# Linq To Sql Part.1

## 介 绍

ScottGu

译者：张子阳

jimmyzhang.cnblogs.com

jimmy_dev@163.com

出处：Linq To Sql(Part.1 - Introduction)

# 术语表：

Extension：扩展
Initializer：初始器
Expression：表达式
Features：特性
Querying Programming Model：查询编程模型
.Net Language Integrated Query：.Net 集成语言查询
Type-safty：类型安全
Compile-time：编译阶段
Extensibility Model：可扩展模型
Domain-Specific：特定域名
Object Relational Mapping：对象关系映射
Designer：设计器
Representation：表现
Entity class：实体类
Association/Relationship：联系/关系
Solution：解决方案
Conduit：管道

过去的几个月中，我写了一系列的博客随笔，讲述了即将到来的 Visual Studio 和 .Net Framework “Orcas” 版本。下面是这些随笔的链接。

- 自动化 属性、对象、集合 初始器
- 扩展方法
- Lambada 表达式
- 查询语法
- 匿名类型

上面这些语言特性有助于建立一种 查询数据 的使用类的编程概念。我们将这种查询编程模型总称为“LINQ” - 代表的意思是：.Net 集成查询语言。

开发者可以在任何数据源中使用 LINQ。他们可以在他们的编程寓言中体验一种高效的查询，可以有选择的 转换/拼装 数据查询结果到任何他们想要的格式，并且，可以很容易的操作这些查询结果。支持 LINQ 的语言能够提供对类型安全和在编译阶段检测查询表达式的全面支持，并且，在编写 LINQ 代码的时候，开发工具可以提供完全的智能提示(NOTE：就是你输入一个 i，int 直接就出来了，你只需要拍一下空格就可以了。)、调试和 Rich refactoring 支持。

LINQ 支持一种非常丰富的可扩展模型，这种模型使得为数据源建立高效的特定域名的操作变得非常便利。

.Net Framework 的“Orcas”版本(NOTE：现在已经更名为.Net Framework 3.5 和 VS2008)

提供了内置的库，以使 LINQ 支持对象、XML 和各种数据库。

# LINQ TO SQL 是什么？

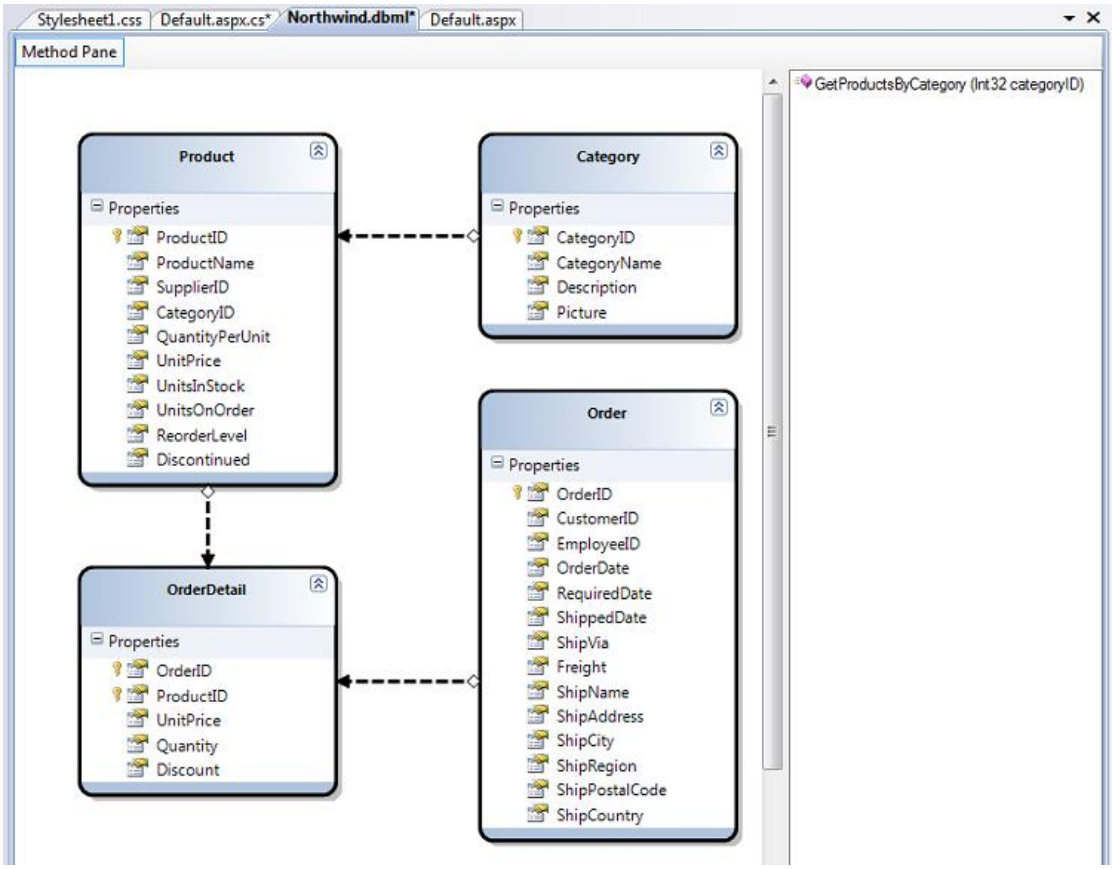
LINQ TO SQL 是包含在.NET Framework 的“Orcas”版中中的一种 O/RM（对象关系映射），O/RM 允许你使用 .NET 的类来对关系数据库进行建模。然后，你可以使用 LINQ 对数据库中的数据进行查询、更新、添加、删除。

