بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش معرفی امکانات و قابلیت های LINQ برای مدیران پروژه ها

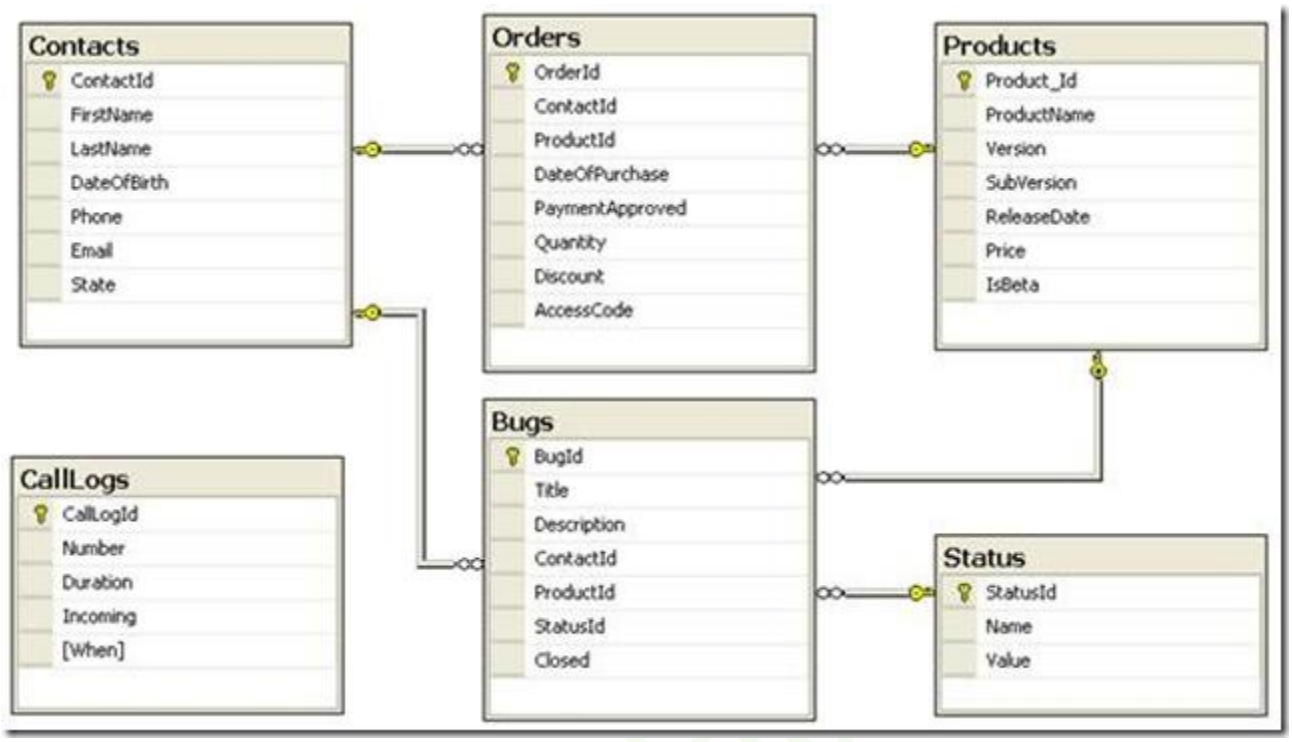
مدرس : مهندس افشین رفوآ

رمز فایل :tahlildadeh.com

LINQ to SQLبه برنامه نویسان .NET اجازه نوشتنquery ها را درزبان .NET. می دهد تا بتوانند داده ها را از بانک اطلاعاتی SQL Serverبازیابی و تغییر دهند. به طور عام، LINQ to SQLاجازه ایجادquery های SQL در syntax زبان .NET.انتخابی مان و کار کردن با مجموعه ای قوی از اشیا به عنوان نتیجه برگشتی را به ما می دهد. می توان تغییراتی را در این اشیا بوجود آورد و سپس دوباره آنها را در database ذخیره کرد.

برای درک مفهوم syntax درLINQ to SQL ، ما ازschema ی بانک اطلاعاتی SQL زیر استفاده می کنیم که نرم افزاری ساده برای ثبت محصولات و helpdesk است؛ و با داده های نمونه populate شده و دارای روابط کلید خارجی (foreign-key relationship) است که در جای مناسب تعریف می شود.

