第 5 讲　ASP.NET 数据库编程

本章学习目标

- ADO.NET 介绍
- 已连接环境
- 非连接的环境
- ADO.NET　对象模型
- ADO.NET 对象

5.1 ADO.NET 简介

数据访问简史

- 早期，每个数据库系统有独立的数据库访问函数，他们直接与数据库通讯，他们之间不能互操作。
- ODBC （ Open Database Connectivity ）开放数据库互联，提供了数据库系统的公共函数集，提供了数据库统一访问接口。
- OLE DB 的数据源可以是传统行列形式的数据库或任何其他存放数据的位置，这些数据源的数据都以表格的形式提供给应用程序，就像它来自数据库一样。（ DAO, RDO ）
- ADO, 位于 OLE DB 的顶部，它为开发基于 COM 的应用程序而设计。

5.1 ADO.NET 简介

ADO.NET 是一个包含在 Microsoft.NET 框架中的类库，它可以帮助 .NET 应用程序访问各种数据源。特点：

- 提供了丰富的类、接口、结构在 .NET 框架内处理数据访问
- ADO 革命性的、更可靠的继承者；
- 为非连接环境设计的系统；
- 支持通过传统数据访问接口使用 XML 数据。

ADO.NET 简介

水库 (数据库)
水管 （ 数据流 ） connection
水龙头 (控制水的) command
杯子 (水的集合 ） dataset

- 连接环境：水龙头 （ 始终打开)
- 非连接环境：水缸 （ 先开后关)

已连接环境

- 在已连接环境中，用户持续连接到数据源
- 优点：
  - 更安全，更易维护
  - 更容易控制并发性
  - 与非连接环境相比，数据刷新更及时
- 缺点：
  - 必须有固定的数据库连接
  - 扩展性不强

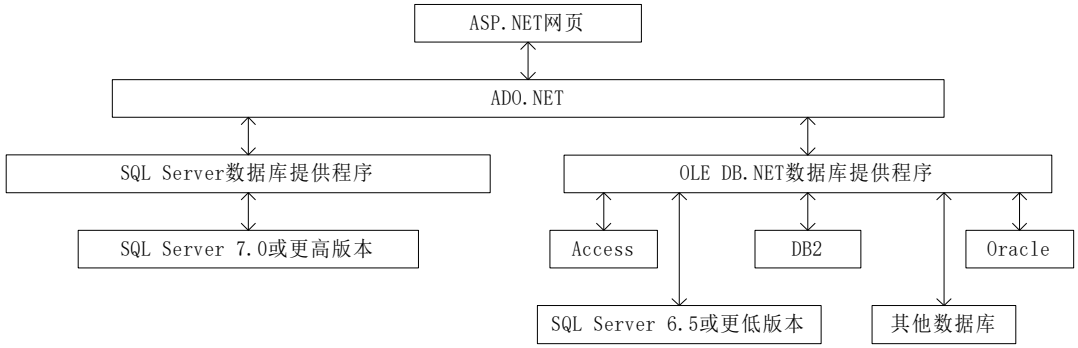
非连接的环境

- 在非连接环境中，用户可以在非连接的计算机上使用数据的子集，以后再将更新提交到数据源
- 优点
  - 任何时候都可用，并可随时连接到数据源进行处理
  - 共享连接资源
  - 提高了应用程序性能和扩展性
- 缺点
  - 数据不能保证是最新的
  - 可能发生更新冲突并且必须设法解决冲突

ADO. NET 的作用与组成

ASP.NET 访问数据源的方式

ADO.NET （ AvtiveX Data Objects.NET ）是 ASP.NET 与数据库的接口，其访问数据源的方式如图所示。



ADO.NET 是通过 .NET 数据库提供程序来访问数据源的。其中： SQL Server.NET 数据提供程序用于访问 Microsoft SQL Server 7.0 或更高版本的数据库，可提供很高的访问效率。 OLE DB.NET 数据提供程序则用于访问 Access 、 SQL Server 5.5 更低版本、 DB2 、 Oracle 或其他支持 OLE DB 驱动程序的数据库。

对商用系统而言， Access 数据库一般是不能满足系统对性能的要求的，应考虑使用 SQL Server 、 Oracle 等专业数据库。

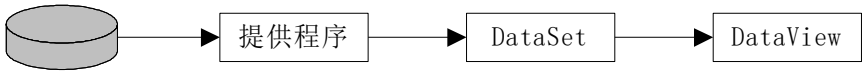
ADO. NET 的作用与组成

2 ADO.NET 的结构

ADO.NET 的对象内容如表所示。

对于复杂的数据库应用系统而言，表中比较重要的是 DataSet, 从数据库中取出数据后，须放到 DataSet，然后将其显示在浏览器中，数据流程如图所示。

对象	描述
Connection	与数据源建立连接
Command	对数据源执行操作命令并返回作结果
DataReader	从数据源提取只读、顺序的数据集
DataAdapter	在 DataSet 与数据源之间建立通道，将数据源中的数据写入 DataSet，或根据 DataSet 中的数据改写数据源。
DataSet	服务器内存中的数据库
数据库控件	用于显示 DataSet 中的数据



