第9章 LINQ

目录

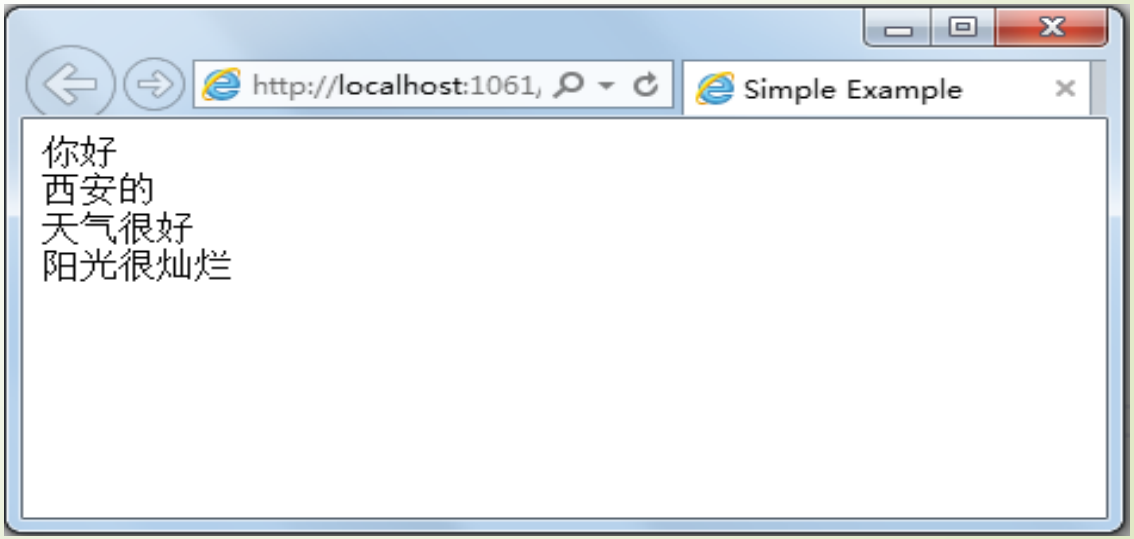
- LINQ概述
- 使用LINQ查询
- 使用LINQ对数据库进行操作
- LINQ中的数据绑定
- 实例

9.1 LINQ概述

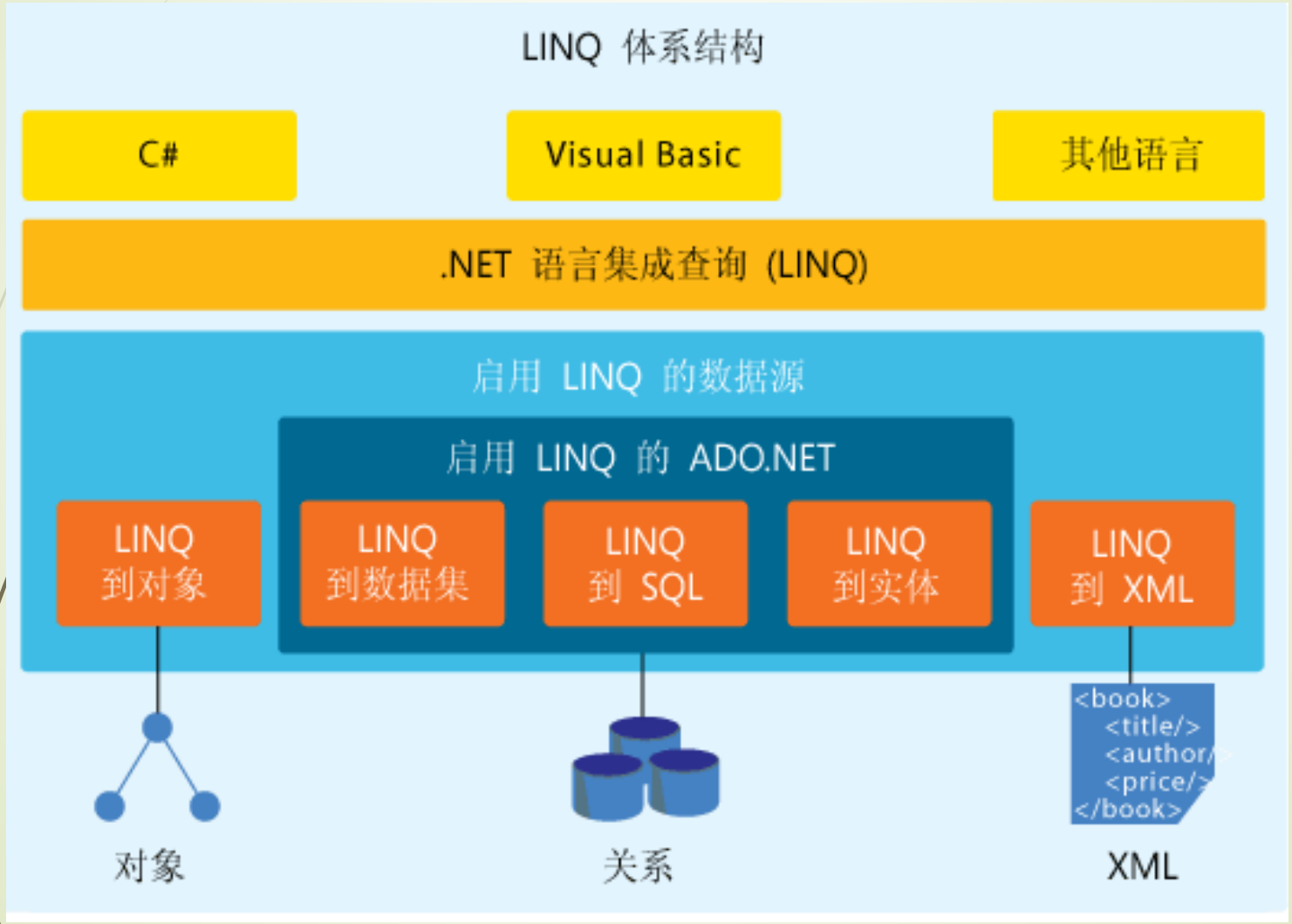
- LINQ（Language Integrated Query）是一种与.NET Framework中使用的编程语言紧密集成的查询语言。
- LINQ允许对各种类型的数据源进行查询，包括关系数据库、XML文档、甚至内存数据结构。

