 .NET Framework 3.5 中的 C# 和 VB 编译器支持 LINQ 的语法，这就使得它可以通过配置数据来存储，而不需要求助于 ADO.NET。

举个例子，在 Northwind 数据库中查询 Constomers 这张表，使用 C# 中的 LINQ ，代码应该是这样：

**var** data = **from** c **in** dataContext.Customers

**where** c.Country == "Spain"

**select** c;

其中：

* from关键字逻辑上依次通过每个集合。
* 包含关键字where的表达式会比较集合中的每个对象。
* select声明会选择被比较出的对象加入到列表中并返回。
* 关键字var用于变量声明。因为返回对象的准确类型不明确，它表明信息需要被动态的推测。

LINQ 查询语句可以应用在任何继承于 IEnumerable 的有数据支撑的类，这里 T 可以是任何一个数据类型，例如 List< Book >。

让我们来看一个示例理解一下概念。示例中使用了如下类:Book.cs

让我们来看一个示例理解一下概念。示例中使用了如下类:Book.cs

**public** **class** Books

{

**public** string ID {get; set;}

**public** string Title { get; set; }

**public** decimal Price { get; set; }

**public** DateTime DateOfRelease { get; set; }

**public** **static** List<Books> GetBooks()

{

List<Books> list = **new** List<Books>();

list.Add(**new** Books { ID = "001",

Title = "Programming in C#",

Price = 634.76m,

DateOfRelease = Convert.ToDateTime("2010-02-05") });

list.Add(**new** Books { ID = "002",

Title = "Learn Jave in 30 days",

Price = 250.76m,

DateOfRelease = Convert.ToDateTime("2011-08-15") });

list.Add(**new** Books { ID = "003",

Title = "Programming in ASP.Net 4.0",

Price = 700.00m,

DateOfRelease = Convert.ToDateTime("2011-02-05") });

list.Add(**new** Books { ID = "004",

Title = "VB.Net Made Easy",

Price = 500.99m,

DateOfRelease = Convert.ToDateTime("2011-12-31") });

list.Add(**new** Books { ID = "005",

Title = "Programming in C",

Price = 314.76m,

DateOfRelease = Convert.ToDateTime("2010-02-05") });

list.Add(**new** Books { ID = "006",

Title = "Programming in C++",

Price = 456.76m,

DateOfRelease = Convert.ToDateTime("2010-02-05") });

list.Add(**new** Books { ID = "007",

Title = "Datebase Developement",

Price = 1000.76m,

DateOfRelease = Convert.ToDateTime("2010-02-05") });

**return** list;

}

}

在 web 网页中使用这个类要有简单的标签控制，来显示书的标题。Page\_Load 方法创建了一个书的列表并且通过使用 LINQ 查询返回标题:

**public** partial **class** simplequery : System.Web.UI.Page

{

**protected** **void** **Page\_Load**(object sender, EventArgs e)

{

List<Books> books = Books.GetBooks();

var booktitles = from b in books select b.Title;

foreach (var title in booktitles)

lblbooks.Text += String.Format("{0} <br />", title);

}

}

## **LINQ 运算符**

### **join 子句**

SQL 中的‘join clause’用来连接两个数据表并显示在两个数据表中都出现的列中的数据集合。LINQ 也可以支持这种功能。

**var** booktitles = **from** b **in**

books **join** s **in** sales

on b.ID equals s.ID

**select** **new** { Name = b.Title, Pages = s.pages };

### **where 子句**

where 子句允许在查询中添加筛选条件。

法中的句柄成下述样子:

**var** booktitles = **from** b **in** books **join** s **in** sales on b.ID equals s.ID

**where** s.pages > 500 **select** **new** { Name = b.Title, Pages = s.pages };

### **正序倒序排序子句**

代码:

**var** booktitles = **from** b **in** books **join** s **in** sales on b.ID equals s.ID

**orderby** b.Price **select** **new** { Name = b.Title, Pages = s.pages, Price = b.Price};

### **Let 子句**

let 子句允许定义一个变量并且将数据计算的一个值赋给它。举个例子，计从上述两个销售值中计算总销售值，你需要这样计算:

TotalSale = Price of the Book \* Sales

为了完成这个算式，加入下面这个代码片段在 Page\_Load 方法的句柄里:

let 子句允许定义一个变量并且将数据计算的一个值赋给它。举个例子，计从上述两个销售值中计算总销售值，你需要这样计算:

**var** booktitles = **from** b **in** book **join** s **in** sales on b.ID equals s.ID

**let** totalprofit = (b.Price \* s.sales)

**select** **new** { Name = b.Title, TotalSale = totalprofit};