5.3 ADO.NET 对象模型

- Ado.net 有两个重要组成部分 DataSet 和 .Net 数据提供者。

- DataSet 类用于以表格形式在程序中放置一组数据，它可将数据以非连接的形式存储在缓存中并进行管理，它不关心数据来源。

- .Net 数据提供者包含针对不同数据源的组件，这些组件允许我们联接到单独的数据源上并与之通讯。数据提供者位于 System.Data 命名空间内。如：

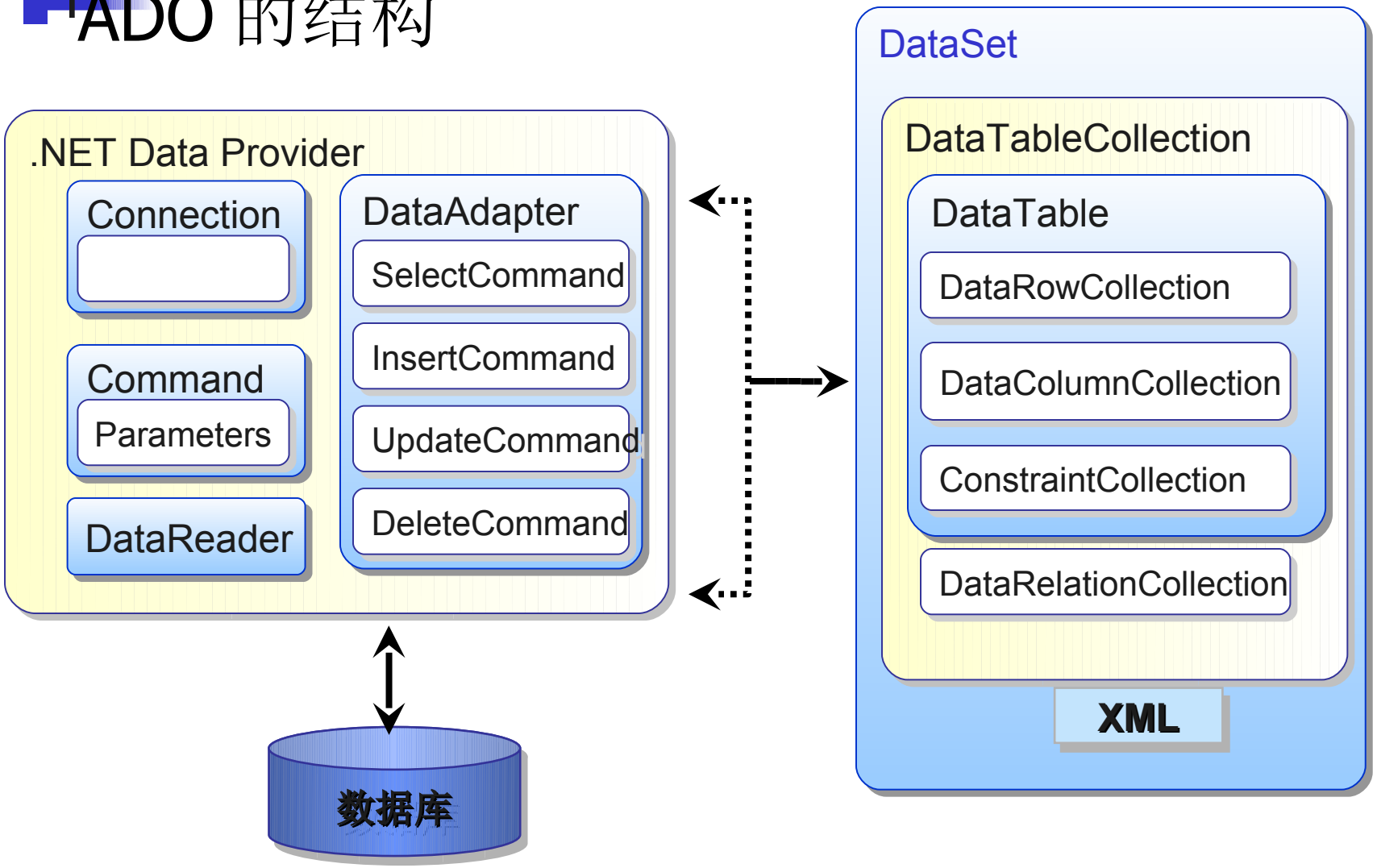
  System.Data.SqlClient

  System.Data.OleDb

  System.Data.Odbc

ADO.NET 对象模型

ADO 的结构



5.3.1 Connection 对象

使用 Connection 对象之前，对于 Access 数据库和 SQL Server 7.0 及以上版本数据库而言，应分别导入 System.Data 、 System.Data.OleDb 和 System.Data 、 System.Data.sql 命名空间。