一个简单的LINQ查询例子

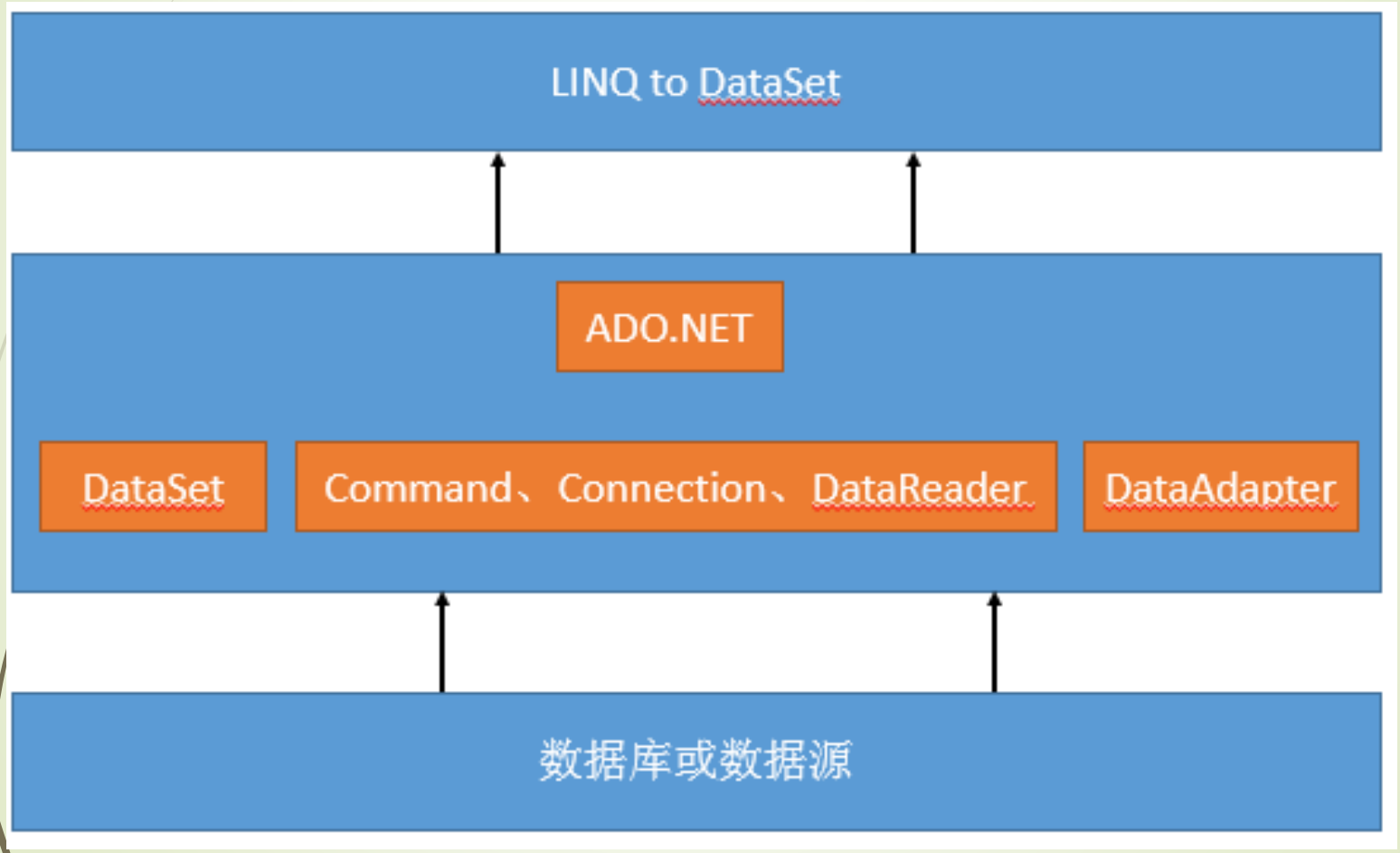
```
protected void Page_Load(object sender, EventArgs e)  {
    string[] str = { "你好", "西安的", "天气很好", "阳光很灿烂" };
    var q = from n in str select n;
    foreach (var n in q)  {
        Response.Write(n.ToString() + "<br/>");
    }
}
```



