第8章 ADO.NET 数据访问技术

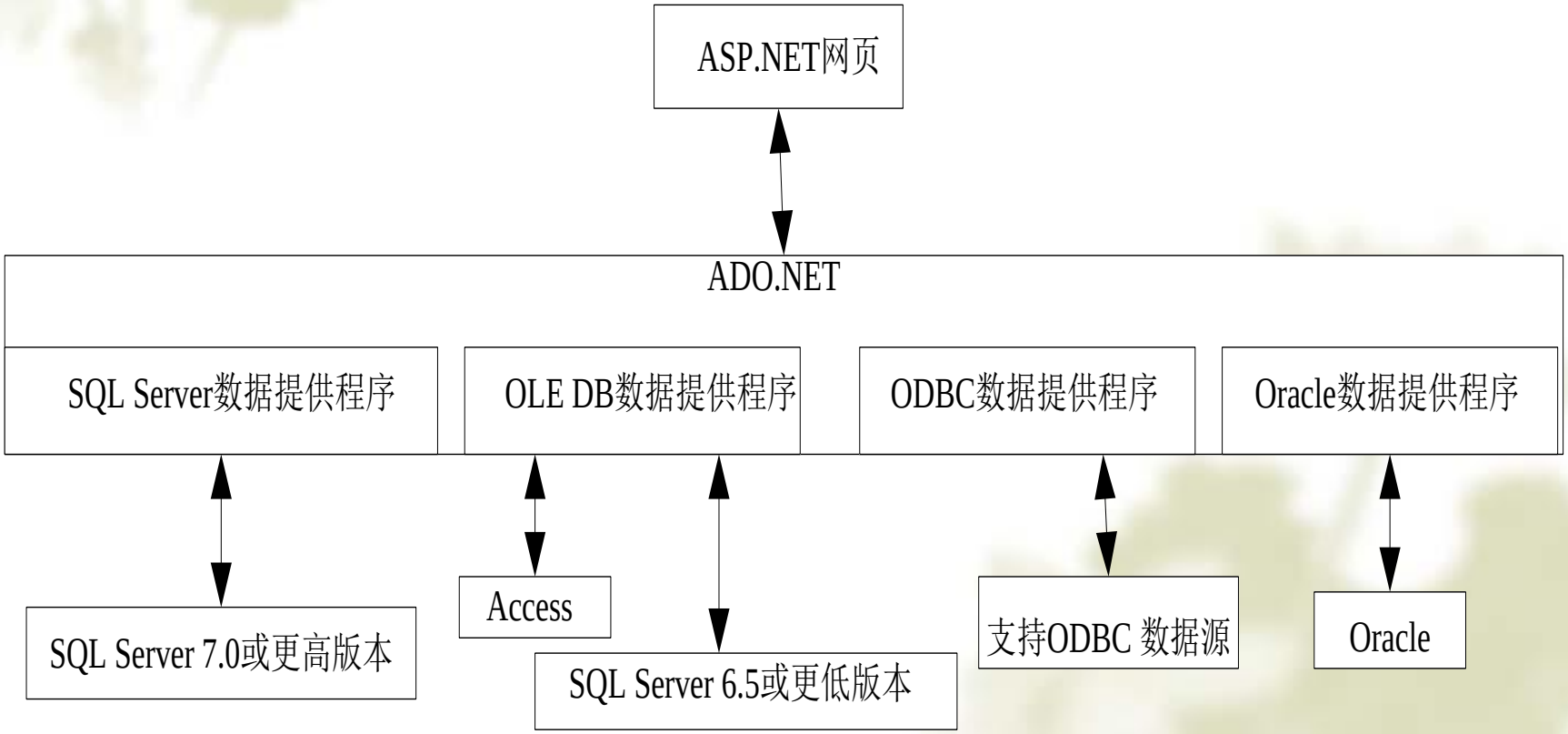
内容

- ADO.NET基础
- 连接模式数据库访问
- 断开模式数据库访问

8.1 ADO.NET基础

- ADO.NET模型
- ADO.NET的组件
- ADO.NET的数据访问模式

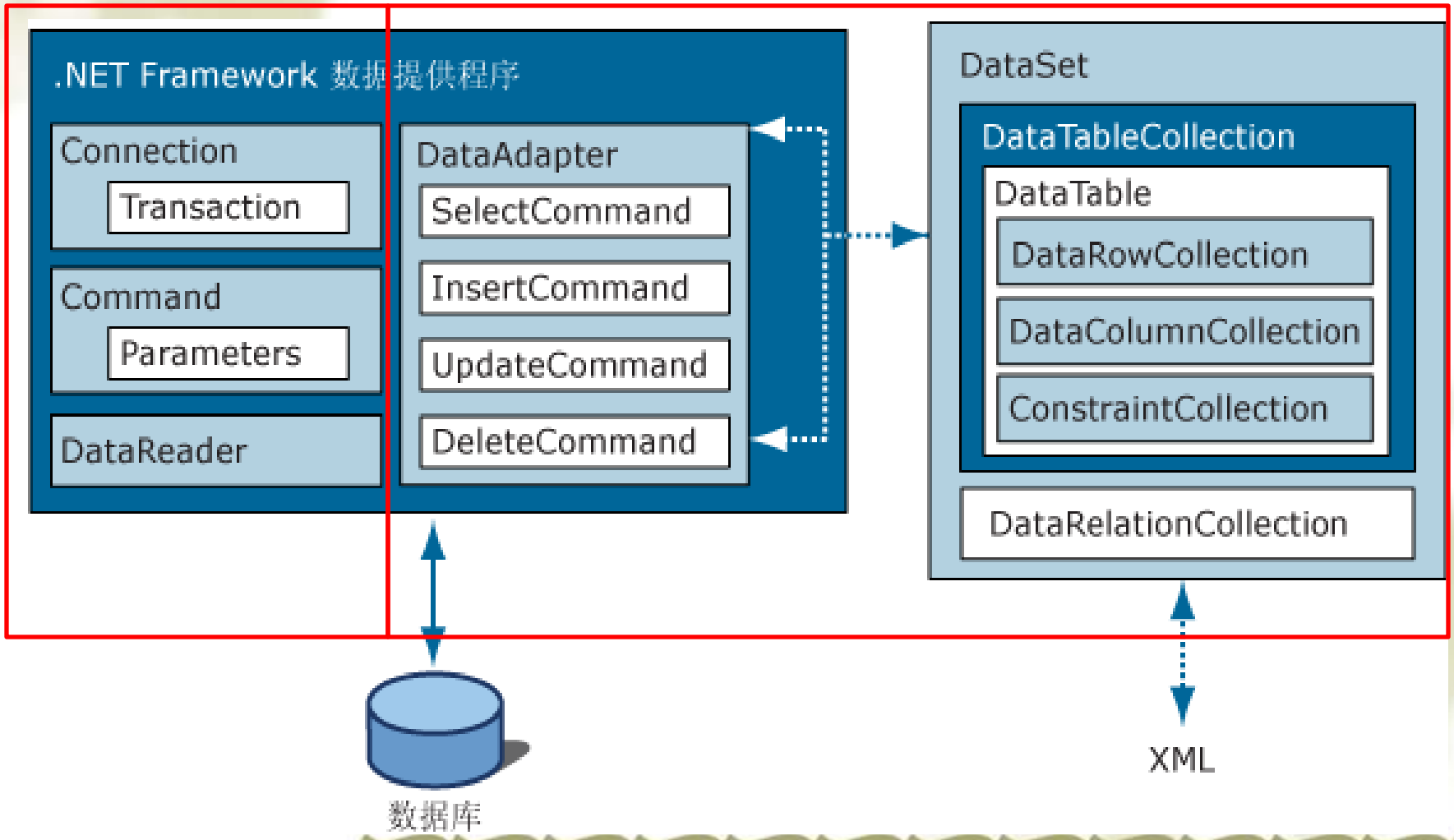
8.1.1 ADO.NET模型



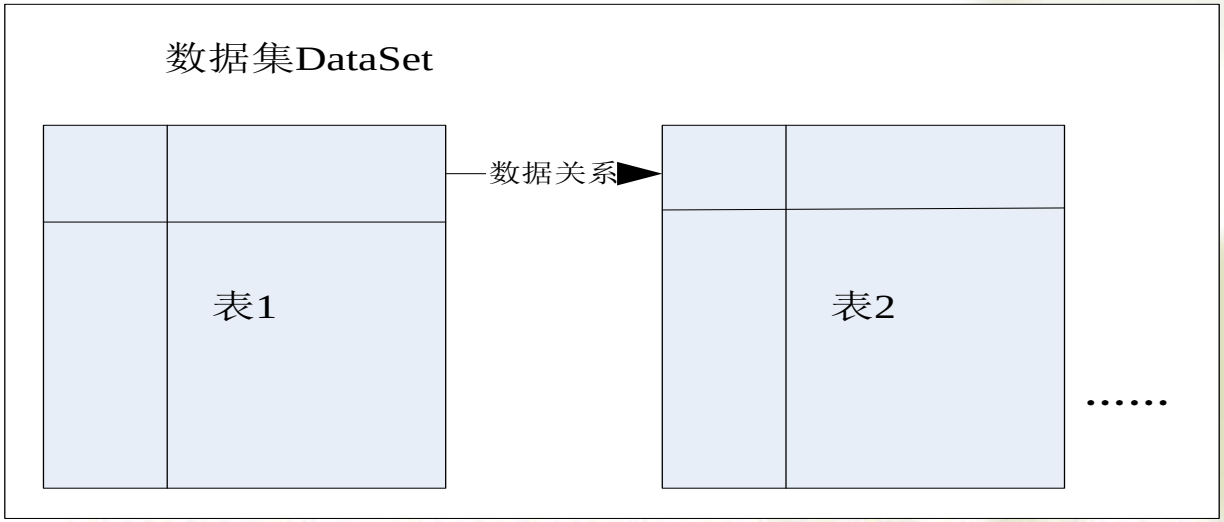
- 在ADO.NET中，连接数据源有4种数据提供程序。如果要在应用程序中使用任何一种数据提供程序，必须在后台代码中引用对应的命名空间，类的名称也随之变化。

数据访问提供程序	名称空间	对应的类名称
SQL Server数据提供程序	System.Data.SqlClient	SqlConnection；SqlCommand；SqlDataReader；SqlDataAdapter
OLE DB数据提供程序	System.Data.OleDb	OledbConnection；OledbCommand；OledbDataReader；OledbDataAdapter
ODBC数据提供程序	System.Data.Odbc	OdbcConnection；OdbcCommand；OdbcDataReader；OdbcDataAdapter
Oracle数据提供程序	System.Data.OracleClient	OracleConnection；OracleCommand；OracleDataReader；OracleDataAdapter