建立与数据库连接的语法如下：

```
Connection  =New SqlConnection(connectionString());
'SQL Server 数据库          (OleDbConnection Acess 数据库 )
```

在建立连接对象后再指定其 ConnectionString 属性。

1． Connection 对象的属性

Connection 对象的常用属性如下：

ConnectionString：给出或设置连接参数。对 OLE DB 数据库而言，需要使用 OLE DB.NET 数据提供程序，对 SQL Server 7.0 及以上版本数据库而言，需要 SQL Server.NET 数据提供程序。

5.3.1 Connection 对象

对于 OLE DB 数据库，常用参数如表所示。下面是三个用来打开 Access 、 SQL Server 5.5 或更低版本及 Oracle 数据库的连接参数。

```
"Provider=Microsoft.JET.OLEDB.4.0;Data Source=F:\ssyyrr.mdb"
"Provider=SQLOLEDB;Data Source=syrwww;Integrated Security=SSPI"
```

"Provider=MSDAORA;Data Source=ORACLE8i7;User Id=syr;Password=syr001"

参数名称	用途
Data Source	设置数据源路径
Password	设置密码
Provider	设置驱动程序
User ID	设置帐号

5.3.1 Connection 对象

下面是一个用来打开 SQL Server 7.0 或更高版本数据库的连接参数。

```
SqlConnection  conn=null;
 //conn=new SqlConnection(@"user id=sa;data
source=localhost;initial catalog=book;password=local");
 //conn = new SqlConnection(@"Server =localhost;Database=book;
Integrated Security=SSPI");
```

2 . Connection 对象的方法

Connection 对象的主要方法如下：

- Open()：打开数据库。
- Close()：关闭数据库连接。当不再使用数据源时，应使用该方法关闭与数据源的连接。

```
//conn.Open();
```

5.3.2 Command 对象

Command 对象用于对数据源进行各种操作（如读取、写入记录等）。
建立 Command 对象的语法如下：

```
SqlCommand cmd=new SqlCommand();
```

1． Command 对象的属性
三个属性： CommandText CommandType Connection

```
cmd.Connection=conn;
cmd.CommandText="select * from dian";
```

2． Command 对象的方法
三个方法： ExecuteNonQuery() ExecuteReader() ExecuteScalar()

```
cmd.ExecuteNonQuery();
```

实例： 用户注册

5.4 DataReader 对象与记录读取

对于只需顺序显示数据表中记录的应用而言， DataReader 对象是比较理想的选择。

可以通过 Command 对象的 ExecuteReader() 方法创建 DataReader 对象。 DataReader 对象一旦建立，即可通过对象的属性、方法访问数据源中的数据。

建立 DataReader 对象的语法如下：

```
SqlDataReader dr=cmd.ExecuteReader();
```

1． DataReader 对象的属性

DataReader 对象的属性有下列几个：

- **FieldCount**：给出字段数目。
- IsClosed：给出 DataReader 对象的状态。 True 代表关闭， False 代表打开。
- Item(name | ordinal)：给出字段的内容。其中 name 为字段名称， ordinal 为字段序号。
- RecordsAffected：该属性在 DataReader 对象被关闭后有效。给出执行 Insert 、 Delete 或 Update 等命令后受影响的记录数。

5.4 DataReader 对象与记录读取

2 ．DataReader 对象的方法

- DataReader 对象的常用方法有下列几个：
- Close()：关闭 DataReader 对象。
- GetName(ordinal)：给出第 ordinal+1 个字段的字段名称。
- GetOrdinal(name)：给出名称为 name 的字段的序号。
- GetValue(ordinal)：给出第 ordinal+1 个字段的内容。
- GetValues(values)：将所有字段内容放入 values 数组。
- IsDbNull(ordinal)：给出一个布尔值，表示第 ordinal+1 个字段的内容是否为空。
- Read()：读取一条记录，并返回一个布尔值，表示本记录是否存在。执行 Read() 方法后，若返回值为 False，则意味着已经完成对所有记录的读取。

5.4 DataReader 对象与记录读取

4 . DataReader 对象应用举例

下面是一个利用 DataReader 对象读取一个数据表中所有记录的例子。其运行结果如图所示。

Ex1: 用 datareader 完成注册 .

Ex2: 浏览 northwind 数据库中客户表的信息 .