آدرس آموزشگاه : تهران ـ خیابان شریعتی ـ بالا تر از خیابان ملک ـ جنب بانک صادرات ـ پلاک ۵۶۱ ـ واحد ۷
88146323 - 88446780 - 88146330



SQL Database Schema که برای مثالهای LINQ to SQL استفاده می شود:

از شما می خواهم چند دقیقه این واقعیت را فراموش کنید که ما برضد نوع داده ای HookedOnLINQ کد نوبیس می کنیم، بعداً توضیح خواهم داد که چگونه آن را در چند صفحه ایجاد کردم، فعلاً مد نظر داشته باشید که این، یک ساختار شی است که از این database schema تقلید می کند.

```
HookedOnLINQ db = new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");

   var q = from c in db.Contact
           where c.DateOfBirth.AddYears(35) > DateTime.Now
           orderby c.DateOfBirth descending
           select c;

   foreach(var c in q)
       Console.WriteLine("{0} {1} b.{2}",
                  c.FirstName.Trim(),
                  c.LastName.Trim(),c.DateOfBirth.ToString("dd-MMM-yyyy"));
```

Output:

Mack Kamph b.17-Sep-1977

Armando Valdes b.09-Dec-1973

عبارت LINQ to SQL Query در بانک اطلاعاتی SQL Server – مخاطبان کمتر از ۳۵ سال سن، ابتدا جوانترین .

هنگامی که حلقه foreach را وارد می کنیم، عبارت SQL زیر توسط LINQ فرموله می شود و روی سرور اجرا می شود. (دانستن این نکته مهم است که SQL فقط اولین باری که ما داده ها را request می کنیم اجرا می شود، تا آن موقع، query به صورت یک عبارت در حافظه نگهداری می شود این فرآیند، Deferred Execution نامیده می شود.)

```
SELECT [t0].[ContactId], [t0].[FirstName], [t0].[LastName], [t0].[DateOfBirth],
[t0].[Phone], [t0].[Email], [t0].[State]
FROM [Contact] AS [t0]
WHERE DATEADD(YEAR, @p0, [t0].[DateOfBirth]) > @p1
ORDER BY [t0].[DateOfBirth] DESC
```

عبارات SQL که توسط LINQ ایجاد شده و مخاطبین بیشتر از تاریخی معین شده را که به عنوان یک پارامتر ارسال شده، باز می گرداند .

عبارت query در C# ، به کد SQL پارامتریزه شده تبدیل شد، پارامترها ایجاد شدند و query روی سرور اجرا شد LINQ to SQL . به برنامه نویسان اجازه می دهد به استفاده از stored procedure ها به جای SQL ادامه دهند، گرچه حالا مجبورید خودتان کد stored procedure را بنویسید، و در نتیجه بعضی از قابلیت های LINQ را از دست می دهید. بعداً در مورد این موضوع بیشتر بحث می کنیم، فعلاً در نظر داشته باشید که LINQ to SQL از stored procedure و همچنین از فراخوانی های SQL که به طور دینامیکی ایجاد شده اند، در هر شرایطی ساپورت می کند.

اگر بانک اطلاعاتی شما دارای روابط کلید خارجی است، آنگاه سلسله مراتب آنها در مدلهای شی ایجاد شده منعکس می شود. می توان از طریق تعیین کردن جدول child به داده های رکوردهای مربوط دسترسی پیدا کنید. مثال بعدی نشان می دهد چگونه می توان در زنجیره روابط کلید خارج، بدون یک عبارت Join به طور مستقیم navigate کرد.

```
HookedOnLINQ db =
     new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");

var q = from o in db.Orders
        where o.Products.ProductName.StartsWith("Asset") &&
              o.PaymentApproved == true
        select new { name    = o.Contacts.FirstName + " " +
                               o.Contacts.LastName,
                     product = o.Products.ProductName,
                     version = o.Products.Version +
                               (o.Products.SubVersion * 0.1)
                   };

foreach(var x in q)
    Console.WriteLine("{0} - {1} v{2}",
                      x.name, x.product, x.version);
```

Output:

Barney Gottshall - Asset Blaster v1

Barney Gottshall - Asset Blaster v1.1

Armando Valdes - Asset Blaster Pro v1

Jeffery Deane - Asset Blaster Pro v1.1

Stewart Kagel - Asset Blaster Pro v1.1

Blaine Reifsteck - Asset Blaster Pro v1.1

Ariel Hazelgrove - Asset Blaster v1.1

دسترسی به روابط کلید خارجی ساده است. نیازی به join syntax نیست، مستقیماً به زیر اعضا(sub-members) دسترسی دارید.