9.1.1 LINQ的体系结构



9.1.2 LINQ与ADO.NET的关系



9.2 使用LINQ查询

查询子句	说明
from子句	指定查询操作的数据源和范围变量
where子句	筛选元素的逻辑条件，一般由逻辑运算符组成
select子句	指定查询结果的类型和表现形式
orderby子句	对查询结果进行排序（升序或降序）
group子句	对查询结果进行分组
into子句	提供一个临时的标识符。该标识符可以引用join、group和select子句的结果
join子句	连接多个查询操作的数据源
let子句	引入用于存储查询表达中子表达式结果的范围变量

9.2.1 from子句

- from子句是LINQ查询语句中最基本的子句。与SQL查询语句不同的是，from关键字必须在LINQ查询语句的开始，后面跟随着项目名称和数据源。基本查询代码如下所示：

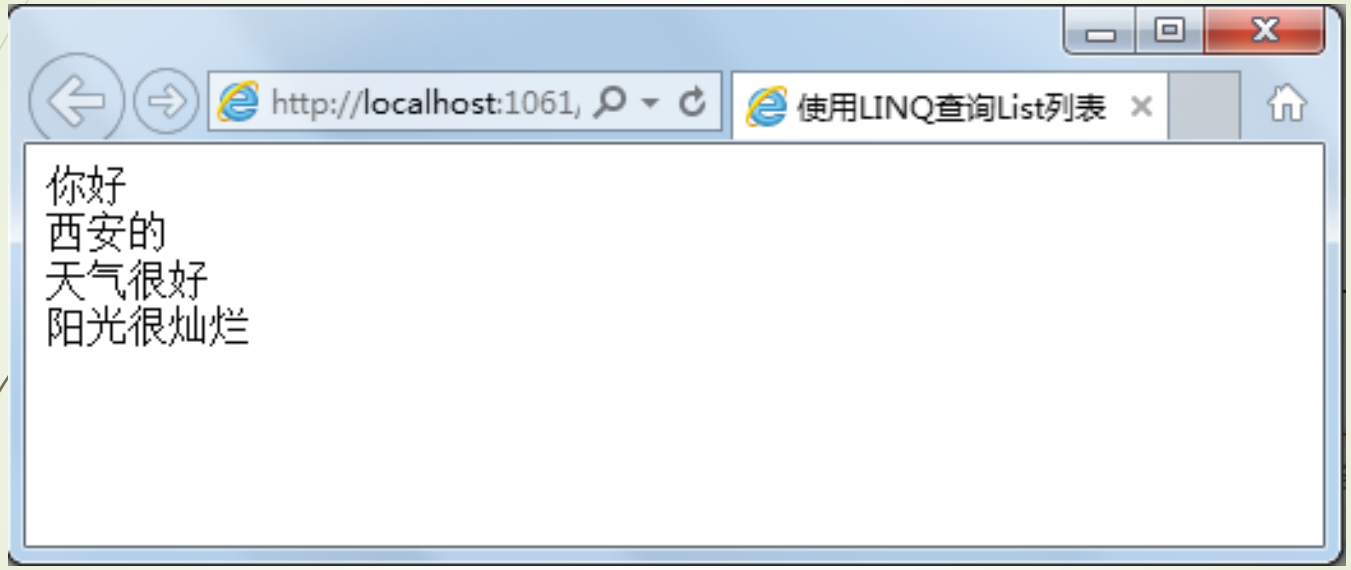
```
var query =  from q in dataSource
             select q;
```

- 使用LINQ对List列表进行查询的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)  {
    List<String> strList = new List<string>();                  //创建一个列表
      strList.Add("你好");                                        //添加数据
      strList.Add("西安的");
      strList.Add("天气很好");
      strList.Add("阳光很灿烂");
      var query = from q in strList                               //LINQ查询
                  select q;
     foreach (var q in strList)                                   //遍历集合
     {
        Response.Write(q.ToString() + "<br/>");                   //输出对象
     }
}
```

9.2.1 from子句

查询结果

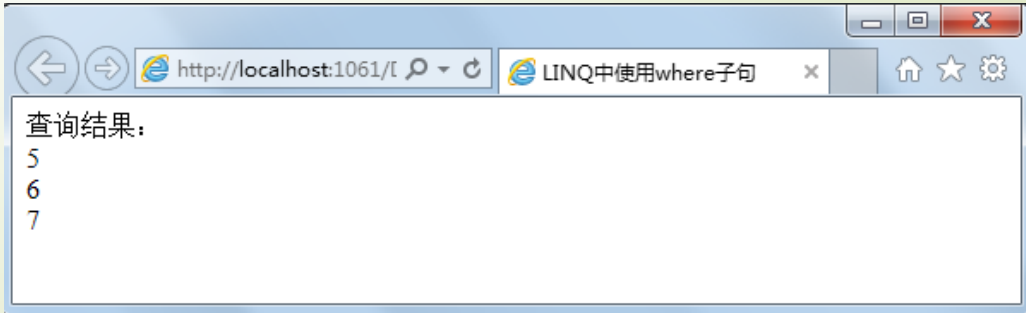


9.2.2 where子句

- where子句指定筛选元素的逻辑条件，一般由逻辑运算符（如逻辑与、逻辑或）组成。

使用where子句查询的示例代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)  {
    int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
    var query = from q in values                          //LINQ查询
                where q < 8 && q > 4                      //条件筛选
                select q;
    Response.Write("查询结果：<br/>");
    foreach (var q in query) {                            //遍历集合
        Response.Write(q.ToString() + "<br/>");           //输出对象
    }
}
```



