDate,
                product.ProductName,
                detail.UnitPrice,
                Cost = detail.UnitPrice * detail.Quantity
              };
```

- [Join1.aspx](#)

Grouping Data

```
from d in db.Order_Details
group new
{
    order = d.Order,
    detail = d,
    customer = d.Order.Customer
} by d.Order.Customer.CompanyName into grpQuery
select new
{
    Company = grpQuery.Key,
    Purchases = grpQuery.Count(),
    FirstOrder = grpQuery.Min(g => g.order.OrderDate),
    LastOrder = grpQuery.Max(g => g.order.OrderDate),
    TotalCost = grpQuery.Sum(g => g.detail.Quantity * g.detail.UnitPrice)
};
```

- Group1.aspx

Calling Stored Procedure

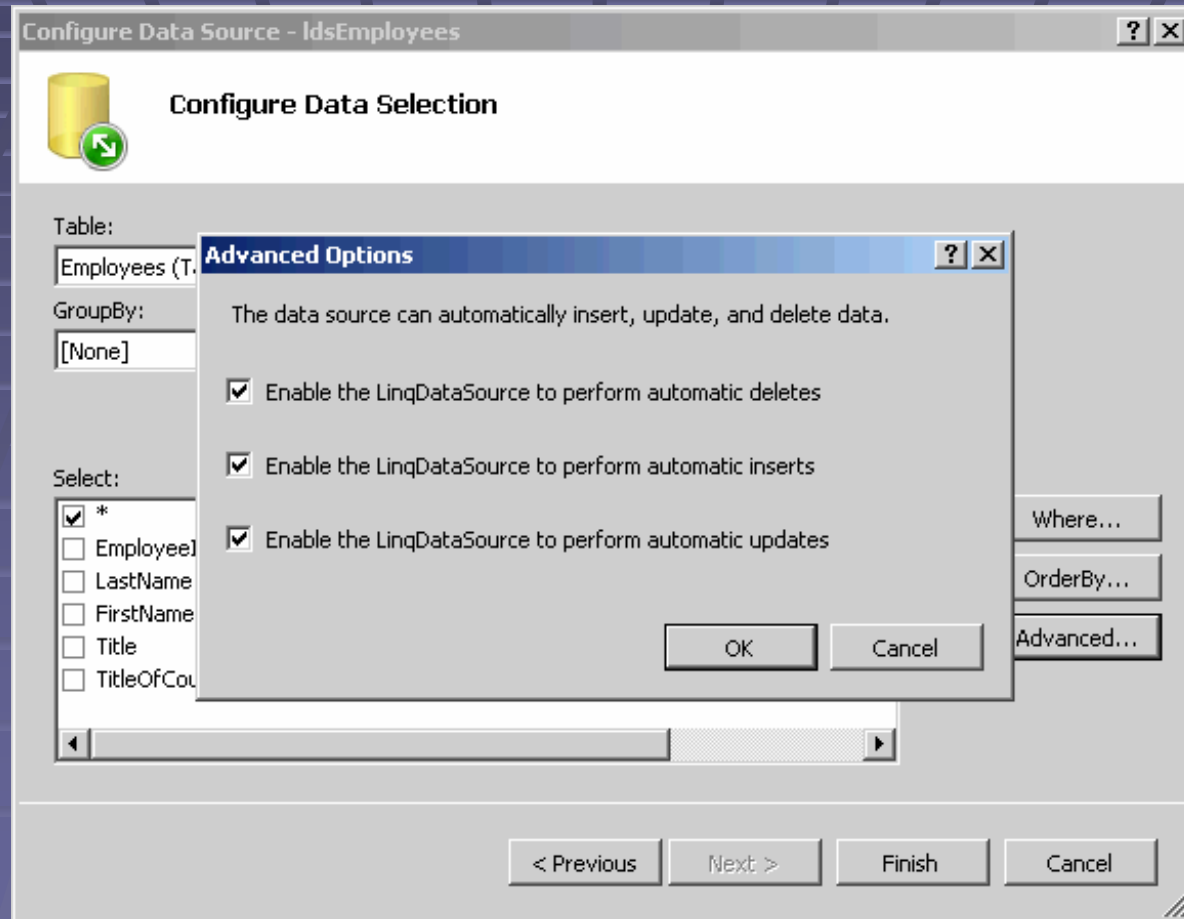
- Drag sproc from Server Explorer to data model
- Call sproc as method of data context
- Example: CustomersFromSproc.aspx