LINQ TO SQL 提供了对事务、视图、存储过程的完全支持。它同样为集成数据校验和业务层逻辑到你的数据模型中提供了一种简单的实现方式。

# 使用 LINQ TO SQL 对数据库进行建模

Visual Studio “Orcas”提供了一个 LINQ TO SQL 设计器，以提供一种简单、可视化的方式来进行数据库到对象的建模。我的下一篇博客随笔将更加深入的使用这个设计器(你也可以下载我在一月份录制的这个视频，这个视频记录了我是如何建立一个 LINQ TO SQL 模型的)。

使用 LINQ TO SQL 设计器，我可以很容易的建立对于范例数据库“NorthWind”的表现。如图所示：




上图所示的我的 LINQ TO SQL 设计来自于四个实体类：Product, Category, Order 和 OrderDetail。每个类的属性映射到相应的数据库表的字段中。每个类的实例代表了数据库中表

的一行。

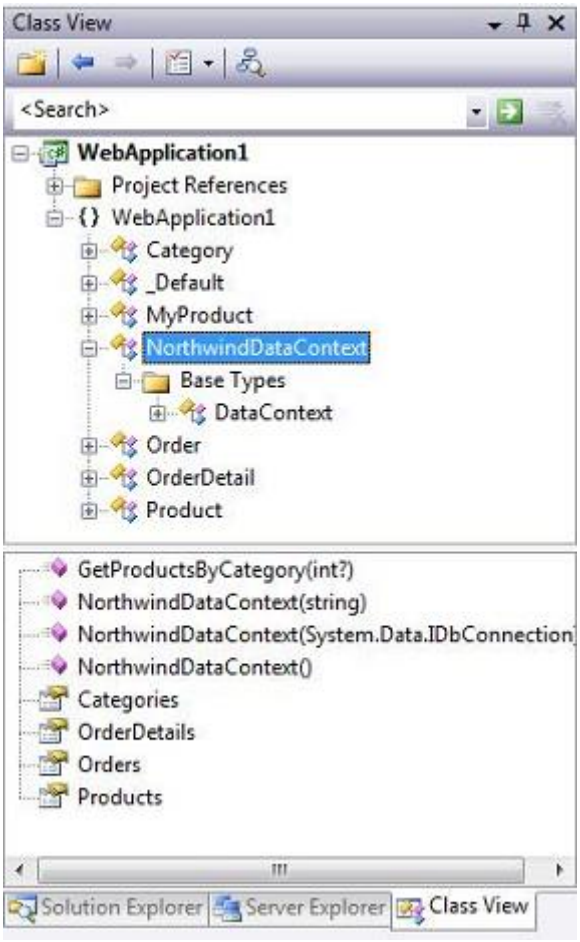
上面四个实体类之间的箭头代表了不同实体间的联系/关系。箭头的方向代表了这种关系是一对多的还是多对一的。强类型化的属性将根据这个添加到实体类中。举个例子，上面的 Category 类和 Product 类有一个一对多的关系。这就意味着 Category 类将有个“Categories”属性，这个属性是属于此 Category 的所有 Product 对象的集合。而 Product 类将有一个“Category”属性，这个属性指向它所属于的 Category 类。

在 LINQ TO SQL 设计器中的靠右边的长方形块中包含了与我们的数据库模型进行交互的存储过程列表。在这个范例中，我添加了一个简单的"GetProductsByCategory"存储过程。它用 CategoryId 作为输入参数，并且返回一个 Product 的实例集合。接下来让我们看看如何在代码中调用这个存储过程。

## 理解 DataContext 类

当你在 LINQ TO SQL 设计器中按“保存”按钮的时候，Visual Studio 将会维护我们建立模型的、代表着实体和数据库关系的那些类。每一个 LINQ TO SQL 设计器文件将会被添加到我们的解决方案中，同时也将创建一个一个自定义的 DataContext 类。这个 DataContext 类是一个主要的管道，我们通过它来进行对数据库的操作和查询。DataContext 类将包含一些属性，这些属性代表着我们在建模时的表和我们添加的存储过程。