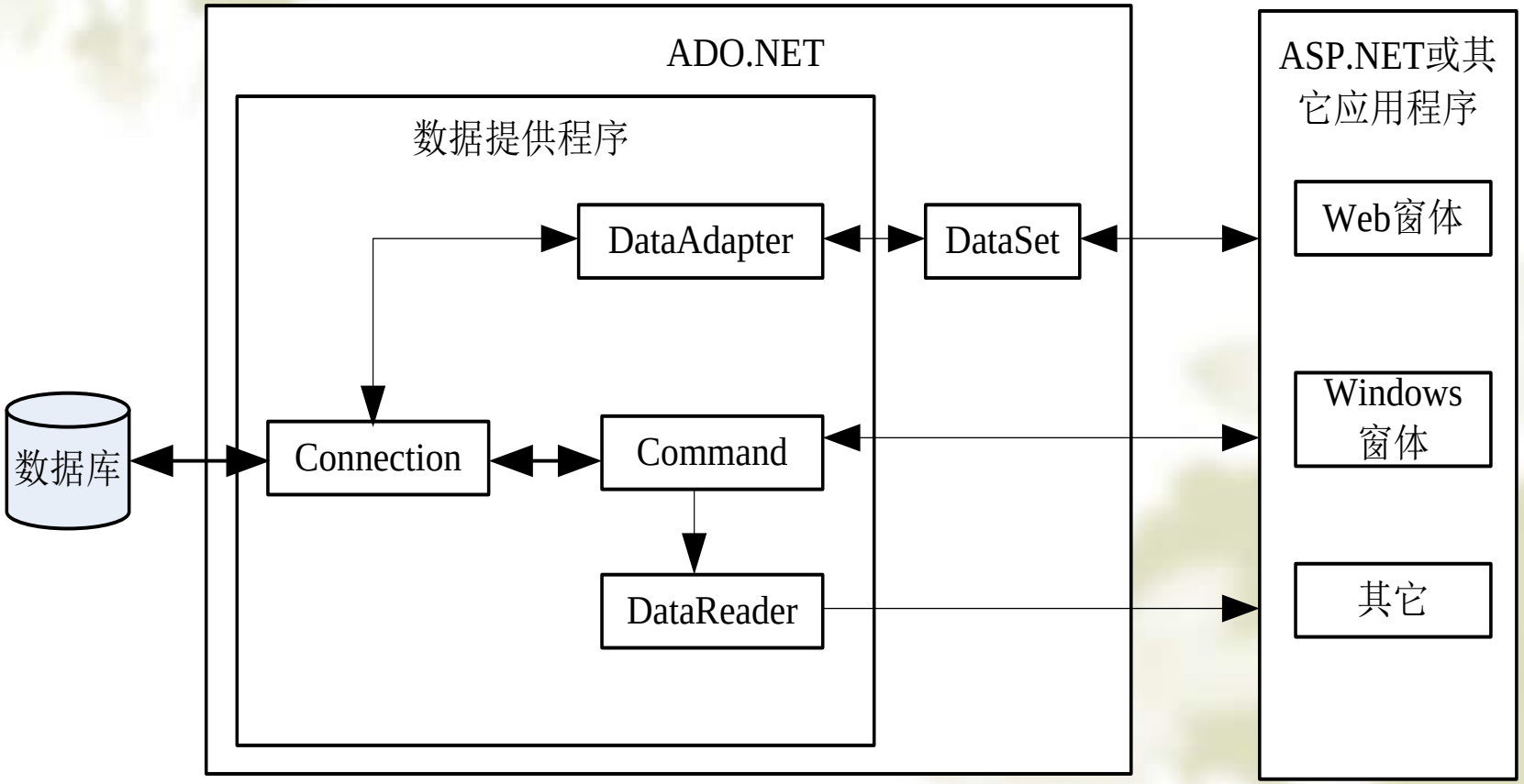
8.1.2 ADO.NET的组件



- 数据集**DataSet**位于**System.Data**命名空间下，用于在内存中暂存数据，可以把它看成是内存中的小型数据库。**DataSet**包含一个或多个数据表（**DataTable**），表数据可来自数据库、文件或**XML**数据。
- **DataSet**一旦读取到数据库中的数据后，就在内存中建立数据库的副本，在此之后的所有操作都是在内存中的**DataSet**中完成，直到执行更新命令为止。



8.1.3 ADO.NET的数据访问模式

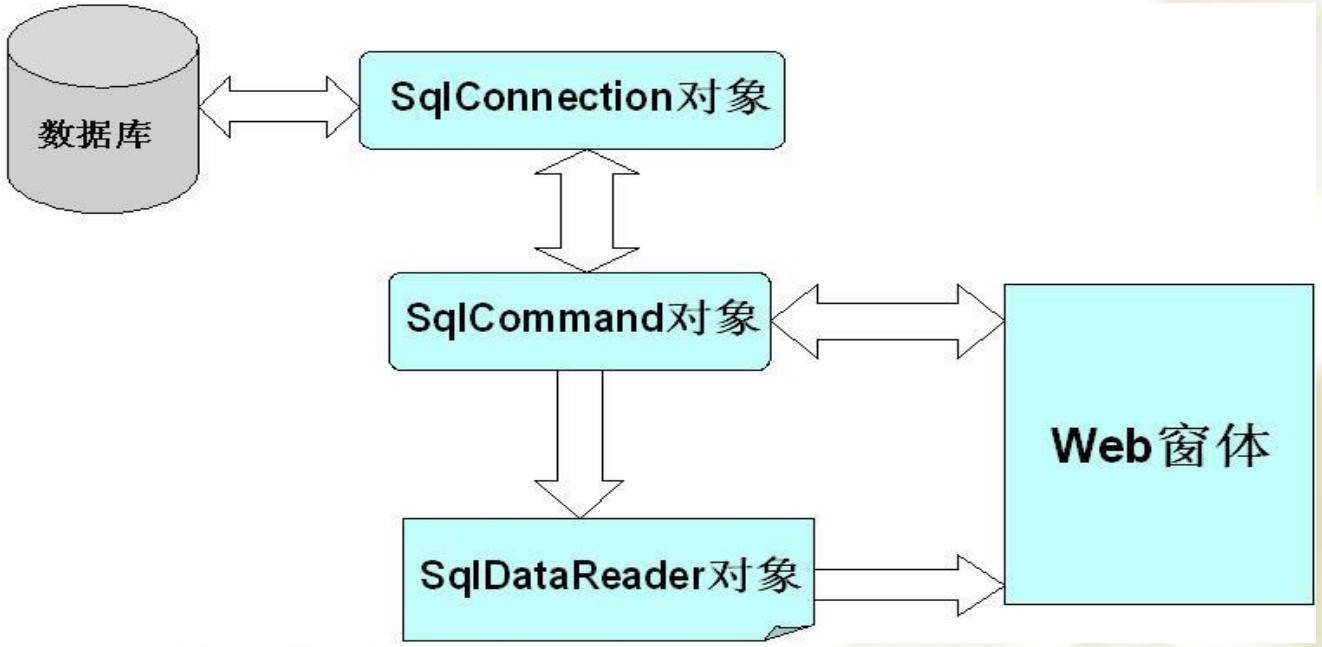


8.2 连接模式数据库访问

- 使用SqlConnection对象连接数据库
- 使用SqlCommand对象执行数据库命令
- 使用SqlDataReader读取数据
- 为SqlCommand传递参数
- 使用SqlCommand执行存储过程
- 使用事务处理

连接模式访问数据库的开发流程有以下几个步骤：

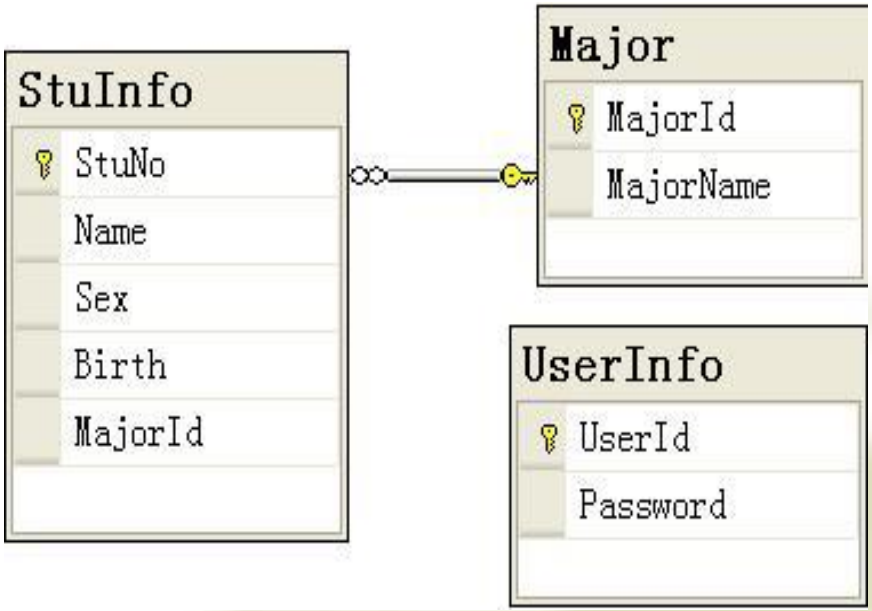
- **SqlConnection对象：连接数据库；**
- **SqlCommand对象：执行SQL命令，包括增、删、改及查询数据库等命令；**
- **SqlDataReader对象：查询数据库数据，并将查到的结果集绑定到控件上。**