```
var customersByCountry =  
    db.procCustomerSelectByCountry(ddlCountry.SelectedValue);  
  
gv.DataSource = customersByCountry;  
gv.DataBind();
```

Updating Data

- Can update, of course, using sprocs
- You can also use LinqDataSource control
 - Example: UpdateCategories1.aspx
- You can also use an ObjectDataSource control against a Data Access Layer (DAL)
 - Example: UpdateCategories2.aspx
 - see excerpt in later slide
 - This example also illustrates use of LINQ for SQL Server data paging

Using LinqDataSource Control



Updating Data Example

```
public void Update(CategoryDAL cat2Update)
{
    var cat = (from c in db.Categories
               where (c.CategoryID == cat2Update.CategoryId)
               select c).Single();

    cat.CategoryName = cat2Update.CategoryName;
    cat.Description = cat2Update.Description;

    db.SubmitChanges();
}
```

SQL Server Data Paging

```
public List<CategoryDAL> Select(int startRowIndex, int maximumRows)
{
    var cat = (from category in db.Categories
               orderby category.CategoryID
               select new
               {
                   category.CategoryID,
                   category.CategoryName,
                   category.Description
               }).Skip(startRowIndex).Take(maximumRows);

    List<CategoryDAL> catList = new List<CategoryDAL>();
    foreach (var c in cat)
    {
        CategoryDAL cd = new CategoryDAL();
        cd.CategoryId = c.CategoryID;
        cd.CategoryName = c.CategoryName;
        cd.Description = c.Description;
        catList.Add(cd);
    }

    return catList;
}
```

Other LINQ Providers

- LINQ to XML
- LINQ to DataSets (aka LINQ to ADO.NET)
- LINQ to Entities

LINQ to DataSets (aka LINQ to ADO.NET)

- Grab data from ADO.NET and put it into a DataSet, DataTable
- Use LINQ to shape the data

LINQ to DataSet Example

- Example: LINQ2Dataset.aspx

```
cnx = new SqlConnection(strCnx);
sda = new SqlDataAdapter("procCustomersSelect", cnx);
sda.SelectCommand.CommandType = CommandType.StoredProcedure;

dt = new DataTable();

sda.Fill(dt);

var cust = from c in dt.AsEnumerable()
           orderby c.Field<string>("ContactName")
           select new
           {
               Contact = c.Field<string>("ContactName"),
               Company = c.Field<string>("CompanyName"),
               Location = c.Field<string>("City") + ", " +
                       c.Field<string>("Country")
           };
```

Other LINQ Providers

- LINQ to Entities
 - Takes O/RM much further than LINQ to SQL
 - Part of Visual Studio SP 1

Discussion Points

- When are LINQ's strengths?
- What are its weaknesses?
- Is LINQ the future of data access in .NET?
- Or is the more "classic" N-tier, ADO.NET, stored procedures approach make more sense in many scenarios?

LINQ Resources

- Slides/Samples for talk
 - <http://cds.fhcrc.org/downloads.aspx>
- LINQPad
 - <http://www.linqpad.net/>
- LINQ Debug Visualizer
 - <http://www.scottgu.com/blogposts/linquery/sqlserverqueryvisualizer.zip>
- Scott Guthrie's Blog
 - <http://weblogs.asp.net/scottgu/>