این مدل شی سلسله مراتبی(hierarchical) ، برای آپدیت ها نیز جواب می دهد. می توانید رکوردها را بوسیله تغییر داده اشیا و اضافه یا حذف کردن اشیا از جداول، در جداول مربوطه assign ، اضافه یا حذف کنید. در پشت پرده، LINQ to SQL، فرمان SQL query زیر را ایجاد و آنرا اجرا می کند. از این نتایج برای populate کردن

مجموعه شی نتیجه (result object collection) استفاده کرد که مجموعه ای از یک type ناشناس (Anonymous)است.

```
SELECT ([t2].[FirstName] + @p2) + [t2].[LastName] AS [value],
[t1].[ProductName], [t1].[Version] + ([t1].[SubVersion] * @p3) AS [value2]
FROM [Orders] AS [t0], [Products] AS [t1], [Contacts] AS [t2]
WHERE ([t2].[ContactId] = [t0].[ContactId]) AND
([t1].[ProductName] LIKE @p0) AND ([t0].[PaymentApproved] = @p1)
AND ([t1].[Product_Id] = [t0].[ProductId])
```

کد SQLکه نشان می دهد چگونه joins to related table through foreign-keys، اضافه شدند .

اگر بانک اطلاعاتی شما دارای روابط کلید خارجی نیست که بین دو جدول تعریف می شوند، LINQ to SQL، دسترسی نسبی (relational access) را توسط تعیینJoin ها در عبارات query ، قبول می کند Query. زیر چگونگی join کردن را در جاییکه یک کلید خارجی بین دو جدول Contacts.Phone و CallLogs.Number تعریف نشده، نشان می دهد.

```
HookedOnLINQ db =
    new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");

var q = from call in db.CallLogs
        join contact in db.Contacts on call.Number equals contact.Phone
        select new {contact.FirstName, contact.LastName,
                    call.When, call.Duration};

foreach(var call in q)
    Console.WriteLine("{0} - {1} {2} ({3}min)",
        call.When.ToString("ddMMM HH:mm"),
        call.FirstName.Trim(), call.LastName.Trim(), call.Duration);
```

اگر هیچ کلید خارجی وجود نداشته باشد، می توانید از اپراتور Joinدر عبارت queryاستفاده کنید.

جهت تغییر دادن و اضافه کردن رکورد به بانک اطلاعاتی مان، فقط باید تغییراتی رابه اشیای موجود در حافظه اعمال کنید و سپس متد SubmitChanges را فراخوانی کنید (مواظب باشید، من یک بار به اشتباه متد AcceptChangesرا فرا خواندم که تغییرات را قبول می کند و همه رکوردها را به صورت اریجینال mark می کند اما در database ذخیره نمی کندLINQ to SQL .، رد تغییرات را نگه می دارد و عبارات SQL را ایجاد می کند تا همه آپدیت ها، insertها، وdelete ها را تحت تاثیر قرار دهد. می توانید این رفتار پیش فرض را Overrideکنید و متدهای پیاده سازی خودتان را تعیین کنید و به جای آن استفاده کنیدLINQ to SQL .، یک تراکنش را در اطراف آپدیت های database ایجاد می کند، پس اگر قسمتی دچار اشکال شود، فرصت دارید تا errorرا capture کنید، آنرا اصلاح و دوباره تلاش کنید. همچنین می توانید کنترل کنید LINQ to SQL چگونه errorهای همزمان را مدیریت کند (وقتی شخص دیگری داده هایی را که قبلاً ویرایش می کردید تغییر می دهد، شما شانس ذخیره کردن را دارید.)

```
HookedOnLINQ db =
      new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");
```

```
// Change - Get an object, make the change in memory, Call SubmitChanges
Contacts q = (from c in db.Contacts
              where c.FirstName == "Armando" && c.LastName == "Valdes"
              select c).FirstOrDefault();

if (q != null) {
       q.Email = "Armando.Valdes@aspiring-technology.com";
}

try {
    db.SubmitChanges();
}
catch (OptimisticConcurrencyException e) {
    // You have your choice of RefreshMode to resolve concurrency conflicts.
    // You can KeepChanges, KeepCurrentValues, OverwriteCurrentValues.
    e.Resolve(RefreshMode.OverwriteCurrentValues);
    db.SubmitChanges();
}
```