5.5 数据集的基本概念

在 ASP.NET 中，数据集（ DataSet ）对象是进行数据库处理的核心部件。 DataSet 是一个位于内存中的数据库，该数据库中的内容由程序设计者直接用程序建立，也可以从已经存在的数据库中传入。以后一方式居多。

数据集（ DataSet ）对象具有下列特点：

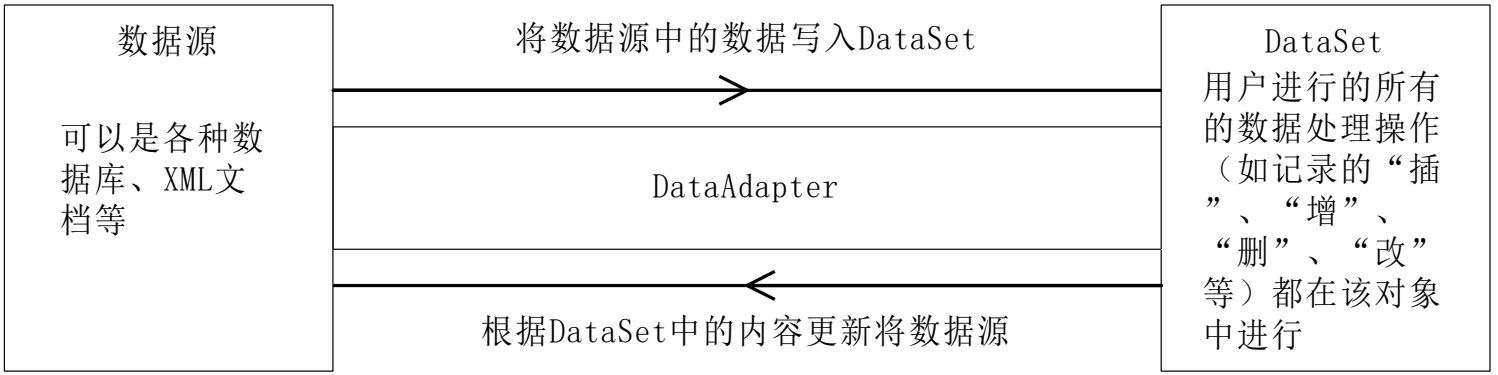
- 利用 DataSet ，可在内存中得到一个数据库，该数据库中的数据表可来自不同的物理数据库；
- DataSet 使用“无连接传输模式”访问数据源；
- 通过对 Adapter 进行设置和编程，可以十分方便地将 DataSet 中的记录回写至数据源中；
- ASP.NET 中可与数据库结合的控件都只能接受 DataSet 中的记录，而与实际的数据源无直接联系。

5.5 数据集的基本概念

在 ASP.NET 中，将 DataSet 与数据源相联系的对象被称为 DataAdapter （数据适配器）。数据源、 DataAdapter 与 DataSet 三者的关系如图所示。

DataSet 对象最常用的属性是 Tables，通过该属性，可以获得或设置数据表行、列的值。例如：

ds.Tables["students"].Rows[i][j] // 表示访问 students 表的第 i 行第 j 列



创建数据集对象的语句格式如下：

```
DataSet ds = new DataSet ();
```

或者

```
DataSet ds = new DataSet ("数据集名");
```

语句中 ds 代表数据集对象。可以通过调用 DataSet 的两个重载构造函数来创建 DataSet 的实例，并且可以选择指定一个名称参数。如果没有为 DataSet 指定名称，则该名称会设置为“NewDataSet”。

DataSet 对象最常用的属性是 Tables，通过该属性，可以获得或设置数据表行、列的值。例如：

```
ds.Tables["students"].Rows[i][j]          // 表示访问 students 表的第 i 行第 j 列
```

DataSet 对象的常用方法有 Clear() 和 Copy()，Clear() 方法清除 DataSet 对象的数据，删除所有 DataTable 对象；Copy() 方法复制 DataSet 对象的结构和数据，返回值是与本 DataSet 对象具有同样结构和数据的 DataSet 对象。

1．数据表和数据表集合

（1）数据表（DataTable）

创建 DataTable 时，不需要为 TableName 属性提供值，可以在其他时间指定该属性，或者将其保留为空。但是，在将一个没有 TableName 值的表添加到 DataSet 中时，该表会得到一个从“Table”（表示 Table0）开始递增的默认名称 TableN。可以使用相应的 DataTable 构造函数创建 DataTable 对象。例如：

```
DataTable workTable = new DataTable("Customers");   //创建DataTable对象并指定名称为Customers
```

表 7.6 列出了 DataTable 对象的常用属性和常用方法。

表 7.6 DataTable 对象的常用属性和常用方法

属性 / 方法	说　　明
Columns	获取数据表的所有字段，即 DataColumnCollection 集合
DataSet	获取 DataTable 对象所属的 DataSet 对象
DefaultView	获取与数据表相关的 DataView 对象。 DataView 对象可用来显示 DataTable 对象的部分数据。可通过对数据表选择、排序等操作获得 DataView （相当于数据库中的视图）
PrimaryKey	获取或设置数据表的主键
Rows	获取数据表的所有行，即 DataRowCollection 集合
TableName	获取或设置数据表名
Copy()	复制 DataTable 对象的结构和数据，返回与本 DataTable 对象具有同样结构和数据的 DataTable 对象
NewRow()	创建一个与当前数据表有相同字段结构的数据行
GetErrors()	获取包含错误的 DataRow 对象数组