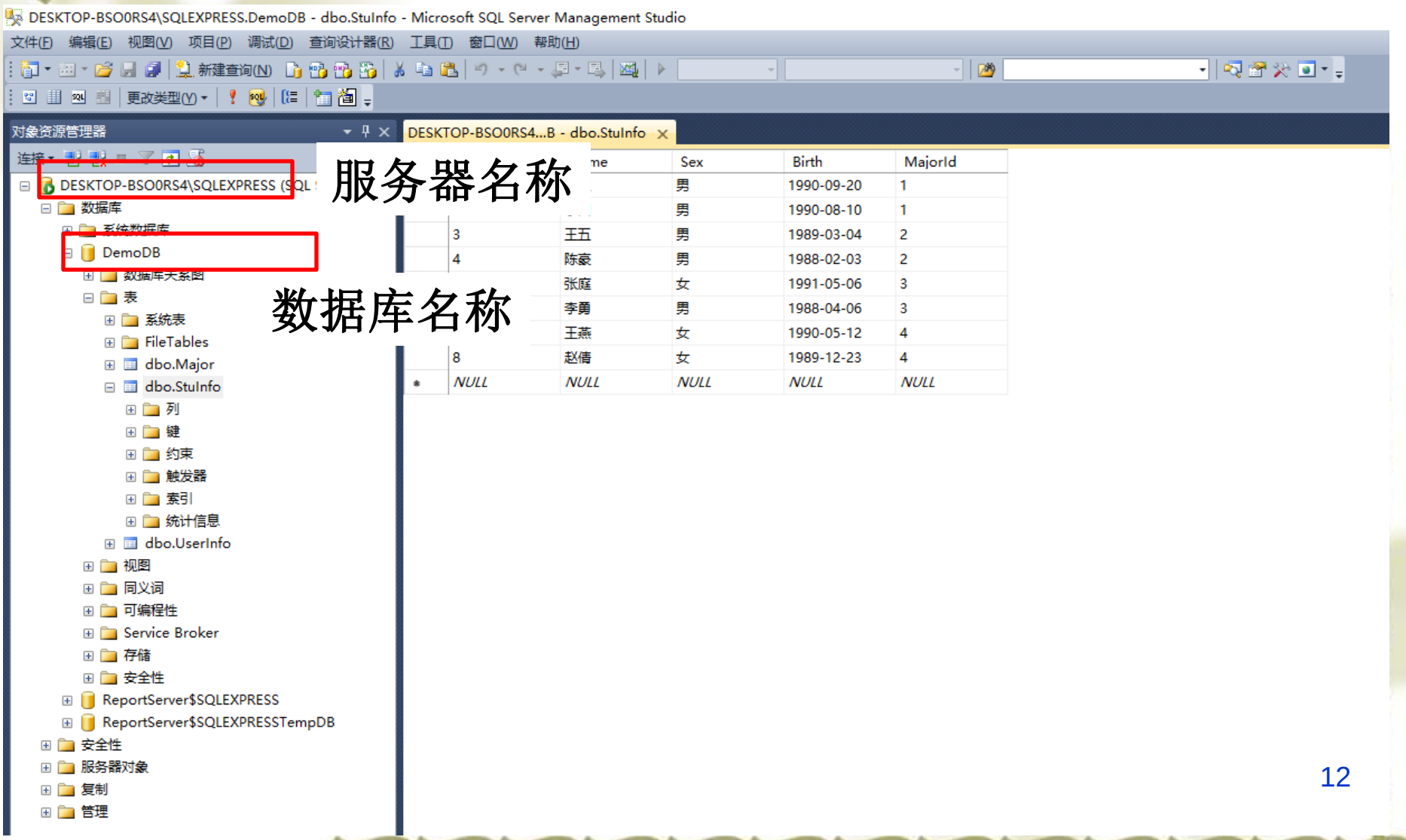
8.2.1 使用SqlConnection对象连接数据库

1. 示例数据库的创建

- 创建一个示例数据库**Student**，该数据库包含**StuInfo**、**Major**和**UserInfo**三张表，数据库表结构关系图。



SQL Server Management Studio



2．创建数据库连接

➢操作数据库的第一步是建立与数据库的连接，因此首先要创建SqlConnection对象。要创建SqlConnection对象必须先了解SqlConnection对象的常用属性和方法。

SqlConnection对象的常用属性

属 性	说 明
ConnectionString	取得和设置连接字符串
ConnectionTimeOut	获取SqlConnection对象的超时时间，单位为秒，0表示不限制。若在这个时间之内无法连接数据源，则产生异常
Database	获取当前数据库名称
DataSource	获取数据源的完整路径和文件名，若是SQL Server数据库则获取所连接的SQL Server服务器名称
State	获取数据库的连接状态，它的值ConnectionState枚举值

- **ConnectionString属性**通常包含以下参数，各参数间用“;”分隔。
  - **Provider**：用于设置数据源的OLE DB驱动程序。如：Access为“Microsoft.Jet.OLEDB.4.0”；SQL Server 6.5或之前版本为“SQLOLEDB”。
  - **Data Source**：设置数据源的实际路径。
  - **Password**：设置登录数据库所使用的密码。
  - **User ID**：设置登录数据库时所使用的帐号。

例如，连接Access数据库的的连接参数为：

```
Provider=Microsoft.Jet.OLEDB.4.0;Data
   Source=D:\\abc.mdb
```

- 对于SQL7.0或更高版本的SQL数据库，ConnectionString属性包含的主要参数有：

➢**Data Source或Server**：设置需连接的数据库服务器名称。

➢**Initial Catalog或Database**：设置连接的数据库名称。

➢**AttachDBFilename**：数据库的路径和文件名。

➢**User ID 或 uid**：登录SQL Server数据库的账户。

➢**Password 或 pwd**：登录SQL Server数据库的密码。

➢**Integrated Security**：是否使用Windows集成身份验证，值有三种：true、false和SSPI，true和SSPI表示使用Windows集成身份验证。

➢**Connection Timeout**：设置SqlConnection对象连接SQL数据库服务器的超时时间，单位为秒，若在所设置的时间内无法连接数据库，则返回失败信息。默认为15秒。