آپدیتی که چگونگی مدیریت کردن error های همزمان را نشان می دهد. شما تغییرات را در اشیا ایجاد می کند و سپس SubmitChanges را فرا می خوانید .

Insert کردن رکوردهای جدید به سادگی ایجاد نمونه جدیدی از اشیا و اضافه کردن آن به مجموعه ی مناسب و سپس فراخوانی SubmitChanges است.

```
HookedOnLINQ db =
    new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");

// Adding Records - (1) Create a new object and sub-objects,
// (2) Add it to the DataContext collection, (3) Call SubmitChanges

// (1)
Contacts newContact    = new Contacts();
newContact.FirstName   = "Troy";
newContact.LastName    = "Magennis";
newContact.Phone       = "425 749 0494";
newContact.Email       = "troy@aspiring-technology.com";
newContact.DateOfBirth = new DateTime(1980, 08, 07);
// Create sub-record and add to this contact
Orders newOrder         = new Orders();
newOrder.Products       = (from p in db.Products
                           where p.ProductName == "Asset Blaster Pro"
                           select p).FirstOrDefault();
newOrder.DateOfPurchase = DateTime.Now;

// (2)
newContact.Orders.Add(newOrder);
db.Contacts.Add(newContact);

// (3)
```

```
db.SubmitChanges();
```

insertکردن یک رکورد جدید و یک زیر رکورد مرتبط. فقط اشیا را ایجاد کنید و به یک collection اضافه کنید .

LINQ to SQLدرSubmitChanges ، عبارات SQL را به ترتیب صحیحی generate می کند تا رکوردهای جدید را در database ذخیره کند و هر کدام را به طور صحیح reference کند. در این مثال، LINQ to SQL، به insert کردن Contact جدید نیاز دارد تا ابتدا کلید اصلی را بدست بیاورد و سپس آنرا هنگام نوشتن ترتیب جدید در database استفاده کند. کل فرآیند دریک transaction انجام می شود، پس اگر هر مرحله ای دچار اشکال شود، آنگاه کل database به حالتی که قبل از اینکه SubmitChanges فراخوانده شود، بر میگردد.

```
Start LOCAL Transaction (ReadCommitted)

INSERT INTO [Contacts](FirstName, LastName, DateOfBirth, Phone, Email, State)
 VALUES(@p0, @p1, @p2, @p3, @p4, @p5)
SELECT [t0].[ContactId]
FROM [Contacts] AS [t0]
WHERE [t0].[ContactId] = (CONVERT(Int,@ @IDENTITY))

INSERT INTO [Orders](ContactId, ProductId, DateOfPurchase, PaymentApproved, Quantity,
Discount, AccessCode)
VALUES(@p0, @p1, @p2, @p3, @p4, @p5, @p6)
SELECT [t0].[OrderId]
FROM [Orders] AS [t0]
WHERE [t0].[OrderId] = (CONVERT(Int,@ @IDENTITY))

Commit LOCAL Transaction
```

SQLهنگام نوشتن یک رکورد و زیر رکورد، اجرا می شود. به wrapکردن کل فرآیند توسط transactionدقت کنید .

این رکوردها بعد از اینکه متد SubmitChanges در مثال شکل ۱۶ فراخوانی شود، اضافه می شوند.

جدول contactها:

ContactId	FirstName	LastName	DateOfBirth	Phone	Email
13	Troy	Magennis	1980-08-07	425 749 0494	troy@aspiring-technology.com

جدول سفارش ها:

OrderId	ContactId	ProductId	DateOfPurchase
12	13	3	2006-11-30 18:50:24.187

چیزی به جدول محصولات اضافه نشد، ولی از یک reference به ProductId of Asset Blaster Pro در رکورد جدول سفارشها استفاده شد. تمامی این جستجوها بر ای کلید های اصلی به طور اتوماتیک مدیریت می شوند.