举例来说，下面是一个 NorthwindDataContext 类，这个类严格的遵循我们在上面的设计。




# LINQ TO SQL 代码示范

一旦我们使用 LINQ TO SQL 设计器对我们的数据库建立了模型，我们可以轻易的编写代码来访问它。下面是一些代码范例，演示了一些基本的任务。

## 1）从数据库中查询 Product

下面的代码使用 LINQ 查询语法去获取一系列 Product 对象。注意到代码是如何通过 Product/Category 多对一关系来进行查询的，仅仅筛选出那些属于“Beverage”Category 的 Products。

```
NorthwindDataContext db = new NorthwindDataContext();

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

## 2)在数据库中更新一个 Product

下面的代码演示了如何从数据库中获取一个 Product，更新它的价格，然后存回数据库。

```
NorthwindDataContext db = new NorthwindDataContext();

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

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

db.SubmitChanges();
```

## 3)在数据库中插入一个新的 Category 和两个新的 Product。

下面的代码演示了如何创建一个新的 Category，然后创建了两个新的 Product，然后建立它们之间的联系。最后，它们全被存回数据库。

请注意，在下面我并不需要手动的设置 主键/外键 关系。相反，仅仅是添加一个 Product 对象到 Category 的“Products”集合，然后通过添加这个 Category 对象到 DataContext 的“Categories”集合，LINQ TO SQL 就会知道需要为我自动维护适当的 主键/外间 关系。

```
NorthwindDataContext db = new NorthwindDataContext();

// Create new Category and Products

Category category = new Category();
category.CategoryName = "Scott's Toys";

Product product1 = new Product();
product1.ProductName = "Toy 1";

Product product2 = new Product();
product2.ProductName = "Toy 2";

// Associate Products with Category

category.Products.Add(product1);
category.Products.Add(product2);

// Add category to database and save changes

db.Categories.Add(category);
db.SubmitChanges();
```

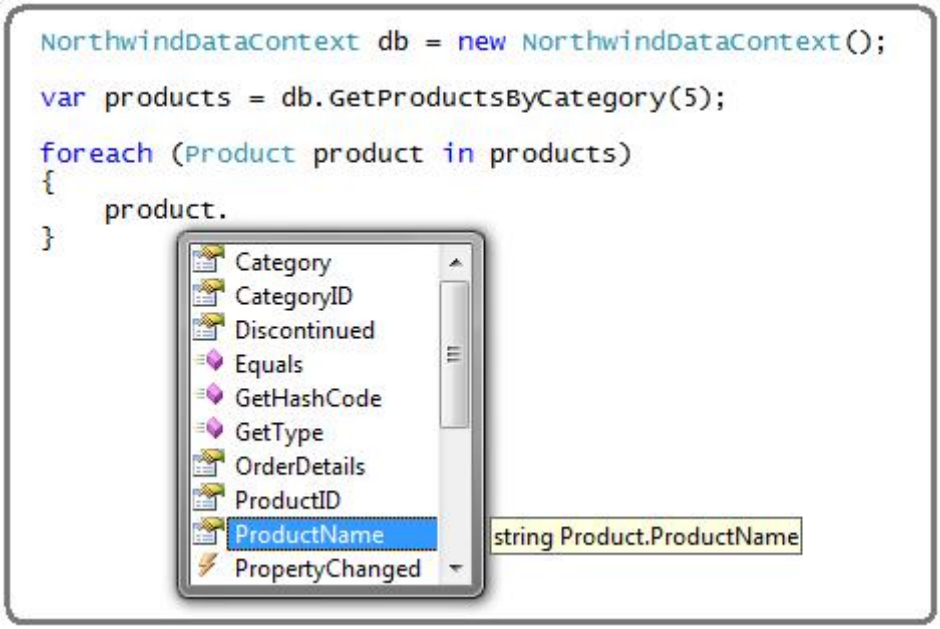
## 4)从数据库中删除 Products

下面的代码示范了如何删除数据库中所有的 Toy Products。

```
NorthwindDataContext db = new NorthwindDataContext();

var toyProducts = from p in db.Products
                  where p.ProductName.Contains("Toy")
                  select p;

db.Products.RemoveAll(toyProducts);

db.SubmitChanges();
```

## 5)调用一个存储过程

下面的代码示范了如何通过调用我们之前添加的 “GetProductsByCategory”存储过程来获取 Product 实体，而不是通过 LINQ 查询语法。请注意，一旦我获取了 Products 结果集，我可以更新、删除他们，然后调用 db.SubmitChanges()以将这些更改回传到数据库中去。




## 6)使用服务器端分页获取 Products

下面的代码演示了作为 LINQ 查询的一部分，如何实现高效的服务器端数据分页。通过使用下面的 Skip()和 Take()操作，我们将只从数据库中返回 10 行—初始 200 行。

```
NorthwindDataContext db = new NorthwindDataContext();

var products = (from p in db.Products
                where p.Category.CategoryName.StartsWith("C")
                select p).Skip(200).Take(10);
```

# 总结

LINQ TO SQL 提供了一种良好、清爽的方式对你应用程序中的数据层进行建模。一旦你定义好你的数据模型，你可以简单、高效的在它上面实现查询、插入、更新、删除。

希望上面的这些介绍和代码范例会帮助你满足学习更多支持的欲望。在接下来的几周里，我将会继续发表这一系列文章，更加深入地探讨 LINQ TO SQL。