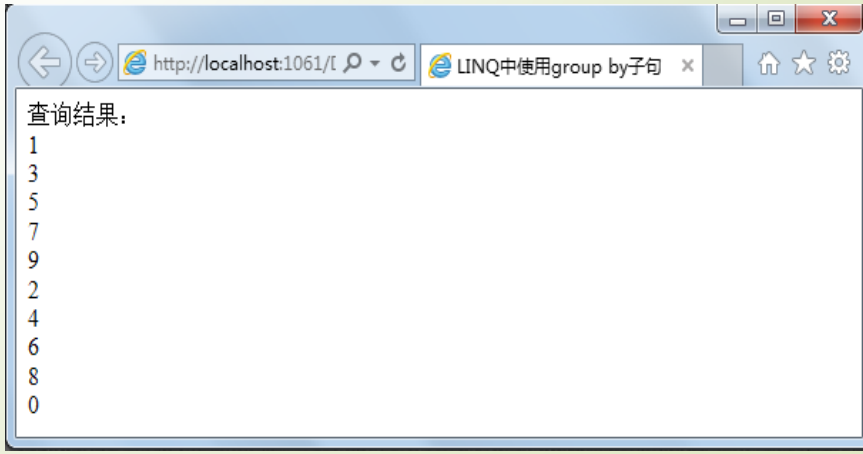
9.2.3 select子句

- select子句同from子句一样，是LINQ查询语句中必不可少的关键字。在LINQ查询表达式中，select子句指定查询结果的类型和表现形式。
- LINQ表达式必须以select子句或group子句结束。

9.2.4 group by子句

- group by子句对from语句执行查询的结果进行分组，并返回元素类型为IGrouping<TKey,TElement>的对象序列。
- 应用实例：

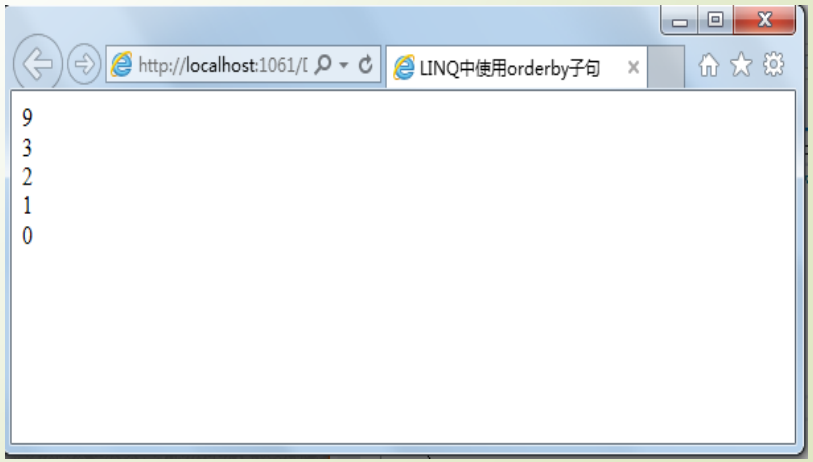
```
protected void Page_Load(object sender, EventArgs e)  {
    int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
    var query = from q in values
                group q by 0 == q % 2;
    Response.Write("查询结果：<br/>");
    foreach (var q in query)  {
        foreach(int p in q)  {
        Response.Write(p + "<br/>");
        }
    }
}
```



9.2.5 orderby子句

- orderby子句可以对查询结果进行排序。排序的方式可以为“升序”或“降序”，且排序的主键可以是一个或者多个。
- 应用实例：

```
protected void Page_Load(object sender, EventArgs e)  {
    int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
    var query = from q in values
                where q < 4 || q > 8
                orderby q descending
                select q;
    foreach (var q in query)
    {
        Response.Write(q + "<br/>");
    }
}
```



9.3 使用LINQ对数据库进行操作

- 创建数据源
- 数据库的查询
- 数据库的插入
- 数据库的更新
- 数据库的删除

9.3.1 创建数据源

1. 启动Visual Studio 2012开发工具，建议一个目标框架为Framework SDK v3.5以上的ASP.NET空网站；
2. 在“解决方案资源管理器”中，右击新建的项目，在弹出的快捷菜单中选择“添加”→“添加ASP.NET文件夹”→“App_Code”。右击App_Code文件夹，在弹出的快捷菜单中选择“添加”→“添加新项”命令；
3. 在模板列表中选择“LINQ to SQL”类，并将其命名为LinqData.dbml；
4. 在服务资源管理器中连接NorthWind数据库，将表Categories与表Products映射到LinqData.dbml中（将Categories表与Products表拖拽到设计视图中）；
5. LINQ数据源创建完毕，LinqDataContext类中的程序代码均自动生成。