ProductId	ProductName	Version	SubVersion	Released
3	Asset Blaster Pro	1	0	2006-01-03

حذف کردن رکوردها بسیار ساده است. می توانید یک شی را از مجموعه اشیایی که در حال حاضر در حافظه هستند و از query قبلی جمع آوری شده اند، حذف کنید.

```
// Delete the record(s) we just created (do sub-items first)
db.Orders.Remove(newOrder);
db.Contacts.Remove(newContact);
db.SubmitChanges();
```

مثال هایی ازحذف کردن رکوردها از database

تا اینجا من یک مرحله مهم را حذف کرده ام. ماqueryهایی را ضد یک type بنام HookedOnLINQ نوشته ایم که با یکdatabase connection string ، وinstance types Contacts ، وOrders ، و Products شروع شده است. اینtype ، از لنگرLINQ to SQL ، کلاسی به نامDataContext ، ارث می برد. این کلاس، عباراتmarshalling query را به عبارات SQL ، و همچنین change tracking را در فراخوانی SubmitChangesمدیریت می کند. به علاوه، ما به داشتنtypeهایی برای نمایش جداول داده هایمان و جنبه های mapping اشیا و روابط با معادل هایSQL شان و بالعکس، نیاز داریم. گرچه همه این کلاسها را می توان به طور دستی ایجاد کرد، اما این کار اصلاً توصیه نمی شود. یک ساپورت زمان طراحی و درونی در Visual Studio

به همراه یک ابزار خط فرمان (command line tool) وجود دارد که کل کارهای سنگین در code generation به جای ما انجام می دهد.

کلاس DataContext سفارشی:

- از System.Data.DLINQ.DataContext type ارث می برد
- مجموعه ای از instance type را آغاز می کند (Table<[type]>) و آنها را قابل دسترس می کند. (مثلاً، می توانیم db.Contacts را از میان عبارات query فرابخوانیم(

کلاس های instance object سفارشی:

- با یک صفت [Table] تغییر شکل می دهد
- حاویfield های رایج یاproperty هایی است که با صفات [Column] تغییر شکل داده اند
- روابط کلید خارجی با صفت [Association] را تعریف می کنند
- رفتارUpdate ، Insert، و Delete را با تعریف متدهایی که با صفات[Update] ، [Insert]، و [Delete]علامتگذاری (mark) شده اند را override می کند
- Store Procedure، View، وFunction wrapper ها را با متدهایی که با صفات [StoredProcedure]، [View]، یا [Function] علامتگذاری (mark) شده اند را تعریف می کند
- تضمین می کند که رویدادهای PropertyChanging وPropertyChanged ، هروقت که value تغییر می کند، روی می دهند.

گزینه هایی برای generate کردن کلاس های wrapper و مشتقDataContext ، که عملکرد LINQ to SQL را روی جداول و اشیایdatabase های دیگر قبول می کنند، در زیر آورده شده:

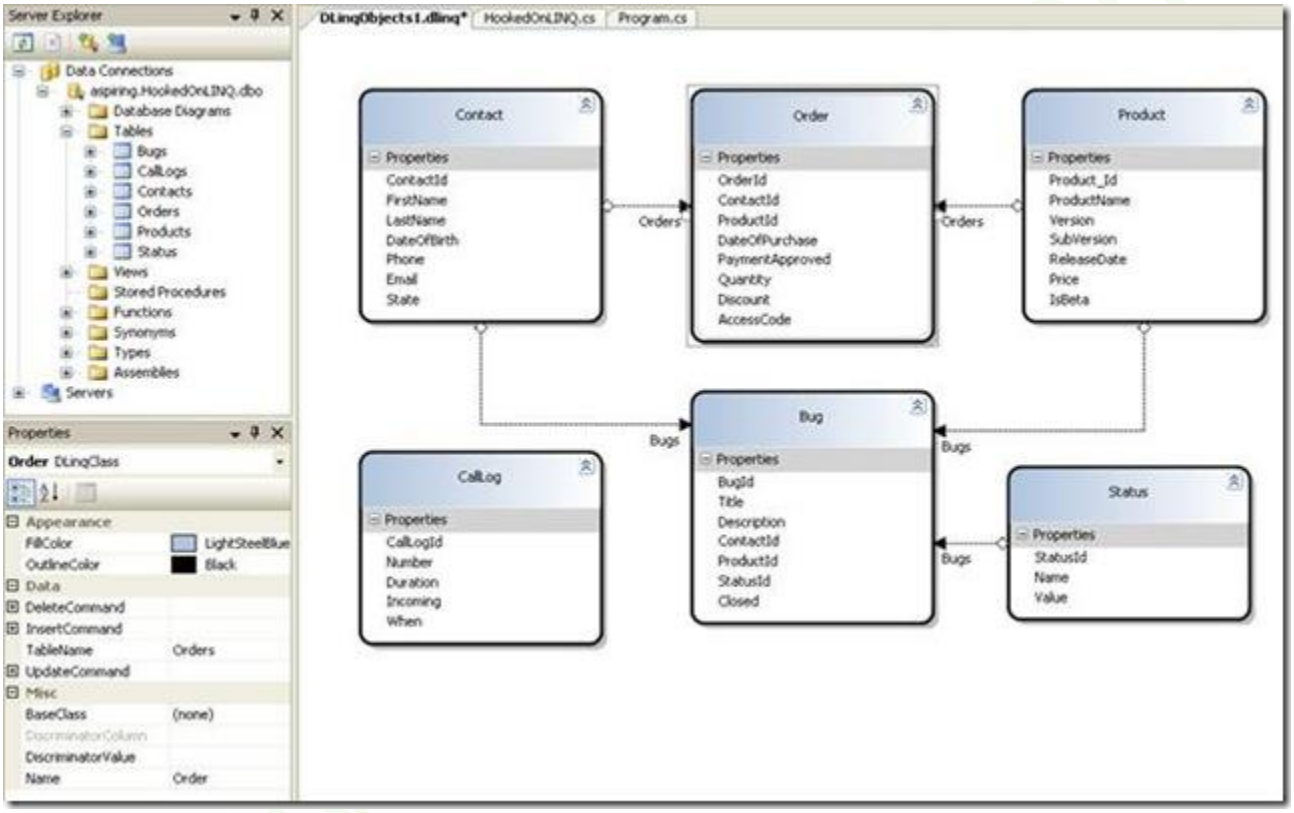
- کل کار را به طور دستی انجام دهیم؛
- از built-in designer برای Visual Studio 2005 استفاده کنیم؛
- از ابزار خط فرمان SQLMetal استفاده کنیم؛
- از یک فایل XML mapping برای لینک کردن جداول و ستونهای database بهtype ها وproperty ها استفاده کنیم. این کار به database و تغییراتmapping ، اجازه می دهد در recompile کردن یک برنامه روی دهند.

برایgenerate کردن object wrapper برای بانک اطلاعاتی نمونه، بنامHookedOnLINQ ، باید برنامه SqlMetalرا با استفاده از ابزار خط فرمان و باargument های زیر اجرا کنید:

```
sqlmetal /server:(local) /database:HookedOnLINQ /code:HookedOnLINQ.cs
```

این کار، یک HookedOnLINQ.cs ایجاد می کند که برای تمامی مثال هایی که تا الان آورده شده اند، کاملاً کاربردی است. من فقط آن را درون پروژه اصلی کپی کردم و solution را compile کردم.

built-in designerبه شما اجازه ایجاد یک DLINQ Object surface را می دهد. می توانید tableها را از پنجرهServer Explorer ، روی آن surface درگ کنید.روابط کلید خارجی به طور اتوماتیک به instanceاضافه می شوند، اگر در بانک اطلاعاتی تعریف شوند، یا می توانید به طور دستی آنها را از surface Toolbox اضافه کنید. هنگام compile کردن، DataContextو instance type ها برای شما ایجاد می شوند. در زیر یک DLINQ Object surfaceآمرده شده کهschema ی HookedOnLINQ را از بانک اطلاعاتی نشان می دهد.