连接数据库时的验证模式

- 连接数据库时，有两种验证模式：
  - 混合验证模式
  - Windows集成验证模式

混合验证模式

- 混合验证模式：

  Data Source =localhost;
  Initial Catalog=northwind;
  User Id=sa;
  pwd=123

其中，**Data Source =localhost**表示连接本机**SQL**数据库的默认服务器；**northwind**为示例数据库。

Windows继承验证模式

- **Windows集成验证模式：**

  Data Source =localhost;
  Initial Catalog=northwind;
  Integrated Security=true

- 混合验证模式必须在连接字符串中以明文形式保存用户名和密码，因此安全性较差。**Windows**集成验证模式不发送用户名和密码；仅发送用户通过身份验证的信息。从安全角度考虑，建议使用**Windows**集成验证模式。

示例数据库连接参数

- 在本书的示例中，数据库都是放在网站的App_Data目录下。如例【8-1】中创建的Student数据库的连接参数应设置为：

  Data Source=.\SQLEXPRESS;
  AttachDbFilename=|DataDirectory|\Student.mdf;
  Integrated Security=True;User Instance=True

- 其中，Data Source=.\SQLEXPRESS表示SQLEXPRESS数据库服务器；AttachDbFilename表示数据库的路径和文件名；|DataDirectory|表示网站默认数据库路径App_Data。

SqlConnection对象的常用方法

方 法	说 明
Open( )	打开与数据库的连接
Close( )	关闭与数据库的连接
BeginTransaction( )	开始一个数据库事务，可以指定事务的名称和隔离级别
ChangeDatabase( )	在打开连接的状态下，更改当前数据库
CreateCommand( )	创建并返回与SqlConnection对象有关的SqlCommand对象
Dispose( )	调用Close( )方法关闭与数据库的连接，并释放所占用的系统资源

创建SqlConnection对象

- 在创建数据库连接对象时，需要指定连接字符串。通常有以下2种方法获取连接字符串：
  1. 创建连接对象，并在应用程序的中硬编码连接字符串。

```
SqlConnection 对象名称 = new SqlConnection("连接字符串");
```

或

```
SqlConnection 对象名称 = new SqlConnection( );
对象名称. ConnectionString="连接字符串";
```

2. 把连接字符串放在应用程序的web.config文件中，再引用web.config文件。

- 在web.config配置文件的<configuration>节中添加如下的代码。

```
<connectionStrings>
  <add name="StudentCnnString"
  connectionString="Data Source=.\SQLEXPRESS;
  AttachDbFilename=|DataDirectory|\Student.mdf;
  Integrated Security=True; User Instance=True"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```

- **web.config文件中有了连接字符串后，就可以从web.config中读取连接字符串。需要使用System.Configuration.ConfigurationManager类读取连接字符串。代码如下：**

```
string strCnn= ConfigurationManager.ConnectionStrings
["StudentCnnString"].ConnectionString;
//读取连接字符串
SqlConnection cnn = new SqlConnection(strCnn);
//定义连接对象
```

为了使上述代码正常工作，必须使用using System.Configuration语句引入命名空间。

打开SqlConnection连接对象

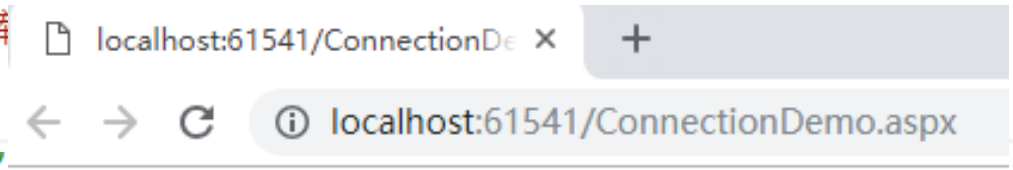
- 创建好**SqlConnection**连接对象后，并没有与数据库建立连接，要建立数据库连接，还必须使用**cnn.Open()**方法打开数据连接，然后才可以对数据库进行各种操作。操作完数据库后，一定要使用**cnn.Close()**方法关闭连接。

【例8-2】演示如何建立Student数据库的连接。

```
protected void Page_Load(object sender, EventArgs e)
{
    string strCnn = ConfigurationManager.ConnectionStrings["StudentCnnString"].ConnectionString;

    SqlConnection cnn = new SqlConnection(strCnn);
    cnn.Open();
    Label1.Text = "成功建立Sql Server数据库
    cnn.Close();
}
```

```
<connectionStrings>
  <!--非EXPRESS版本，修改User Instance的值为FALSE。
        Initial Catalog=|DataD
        Integrated Security=SS
        providerName="System.D
  <add name="StudentCnnString"
        connectionString="Data Source=.\SQLEXPRESS;
        Initial Catalog=testdb;
        Integrated Security=True;
        User Instance=False"
        providerName="System.Data.SqlClient"/>
</connectionStrings>
```



8.2.2 使用SqlCommand对象执行数据库命令

- 成功连接数据库后，接着就可以使用**SqlCommand**对象对数据库进行各种操作，如读取、写入、修改和删除等操作。

SqlCommand对象的常用属性

属 性	说 明
CommandText	获取或设置要对数据源执行的SQL命令、存储过程或数据表名称
CommandType	获取或设置命令类型，可取的值：CommandType.Text、CommandType.StoredProduce或CommandType.TableDirect，分别对应SQL命令、存储过程或数据表名称，默认为Text。
Connection	获取或设置SqlCommand对象所使用的数据连接属性
Parameters	SQL命令参数集合
Transaction	设置Command对象所属的事务

建立SqlCommand对象的方法有4种：

- SqlCommand 对象名 = new SqlCommand();
- SqlCommand 对象名 = new SqlCommand("SQL命令");
- SqlCommand 对象名= new SqlCommand("SQL命令" , 连接对象);
- SqlCommand 对象名= new SqlCommand("SQL命令" , 连接对象 , 事务对象);

SqlCommand对象的常用方法

方 法	说 明
Cancel	取消SqlCommand对象的执行
CreateParameter	创建Parameter对象
ExecuteNonQuery 	执行CommandText属性指定的内容，返回数据表被影响的行数。该方法只能执行Insert、Update和Delete命令
ExecuteReader 	执行CommandText属性指定的内容，返回DataReader对象。该方法用于执行返回多条记录的Select命令
ExecuteScalar 	执行CommandText属性指定的内容，以object类型返回结果表第一行第一列的值。该方法一般用来执行查询单值的Select命令
ExecuteXmlReader	执行CommandText属性指定的内容，返回XmlReader对象。该方法以XML文档格式返回结果集

1.ExecuteNonQuery方法

ExecuteNonQuery方法只能执行Insert、Update和Delete命令，因此可以增加、修改和删除数据库中的数据。增加、修改和删除数据库中的数据的步骤相同，具体描述如下：

① 创建SqlConnection对象，设置连接字符串；

② 创建SqlCommand对象，设置它的Connection和CommandText属性，分别表示数据库连接和需要执行的SQL命令。

③ 打开与数据库连接；

④ 使用SqlCommand对象的ExecuteNonQuery方法执行CommandText中的命令；并根据返回值判断是否对数据库操作成功。

⑤ 关闭与数据库连接；

【例8-3】

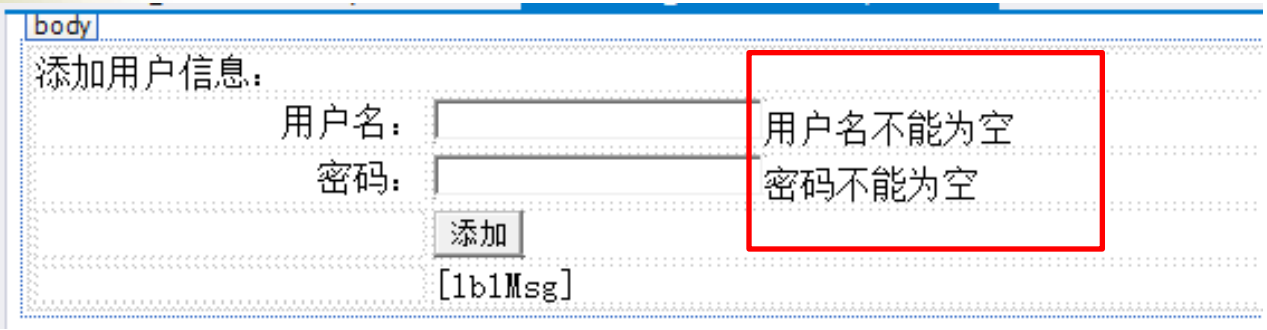
演示如何使用EXECUTENONQUERY方法增加数据库中USERINFO表的用户信息。

【例8-3】第1步 web.config设置连接数据库