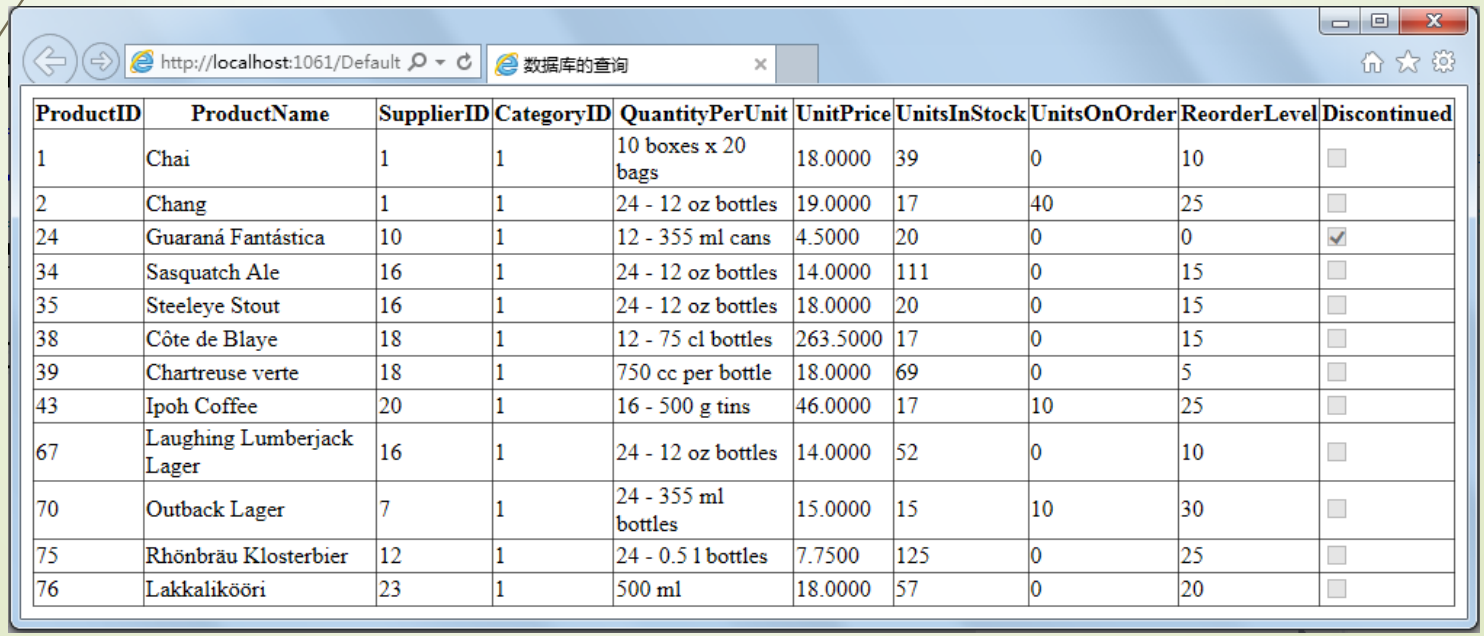
9.3.1 创建数据源

LinqDataDataContext　　mappingSource

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Linq.Expressions;
using System.Reflection;



[global::System.Data.Linq.Mapping.DatabaseAttribute(Name="Northwind")]
public partial class LinqDataDataContext : System.Data.Linq.DataContext
{

    private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();

  可扩展性方法定义

    public LinqDataDataContext() :
            base(global::System.Configuration.ConfigurationManager.ConnectionStrings["NorthwindConnectionSt
    {
        OnCreated();
    }

    public LinqDataDataContext(string connection) :
            base(connection, mappingSource)
    {
        OnCreated();
    }

    public LinqDataDataContext(System.Data.IDbConnection connection) :
            base(connection, mappingSource)
```

9.3.2 数据库的查询

1. 创建一个Web窗体Default.aspx，在Web窗体中添加一个GridView控件；
2. 声明LinqDataDataContext类对象linqDB；
3. 编写LINQ查询代码，使用LINQ查询表达式查询CategoryID=1的查询结果，并将查询结果保存到result变量中；
4. 将GridView控件的数据源设为result，并绑定数据。



ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Chai	1	1	10 boxes x 20 bags	18.0000	39	0	10	☐
2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25	☐
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5000	20	0	0	☑
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.0000	111	0	15	☐
35	Steeleye Stout	16	1	24 - 12 oz bottles	18.0000	20	0	15	☐
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5000	17	0	15	☐
39	Chartreuse verte	18	1	750 cc per bottle	18.0000	69	0	5	☐
43	Ipoh Coffee	20	1	16 - 500 g tins	46.0000	17	10	25	☐
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.0000	52	0	10	☐
70	Outback Lager	7	1	24 - 355 ml bottles	15.0000	15	10	30	☐
75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.7500	125	0	25	☐
76	Lakkalikööri	23	1	500 ml	18.0000	57	0	20	☐

9.3.3 数据库的插入

1. 创建一个Web窗体，并在其中添加相应数量的TextBox控件或DropdownList控件；
2. 添加一个Button控件，并在其Click事件下编写LINQ代码；
3. 声明实体类对象category，并设置该类对象中的实体属性，为实体属性赋值；
4. 使用InsertOnSubmit方法将实体类对象category添加到linqDB对象的Categories表中，然后调用SubmitChanges方法将实体类中数据添加到数据库中。

9.3.4 数据库的更新

1. 创建一个Web窗体，命名为Default.aspx；
2. 声明LinqDataContext类对象linqDB；
3. 使用LINQ查询需要修改的数据，更新需要修改的值；
4. 调用SubmitChanges()方法将更新的数据保存到数据库中。

9.3.4 数据库的更新