（2）数据表集合（DataTableCollection）

DataSet 的所有数据表包含于数据表集合 DataTableCollection 中，通过 DataSet 的 Tables 属性访问 DataTableCollection 。 DataTableCollection 有以下两个属性。

① Count： DataSet 对象所包含的 DataTable 个数。

② Tables[index,name]：获取 DataTableCollection 中下标为 index 或名称为 name 的数据表。例如：

```
ds.Tables[0]                    // 表示数据集对象 ds 中的第一个数据表
ds.Tables[1]                    // 表示数据集对象 ds 中的第二个数据表
ds.Tables["students"]           // 表示数据集对象 ds 中名称为“students”的数据表
```

可以通过使用 Add 方法将 DataTable 添加到 DataTable 对象的 Tables 集合中，将其添加到 DataSet 中。例如：

```
ds.Tables.Add("CustomersTable");      // 将数据表 CustomersTable 添加到数据集 ds 中
```

2．数据行和数据行集合

（1）数据行（DataRow）

数据表中的每个数据行都是一个 DataRow 对象。DataRow 对象是给定数据表中的一行数据，或者说是数据表中的一条记录。DataRow 对象的方法提供了对表中数据的插入、删除、更新和查询等功能。提取数据表中的行的语句如下：

```
DataRow dr = dt.Rows[n];        //提取数据表中的行
```

其中，DataRow 代表数据行类；dr 是数据行对象；dt 代表数据表对象；n 代表行的序号（序号从 0 开始）。

DataRow 对象的属性主要有：

① Rows[index,columnName]：获取或设置指定字段的值。

② Table：获取包含该数据行的 DataTable 对象。

DataRow 对象的方法主要有：

① AcceptChanges()：将所有变动过的数据行更新到 DataRowCollection。

② Delete()：删除数据行。

③ IsNull({colName,index,Column 对象名})：判断指定列或 Column 对象是否为空值。

（２）数据行集合（DataRowCollection）

数据表的所有行都被存放于数据行集合 DataRowColection 中，通过 DataTable 的 Rows 属性访问 DataRowCollection 。例如：

```
stuTable.Rows[i][j]                    // 表示访问 stuTable 表的第 i 行、第 j 列数据
```

3．数据列和数据列集合

（1）数据列（DataColumn）

获取某列的值需要在数据行的基础上进行。其语句格式如下：

```
string dc = dr.Columns["字段名"].ToString();
```

或者

```
string dc = dr.Column[index].ToString();
```

两条语句具有同样的作用。其中 dr 代表引用的数据行，dc 是该行某列的值（用字符串表示），index 代表列（字段）对应的索引值（列的索引值从 0 开始）。

综合前面的语句，要取出数据表（dt）中第 3 行记录中的“姓名”字段，并将该字段的值放入一文本框（textBox1）中，语句可以写成：

```
DataTable dt = ds.Tables["Customers"]                    // 从数据集中提取数据表 Customers
DataRow dRow = dt.Rows[2 ];                              // 从数据表中提取了第 3 行记录
string textBox1.Text=dRow["CompanyName"].ToString();// 从行中取出名为 CompanyName 字段的值
```

表 7.7 列出了 DataColumn 对象的常用属性。

表 7.7 DataColumn 对象的常用属性

属　　性	说　　明
AllowDBNull	设置该字段可否为空值。默认值为 true
Caption	获取或设置字段标题。若未指定字段标题，则字段标题即为字段名
ColumnName	获取或设置字段名
DataType	获取或设置字段的数据类型
DefaultVale	获取或设置新增数据行时，字段的默认值 .
ReadOnly	获取或设置新增数据行时，字段的值是否可修改。默认值为 false
Table	获取包含该字段的 DataTable 对象

通过 DataColumn 对象的 DataType 属性设置字段数据类型时，不可直接设置数据类型，而需按照以下语法格式：

DataColumn 对象名 .DataType = typeof （数据类型）

其中的“数据类型”取值为 .NET Framework 数据类型。

（2）数据列集合（DataColumnCollection）

数据表中的所有字段都被存放于数据列集合 DataColumnColection 中，通过 DataTable 的 Columns 属性访问 DataColumnCollection 。例如：