```
<connectionStrings>
  <!--非EXPRESS版本，修改User Instance的值为FALSE。
  解决网页运行时出现的问题：
  “此版本的 SQL Server 不支持用户实例登录标志。该连接将关闭。”-->
  <!--修改AttachDbFilename属性为InitialCatalog，
  解决问题：
  已存在同名的数据库，或指定的文件无法打开或位于 UNC 共享目录中
  <add name="StudentCnnString"
       connectionString="Data Source=.\SQLEXPRESS;
       Initial Catalog=|DataDirectory|\Student.mdf;
       Integrat
       providerName="System.Data.SqlClient"/>-->
  <add name="StudentCnnString"
       connectionString="Data Source=.\SQLEXPRESS;
       Initial Catalog=testdb;
       Integrated Security=True;
       User Instance=False"
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

设置服务器名称、数据库名称、登陆方式

【例8-3】第2步 添加Web窗体及其控件




```
<tr>
    <td style="text-align:right">用户名：</td>
    <td>
        <asp:TextBox ID="txtName" runat="server">
        </asp:TextBox>
        <asp:RequiredFieldValidator
            ID="RequiredFieldValidator1"
            runat="server"
            ErrorMessage="用户名不能为空"
            ControlToValidate="txtName">
        </asp:RequiredFieldValidator>
    </td>
</tr>
```