LINQ to SQL Designer Surface.درگ کردن جدول ها از server explorerباعث ایجاد object modelمی شود و به طور اتوماتیک روابط را تعریف می کند.

متد جایگزین استفاده از صفات که مدل رابطه ای را به مدل شی ای لینک می کند، منتقل کردنmapping ها به یک فایل XML است. ابزار خط فرمانSQLMetal ، این فایل XML را برای شما ایجاد می کند، اما می توانید generateکردن آنرا هرطور که دوست دارید اتوماتیک کنید. وقتی که DataContext را ایجاد می کنید، می توانید mapping XML را ارسال کنید، و این کار دقیقاً همان تاثیر استفاده از صفات را خواهد داشت، غیر از اینکه، هنگام compile کردن، به برنامه شما hardcode نمی شود.

خیلی ها معتقدند که database access همیشه باید از طریق Stored Procedure اجرا شوند تا امنیت را ارتقا دهند LINQ to SQL .به طور کاملStored Procedure ها را برای فراخوانی های عمومی و عملیات های update، insert، وdelete ، ساپورت می کند؛ و در موارد زیادی، تجربه شما را با خلاص کردن شما از ایجاد پارامترهای ورودی توسط دست، افزایش می دهد. اما، استفاده محض ازStored Procedure ها، مزایای نوشتن عبارت های Query را در زبان برنامه نویسی اصلی برنامه نویس، از بین می برد. می توان از Stored Procedureها برای همه عملیات هایInsert ، Update، و Delete بکار برد و از عبارات Query برای بازیابی داده ها استفاده کرد. این کار باعث محافظت database در برابر از بین رفتن داده ها می شود، و به برنامه نویسان اجازه می دهد عبارات Query را در VB یا C# بسازند.

فراخوانیstored procedure ها بسیار راحت شده. هنگام استفاده ازADO.NET ، مجبور بودید قبل از ایجاد یک connection بهdatabase ، و فراخوانی واقعیstored procedure ها، پارامترهایی را به طور دستی بسازید. ابزار generate کردن، به عنوان بخشی ازLINQ to SQL ، تابع های wrapper را برای stored procedureها ایجاد می کند.

کد stored procedure زیر، لیستی از پرداختهای سررسیده را بازیابی می کند. روزهایی که از موعد پرداخت گذشته، به عنوان یک پارامتر ارسال می شود. نتیجه این کار، یک cursor با تعدادی ستون است، که یقیناً آن typeنیست که قبلاً در اشیاء C# تعریف کرده ایم.

```
ALTER PROCEDURE [dbo].[GetOverdueAccounts]
    @daysOverdue int = 15
AS
BEGIN
    SET NOCOUNT ON;

    SELECT    o.OrderId, o.Quantity, o.DateOfPurchase, o.Discount,
            c.FirstName + ' ' + c.LastName AS CustomerName,
            c.Phone, c.Email,
            p.ProductName, p.Price,
            ((p.Price*o.Quantity)*((100-o.Discount)/100)) AS Cost,
            DATEDIFF(day, o.DateOfPurchase, GETDATE()) AS OverdueDays
    FROM        Orders o,
            Contacts c,
            Products p
    WHERE        o.ContactId = c.ContactId
    AND        o.ProductId = p.Product_Id
    AND         o.PaymentApproved = 0
    AND        p.IsBeta = 0
    AND        DATEADD(day, @daysOverdue, o.DateOfPurchase) < GETDATE()

END
```

ابزار generate کردن کد، دارای یک switch است که wrapper و result type را برایstored procedure ها، generateمی کند.

```
sqlmetal /server:(local) /database:HookedOnLINQ /sprocs /code:HookedOnLINQ.cs
```

```
HookedOnLINQ db =
    new HookedOnLINQ("Data Source=(local);Initial Catalog=HookedOnLINQ");

var overdue = db.GetOverdueAccounts(30);

foreach (GetOverdueAccountsResult c in overdue)
    Console.WriteLine("{0} days - {1:c}: {2}",
        c.OverdueDays, c.Cost, c.CustomerName);
```

Output:

215 days - $300.00: Armando Valdes

30 days - $180.00: Adam Gauwain

30 days - $247.50: Adam Gauwain