```
protected void Page_Load(object sender, EventArgs e)  {
        LinqDataDataContext linqDB = new
LinqDataDataContext( ConfigurationManager.ConnectionStri
ngs["NorthwindConnectionString"].ConnectionString.ToString());
        //查询要修改的数据
        var result = from q in linqDB.Products
                     where q.CategoryID == 1
                     select q;
        //更新数据
        foreach (Products product in result)
        {
            product.CategoryID = 2;
        }
        //执行更新
        linqDB.SubmitChanges();
}
```

9.3.4 数据库的删除

使用LINQ to SQL进行数据库的删除操作的步骤与更新基本一致，在查询到需要删除的数据之后，调用DeleteAllOnSumit()方法和SubmitChanges()方法，删除指定的数据并提交到数据库中

```
protected void Page_Load(object sender, EventArgs e)
    {
        LinqDataDataContext linqDB = new
LinqDataDataContext(ConfigurationManager.ConnectionStrin
gs["NorthwindConnectionString"].ConnectionString.ToString());
        //查询要修改的数据
        var result = from q in linqDB.Products
                     where q.CategoryID == 1
                     select q;
       //删除数据，并提交到数据库中
        linqDB.Products.DeleteAllOnSubmit(result);
        linqDB.SubmitChanges();
    }
```

9.4 LINQ中的数据绑定

9.4.1 LinqDataSource

LinqDataSource控件把在控件上设置的属性转换为LINQ查询，从而为应用程序中的目标数据对象生成查询。

配置LinqDataSource的主要步骤如下：

1.将LinqDataSource控件拖放到ASP.NET设计界面上，点击“配置数据源”；

2.选择上下文对象以及希望绑定的数据；

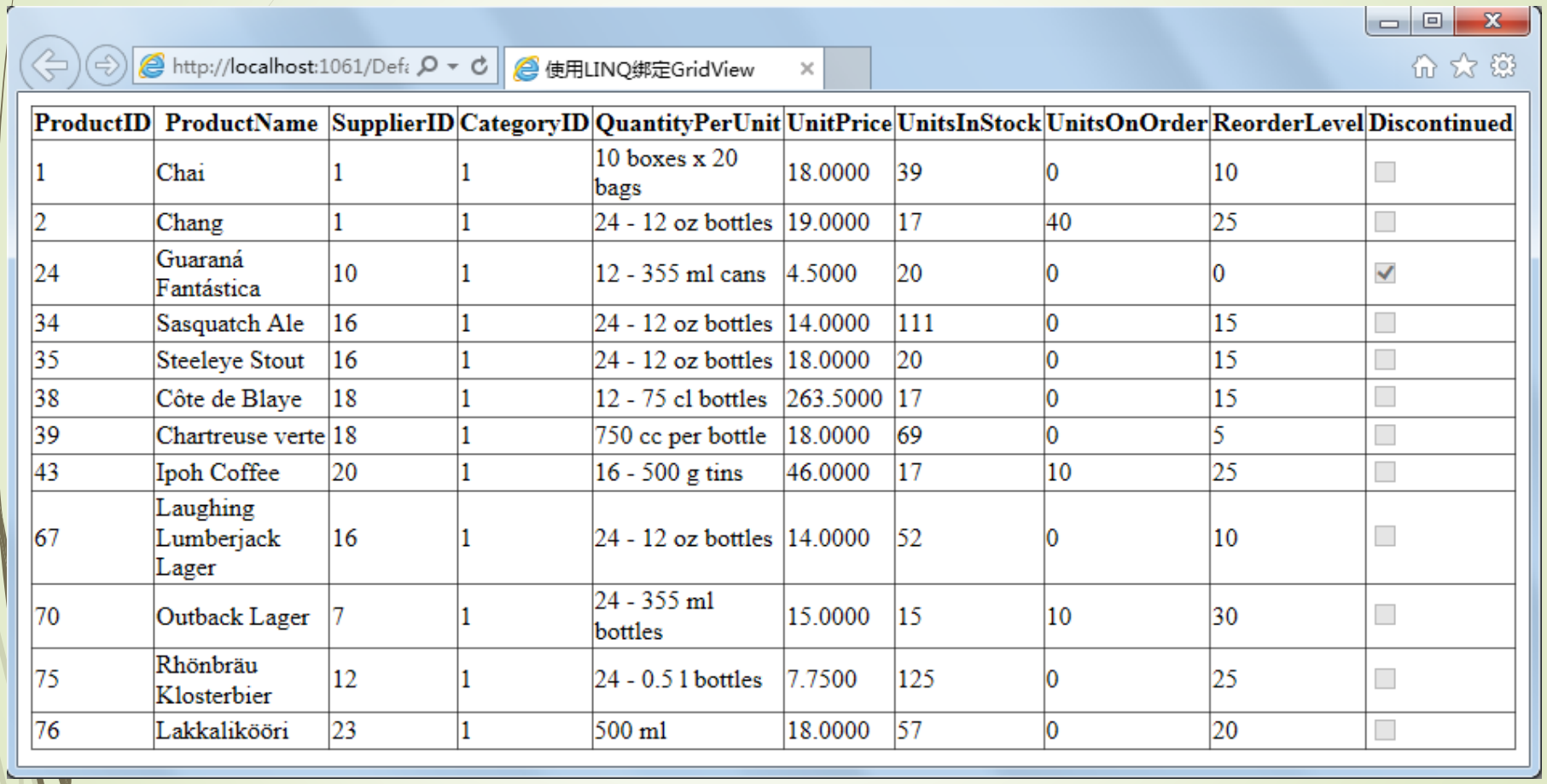
3.完成数据配置

9.4.2 数据绑定

1.数据源控件采用LinqDataSource

- 将LinqDataSource控件拖入ASP.NET设计页面中，点击“配置数据源”;
- 在选择数据源界面中，选择Products表，勾选“*”选项进行全部查询。点击右下方“where”按钮配置where表达式;
- 在列的下拉列表中选择“CategoryID”，运算符为“==”，源为“None”，参数属性值为“1”。点击“添加”按钮并确定，完成配置
- 将GridView控件拖入ASP.NET设计页面，在选择数据源下拉列表中选择刚刚配置好的数据源LinqDataSource1。保存并运行。

9.4.2 数据绑定



ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Chai	1	1	10 boxes x 20 bags	18.0000	39	0	10	☐
2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25	☐
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5000	20	0	0	☑
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14.0000	111	0	15	☐
35	Steeleye Stout	16	1	24 - 12 oz bottles	18.0000	20	0	15	☐
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5000	17	0	15	☐
39	Chartreuse verte	18	1	750 cc per bottle	18.0000	69	0	5	☐
43	Ipoh Coffee	20	1	16 - 500 g tins	46.0000	17	10	25	☐
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.0000	52	0	10	☐
70	Outback Lager	7	1	24 - 355 ml bottles	15.0000	15	10	30	☐
75	Rhönbräu Klosterbier	12	1	24 - 0.5 l bottles	7.7500	125	0	25	☐
76	Lakkalikööri	23	1	500 ml	18.0000	57	0	20	☐

9.4.2 数据绑定

2. 在后台代码中实现数据绑定

- 创建LINQ数据源文件;
- 将GridView控件拖入设计界面中，在Page_Load事件下编写实现代码;
- 创建LINQ查询表达式，其中，查询表达式既可以有中文字段，也可以有英文字段，其字段类型会在GridView控件中体现出来。将查询结果保存到result变量中;
- 将result变量存储的结果设置为GridView控件的数据源。

9.4.2 数据绑定

页面的Page_Load中代码如下所示