stuTable.Columns[i].Caption　　// 代表 stuTable 数据表的第 i 个字段的标题

DataColumnColection 有以下 2 个属性：

① Count ：数据表所包含的字段个数。

② Columns[index,name] ：获取下标为 index 或名称为 name 的字段。例如：

DS.Tables[0].Columns[0]　　// 表示数据表 DS.Tables[0] 中的第一个字段

DS.Tables[0].Columns["studentid"]　　// 表示数据表 DS.Tables[0] 的字段名为 studentid 的字段

【例 7-12】 利用 DataSet 在内存中创建数据表 Person，给表定义 4 个字段：姓名、性别、专业、出生日期。向表内插入一条记录，并显示到页面上。

设计步骤如下：

（1）在网站 Chapter7 中加入一个页面“example7-12.aspx”，拖放一个 GridView 控件到页面上。

（2）在 Page_Load 方法内添加如下代码。

（3）按【Ctrl+F5】组合键运行网页，结果如图 7.33 所示。

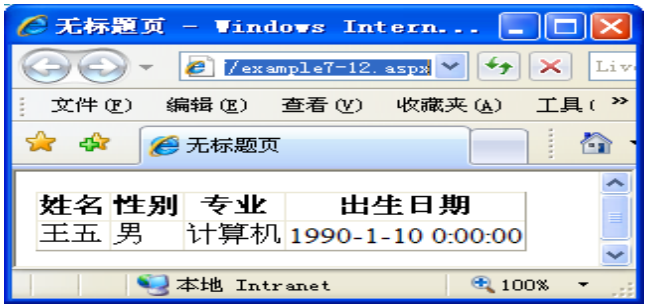


图 7.33 example7-12 网页的运行结果

4． DataAdapter

如果所连接的是 SQL Server 数据库，则可以将 SqlDataAdapter 与关联的 SqlCommand 和 SqlConnection 对象一起使用，从而提高总体性能。对于支持 OLEDB 的数据源，可以使用 OleDbDataAdapter 及其关联的 OleDbCommand 和 OleDbConnection 对象。对于支持 ODBC 的数据源，可使用 OdbcataAdapter 及其关联的 OdbcCommand 和 OdbcConnection 对象。对于 Oracle 数据库，可使用 OracleDataAdapter 及其关联的 OracleCommand 和 OracleConnection 对象。

定义 DataAdapter 对象的语法格式有 4 种：

OleDbDataAdapter 对象名 = new OleDbDataAdapter()

OleDbDataAdapter 对象名 = new OleDbDataAdapter(OleDbCommand 对象)

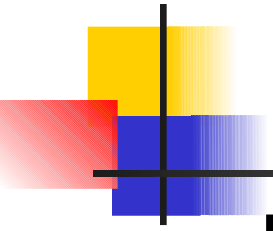
OleDbDataAdapter 对象名 = new OleDbDataAdapter (SQL 命令, OleDbConnection 对象)

OleDbDataAdapter 对象名 = new OleDbDataAdapter (SQL 命令, OleDbConnection 对象)

创建 SqlDataAdapter 对象语法格式与之类似，只要将所有的“ OleDb”改为“ Sql” 即可。

DataAdapter 有一个重要的 Fill 方法，此方法将数据填入数据集，语句如下：

dataAdapter1.Fill (dataSet1, "Products")



DataAdapter 四个命令

- Selectcommand
- Insertcommand
- Updatecommand
- deletecommand

Ex1: 浏览客户 dataadapter.aspx

```
SqlConnection conn = new SqlConnection("data source=localhost;initial
catalog=northwind; user id =sa; password=local");
    conn.Open();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select * from Categories";
    SqlDataAdapter adapter = new SqlDataAdapter();
    adapter.SelectCommand = cmd;
    DataSet ds = new DataSet();
    adapter.Fill(ds, " Categories");
    Response.Write("<table> <tr> 客户表 </tr>");
    for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
    {
        Response.Write("<tr>");
        for (int j = 0; j < ds.Tables[0].Columns.Count - 1; j++)
            Response.Write("<td>" + ds.Tables[0].Rows[i][j] + "</td>");
        Response.Write("</tr>");
    }
    Response.Write("</table>");
    conn.Close();
```

利用 DataSet 显示记录

Ex2: 用 gridview 浏览客户 dataapdater.aspx
Ex3: 浏览 employees 表

```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False">
      <Columns >
       <asp:BoundField HeaderText =" 编号 "
ReadOnly ="True" DataField ="CategoryID" />
       <asp:BoundField  HeaderText =" 名称 " ReadOnly
="True"  DataField ="Categoryname"/>
       </Columns>
       </asp:GridView>
```

Dataset 数据更新— dataadapter 的 update ()

如何生成 Insertcommand,Updatecommand,deletecommand 属性以及使其能对应到 selectcommand 的命令，以便能保证正确的写回数据库呢？——commandbuilder 对象