验证控件

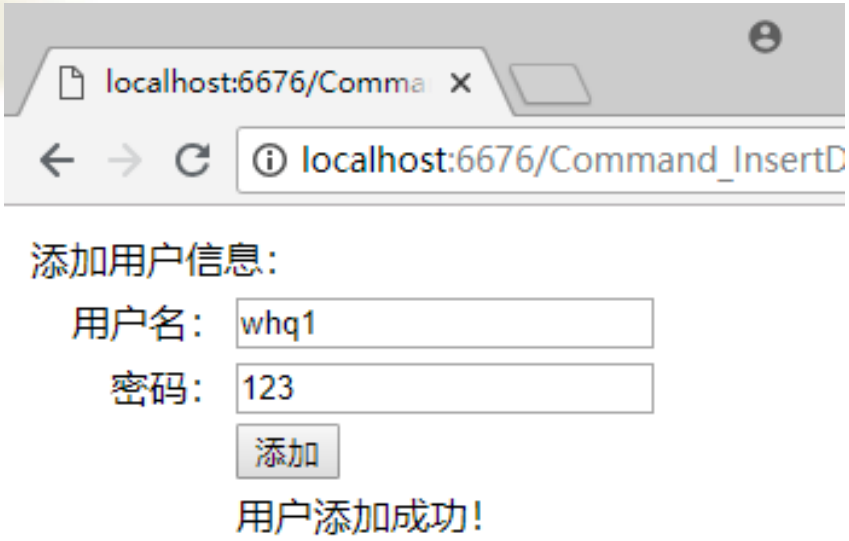
修改web.config中Framework版本为4.0

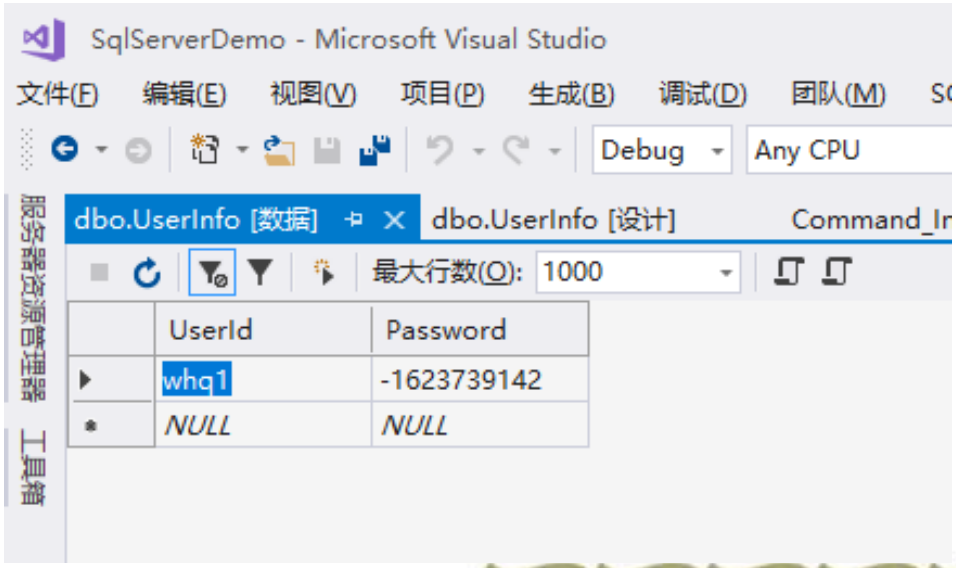
【例8-3】第3步 按钮事件操作数据库

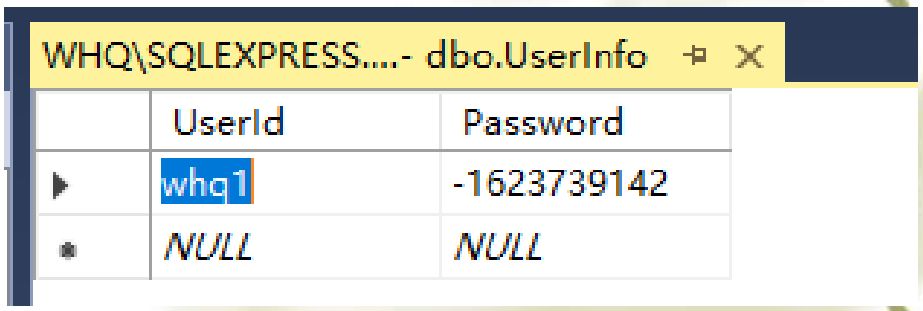
```
if (this.IsValid) //页面验证通过
{
    string strCnn = ConfigurationManager.ConnectionStrings["StudentCnnString"].ConnectionString;
    SqlConnection cnn = new SqlConnection(strCnn);
    SqlCommand cmd = new SqlCommand();//建立命令对象
    cmd.Connection = cnn;//设置命令对象的数据连接属性
                         //把SQL语句赋给命令对象
    cmd.CommandText = "insert into UserInfo(UserId,Password) values('"
        + txtName.Text.Trim() + "','" + txtPassword.Text.Trim().GetHashCode() + "')";
    try
    {
        cnn.Open();//打开连接
        cmd.ExecuteNonQuery();//执行SQL命令
        lblMsg.Text = "用户添加成功！";
    }
    catch (Exception ex)
    {
        lblMsg.Text = "用户添加失败，错误
    }
    finally
    {
        if (cnn.State == ConnectionState.Open)
            cnn.Close();
    }
}
```

Open函数打开数据库连接；
ExecuteNonQuery执行Insert操作

【例8-3】第4步 运行验证


localhost:6676/Comma
localhost:6676/Command_InsertD
添加用户信息:
用户名:
whq1
密码:
123
添加
用户添加成功!

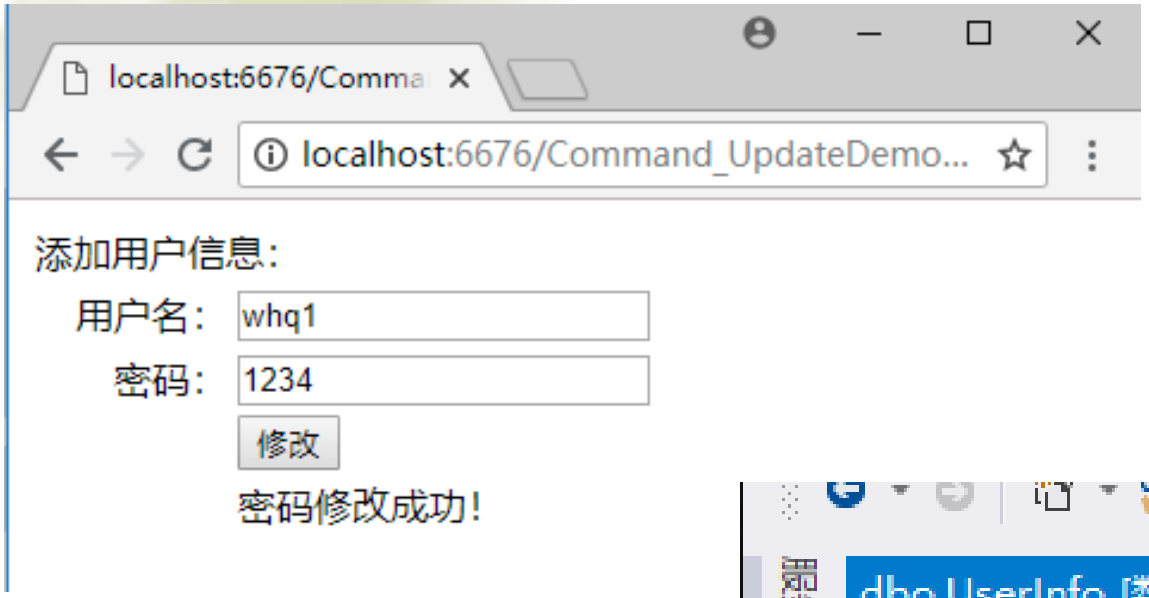

SqlServerDemo - Microsoft Visual Studio
文件(F) 编辑(E) 视图(V) 项目(P) 生成(B) 调试(D) 团队(M)
Debug
Any CPU
dbo.UserInfo [数据]
dbo.UserInfo [设计]
Command_Ir
最大行数(O): 1000
UserId
Password
whq1
-1623739142
NULL
NULL
服务器资源管理器
工具箱