```
    protected void Page_Load(object sender, EventArgs e)  {
        LinqDataDataContext linqDB = new LinqDataDataContext(ConfigurationManager.ConnectionStrings["NorthwindConnectionString"].ConnectionString.ToString());
        //创建查询表达式
        var result = from q in linqDB.Products
                     where q.CategoryID == 1
                     select q;
        //绑定查询结果
        GridView1.DataSource = result;
        GridView1.DataBind();
    }
```

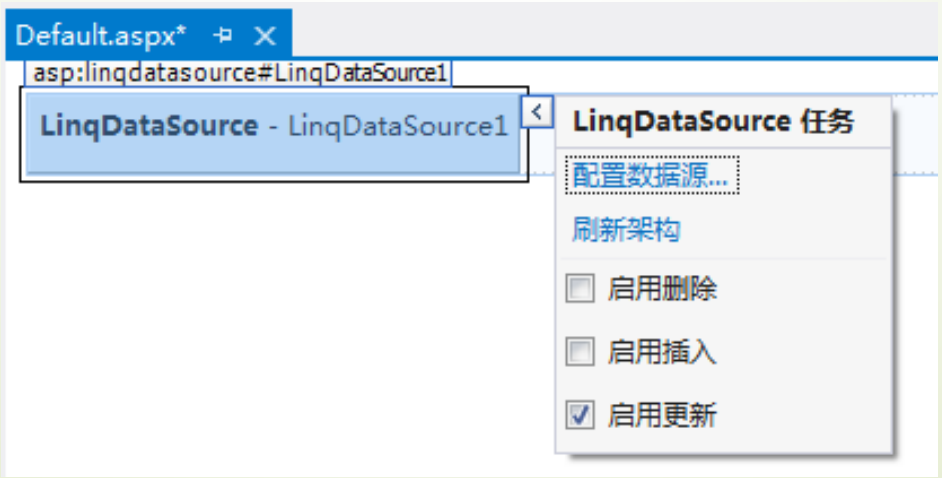
9.5实例：利用LINQ更新商品信息

- 设计说明

在电子商务网站中，常常需要系统管理员在后台对商品进行维护更新。利用LINQ技术能方便地进行更新操作。

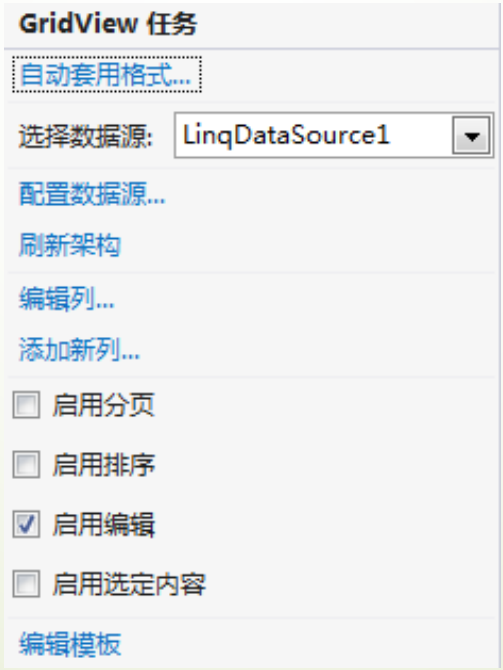
9.5实例：利用LINQ更新商品信息

- 程序实现
  - 在Default.aspx页面中，添加LinqDataSource控件。在选择数据源界面中，选择Products表，勾选“*”选项进行全部查询。配置数据源完成后，在LinqDataSource控件上，勾选启用更新操作；



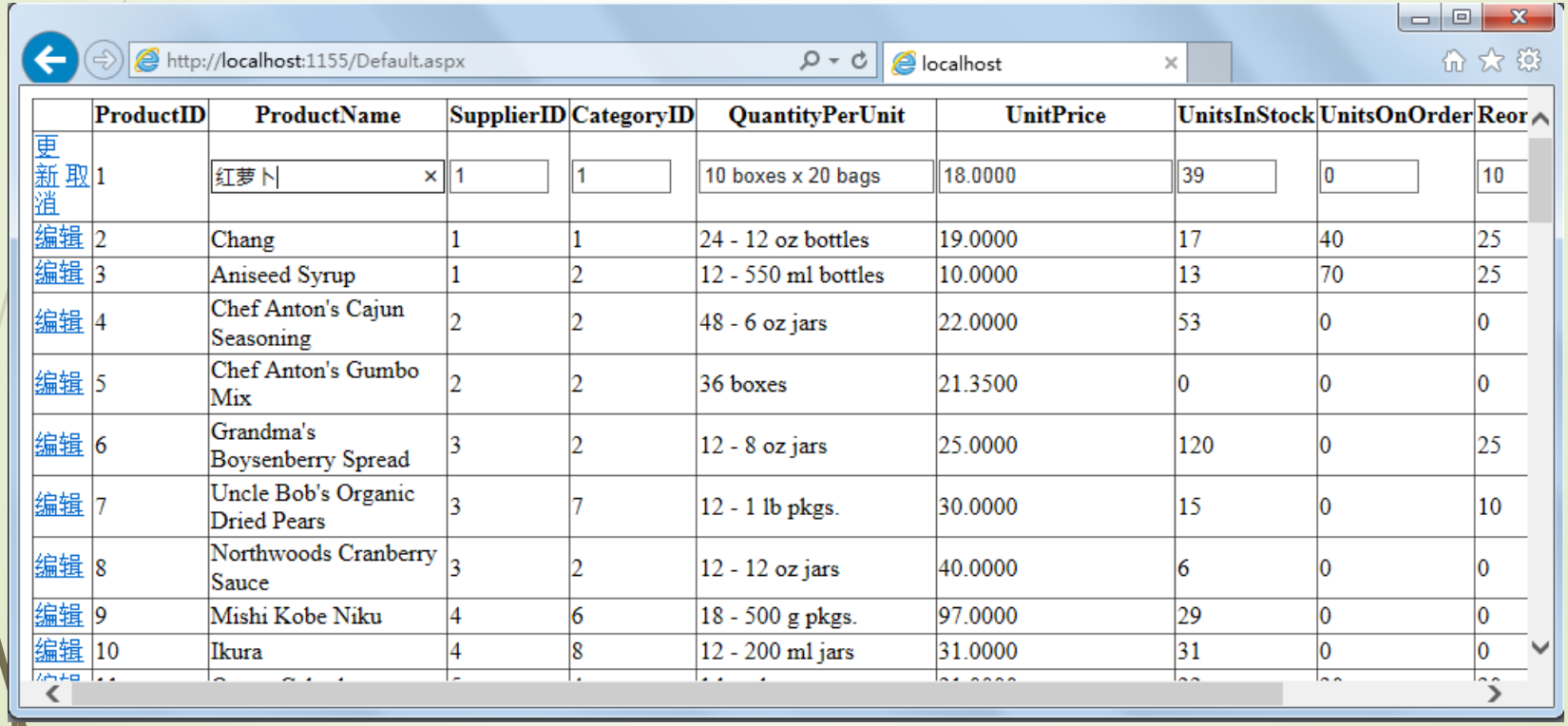
9.5实例：利用LINQ更新商品信息

- 程序实现
  - 添加GridView控件，并将数据源设置为LinqDataSource1，并勾选启用编辑选项；



9.5实例：利用LINQ更新商品信息

- 程序实现



	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	Reor
更新 取消	1	红萝卜	1	1	10 boxes x 20 bags	18.0000	39	0	10
编辑	2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25
编辑	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.0000	13	70	25
编辑	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.0000	53	0	0
编辑	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3500	0	0	0
编辑	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.0000	120	0	25
编辑	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0000	15	0	10
编辑	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.0000	6	0	0
编辑	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000	29	0	0
编辑	10	Ikura	4	8	12 - 200 ml jars	31.0000	31	0	0