eg: 数据添加

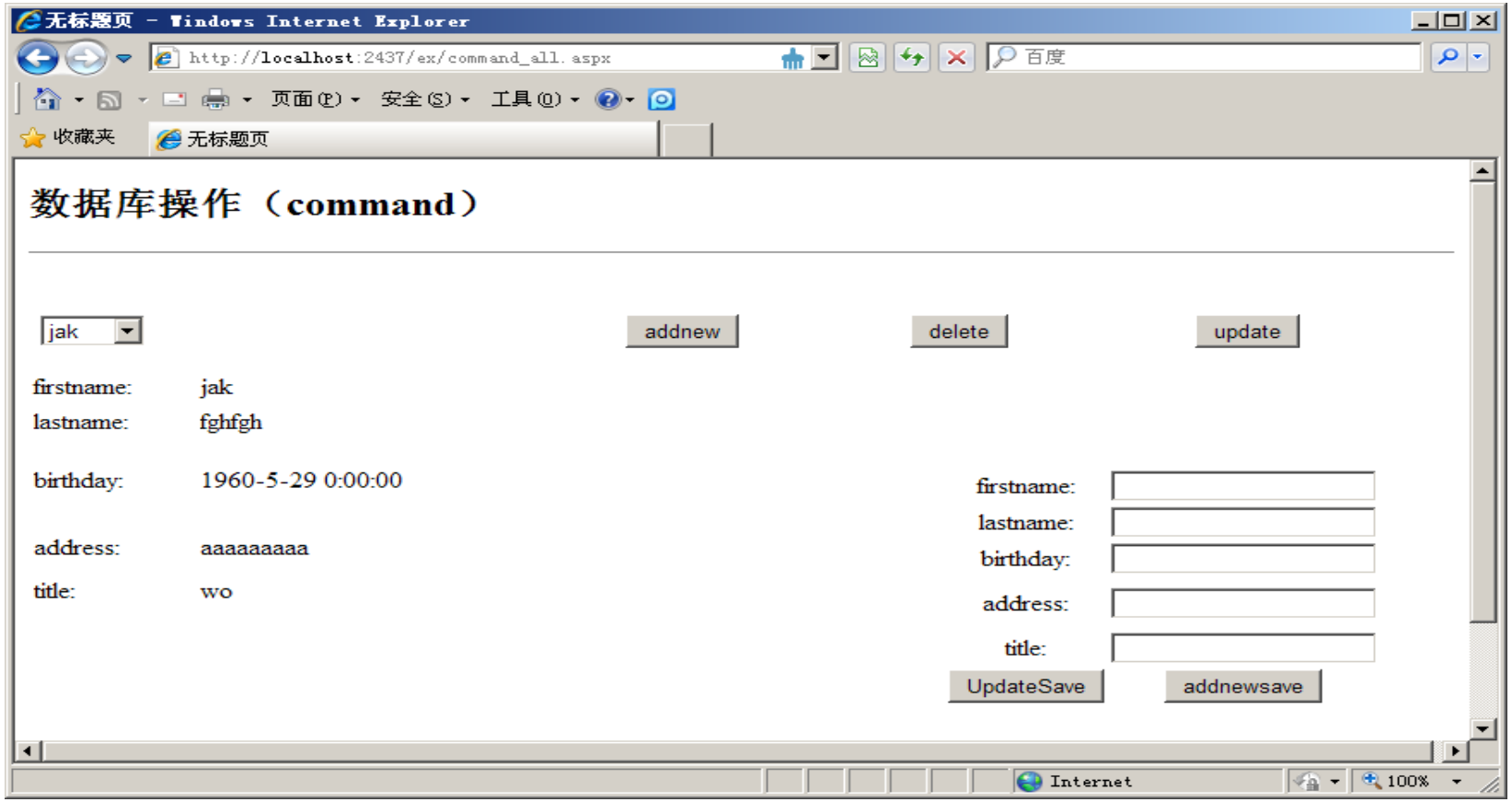
```
SqlDataAdapter ad = new SqlDataAdapter(strsql, conn);
SqlCommandBuilder cmb = new SqlCommandBuilder(ad);
ad.Fill(ds, "Employees");
DataTable dt = ds.Tables["Employees"];
DataRow row = dt.NewRow(); // 添加行
row["FirstName"] = TextBox1.Text;
row["LastName"] = TextBox2.Text;…
dt.Rows.Add(row);
```

删除行 / 更新

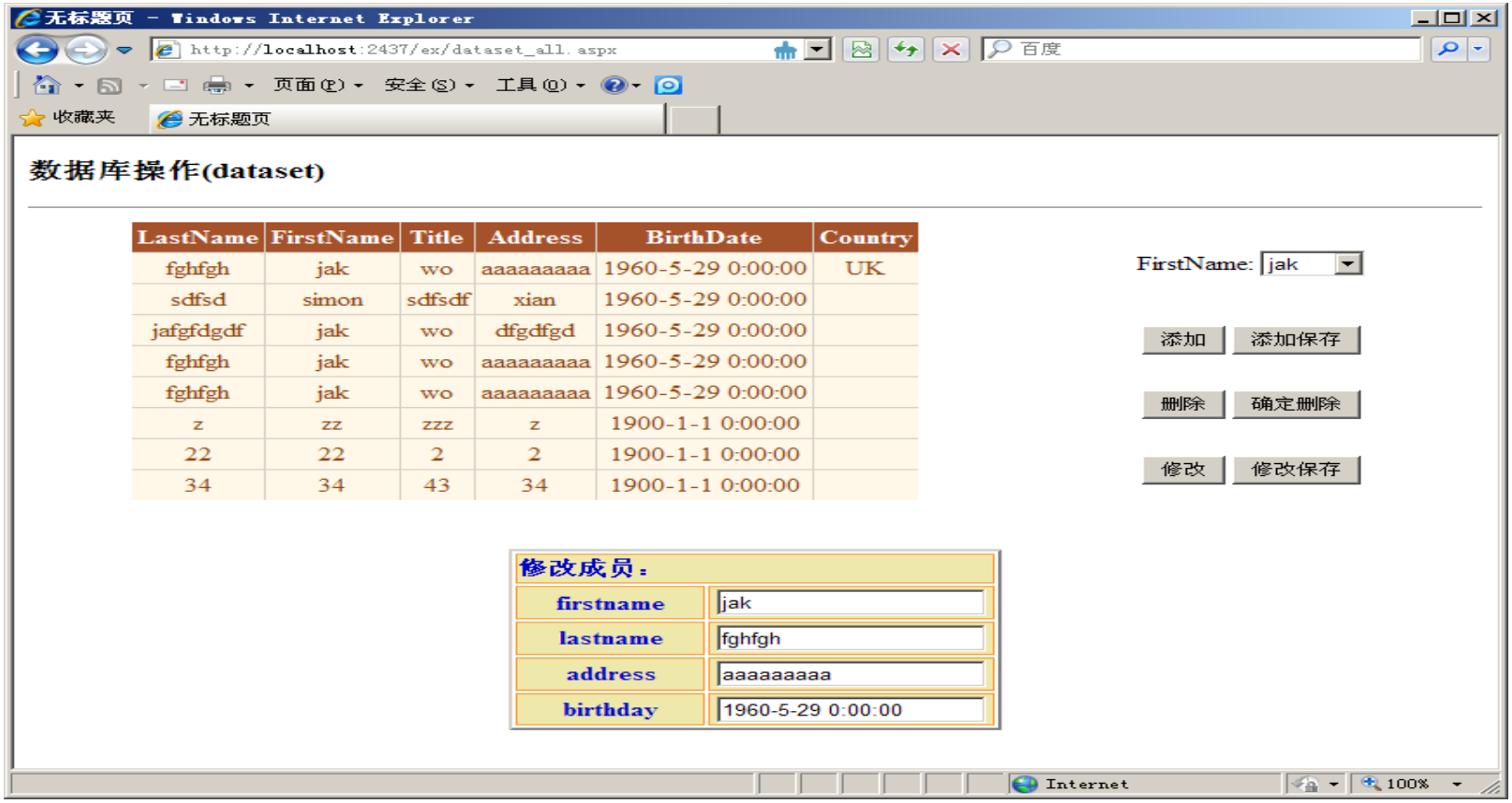
```
string strsql = "select * from Employees where EmployeeID=" + DropDownList1.SelectedValue;
SqlDataAdapter ad = new SqlDataAdapter(strsql, conn);

SqlCommandBuilder cmb = new SqlCommandBuilder(ad);
DataSet ds = new DataSet();
ad.Fill(ds, "Employees");
DataTable dt = ds.Tables["Employees"];
DataRow row = dt.Rows[0];
row.Delete() ;// 更新 row["FirstName"] = TextBox1.Text;…
ad.Update(ds, "Employees");
```

综合实训 1


无标题页 - Windows Internet Explorer
http://localhost:2437/ex/command_all.aspx
百度
页面(P) 安全(S) 工具(O)
收藏夹
无标题页
数据库操作（command）
jak
addnew
delete
update
firstname: jak
lastname: fghfgh
birthday: 1960-5-29 0:00:00
address: aaaaaaaaa
title: wo
firstname:
lastname:
birthday:
address:
title:
UpdateSave
addnewsave
Internet
100%

综合实训 2


无标题页 - Windows Internet Explorer
http://localhost:2437/ex/dataset_all.aspx
百度
页面(P) 安全(S) 工具(O)
收藏夹
无标题页
数据库操作(dataset)
LastName FirstName Title Address BirthDate Country
fghfgh jak wo aaaaaaaaa 1960-5-29 0:00:00 UK
sdfsd simon sdfsdf xian 1960-5-29 0:00:00
jafgfdgdf jak wo dfgdfgd 1960-5-29 0:00:00
fghfgh jak wo aaaaaaaaa 1960-5-29 0:00:00
fghfgh jak wo aaaaaaaaa 1960-5-29 0:00:00
z zz zzz z 1900-1-1 0:00:00
22 22 2 2 1900-1-1 0:00:00
34 34 43 34 1900-1-1 0:00:00
FirstName: jak
添加 添加保存
删除 确定删除
修改 修改保存
修改成员：
firstname jak
lastname fghfgh
address aaaaaaaaa
birthday 1960-5-29 0:00:00
Internet
100%