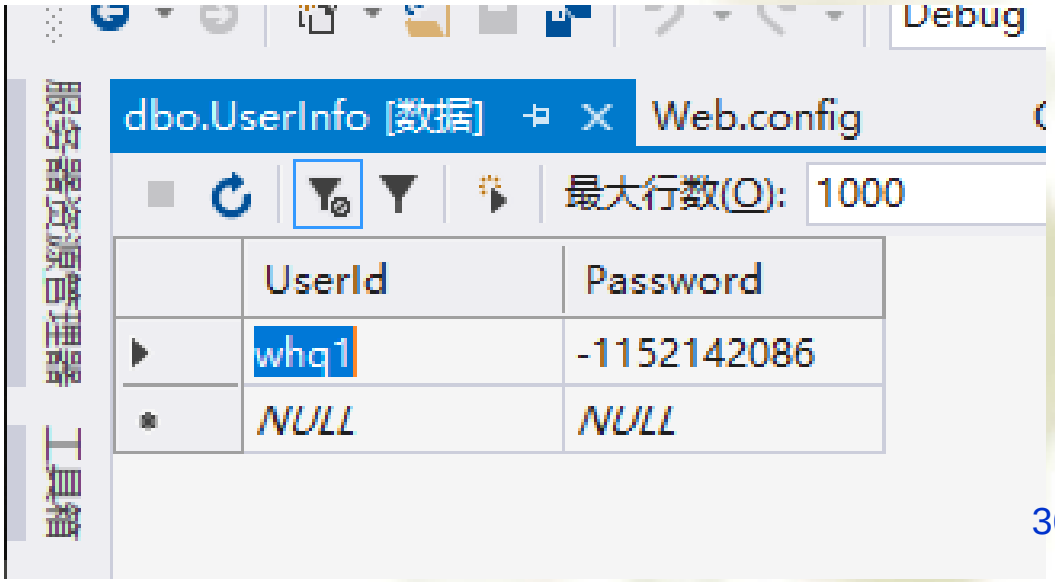

WHQ\SQLEXPRESS....- dbo.UserInfo
UserId
Password
whq1
-1623739142
NULL
NULL

【例8-4】演示如何使用ExecuteNonQuery方法修改数据库中UserInfo表的用户信息。

```
string strCnn = ConfigurationManager.ConnectionStrings["StudentCnnString"]
SqlConnection cnn = new SqlConnection(strCnn);
SqlCommand cmd = new SqlCommand();//建立命令对象
cmd.Connection = cnn;//设置命令对象的数据连接属性
                     //把SQL语句赋给命令对象
cmd.CommandText = "update UserInfo set Password='"
    + txtPassword.Text.Trim().GetHashCode()
    + "' where UserId='" + txtName.Text.Trim() + "'";
try
{
    cnn.Open();//打开连接
    int updateCount = cmd.ExecuteNonQuery();//执行SQL命令
    if (updateCount == 1)
        lblMsg.Text = "密码修改成功！";
    else
        lblMsg.Text = "该用户记录不存在！";
}
```

【例8-4】运行验证


localhost:6676/Comma
localhost:6676/Command_UpdateDemo...
添加用户信息:
用户名: whq1
密码: 1234
修改
密码修改成功!


Debug
dbo.UserInfo [数据]
Web.config
最大行数(O): 1000
UserId
Password
whq1
-1152142086
NULL
NULL
服务器资源管理器
工具箱

【例8-5】演示如何使用ExecuteNonQuery方法删除数据库中UserInfo表的用户信息。

```
cmd.CommandText = "delete UserInfo  where UserId='" + txtName.Text.Trim() + "'";
try
{
    cnn.Open();//打开连接
    int deleteCount = cmd.ExecuteNonQuery();//执行SQL命令
    if (deleteCount == 1)
        lblMsg.Text = "用户删除成功！ ";
    else
        lblMsg.Text = "该用户记录不存在！ ";
}
```

2. ExecuteScalar方法

- **ExecuteScalar方法一般用来执行查询单值的Select命令，它以object类型返回结果表第一行第一列的值。**

【例8-6】演示如何使用ExecuteScalar方法查询数据库中StuInfo表的学生人数。

```
protected void Page_Load(object sender, Eve
{
    string strCnn = ConfigurationManager.Co                                    onString;
    SqlConnection cnn = new SqlConnection(s
    SqlCommand cmd = new SqlCommand();//建
    cmd.Connection = cnn;//设置命令对象的数
                         //把SQL语句赋给命令对象
    cmd.CommandText = "select count(*) from stuInfo";
    try
    {
        cnn.Open();//打开连接
        object count = cmd.ExecuteScalar();//执行SQL命令,返回Object类型的数据
        lblMsg.Text =
    }
    catch (Exception
    {
        lblMsg.Text =
    }
    finally
    {
        if (cnn.State
            cnn.Close
    }
}
```

思考：若select语句修改为：
Select * from stuInfo
执行结果？

object SqlCommand.ExecuteScalar()

执行查询，并返回由查询返回的结果集中的第一行的第一列。 其他列或行将被忽略。

异常:

- InvalidCastException
- SqlException
- InvalidOperationException
- System.IO.IOException
- ObjectDisposedException

3. ExecuteReader方法

- 读取数据时需要与数据源保持实时连接，以循环的方式读取结果集中的数据。
- 调用**SqlCommand**对象的**ExecuteReader**方法获取查询结果，查询结果为**SqlDataReader**对象。
- **SqlDataReader**对象一旦创建，即可通过对象的属性、方法访问数据源中的数据。

执行ExecuteReader方法

```
string strCnn = ConfigurationManager.ConnectionStrings["StudentCnnString"].Con
SqlConnection cnn = new SqlConnection(strCnn);
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnn;
cmd.CommandText = "select * from StuInfo";
cnn.Open();
//创建DataReader对象,执行SQL命令，并获取查询结果
SqlDataReader stuReader = cmd.ExecuteReader();
```

SqlDataReader对象

- **SqlDataReader对象的常用属性：**
  - **FieldCount：获取由SqlDataReader得到的一行数据中的字段数。**
  - **isClosed：获取SqlDataReader对象的状态。true表示关闭，false表示打开。**
  - **HasRows：表示SqlDataReader是否包含数据。**

SqlDataReader对象的常用方法(1)

- **Close()方法：**不带参数，无返回值，用来关闭SqlDataReader对象。
- **Read()方法：**让记录指针指向本结果集中的下一条记录，返回值是true或false。
- **NextResult()方法：**当返回多个结果集时，使用该方法让记录指针指向下一个结果集。当调用该方法获得下一个结果集后，依然要用Read方法来遍历访问该结果集。

SqlDataReader对象的常用方法(2)

- **GetValue(int i)方法**：根据传入的列的索引值，返回当前记录行里指定列的值。由于事先无法预知返回列的数据类型，所以该方法使用**Object**类型来接收返回数据。
- **GetName(int i)方法**：通过输入列索引，获得该列的名称。综合使用**GetName**和**GetValue**两方法，可以获得数据表里列名和列的字段。

【例8-7】演示如何使用SqlDataReader对象读取StuInfo表的记录。

GETName：
获取列名

GETValue：
获取列的值

```
cmd.CommandText = "select * from StuInfo";
SqlDataReader stuReader = null;//创建DataReader对象的引用
try
{
    if (cnn.State == ConnectionState.Closed)
        cnn.Open();
    //执行SQL命令，并获取查询结果
    stuReader = cmd.ExecuteReader();
    //依次读取查询结果的字段名称，并以表格的形式显示
    Response.Write("<table border='1'><tr align='center'>");
    for (int i = 0; i < stuReader.FieldCount; i++)
    {
        Response.Write("<td>" + stuReader.GetName(i) + "</td>");
    }
    Response.Write("</tr>");
    //如果DataRead对象成功获得数据，返回true，否则返回false
    while (stuReader.Read())
    {
        //依次读取查询结果的字段值，并以表格的形式显示
        Response.Write("<tr>");
        for (int j = 0; j < stuReader.FieldCount; j++)
        {
            Response.Write("<td>" + stuReader.GetValue(j) + "</td>");
        }
        Response.Write("</tr>");
    }
    Response.Write("</table>");
```

DataReader.aspx.cs

StuNo	Name	Sex	Birth	MajorId
1	张三	男	1990/09/20 0:00:00	1
2	李四	男	1990/08/10 0:00:00	1
3	王五	男	1989/03/04 0:00:00	2
4	陈豪	男	1988/02/03 0:00:00	2
5	张庭	女	1991/05/06 0:00:00	3

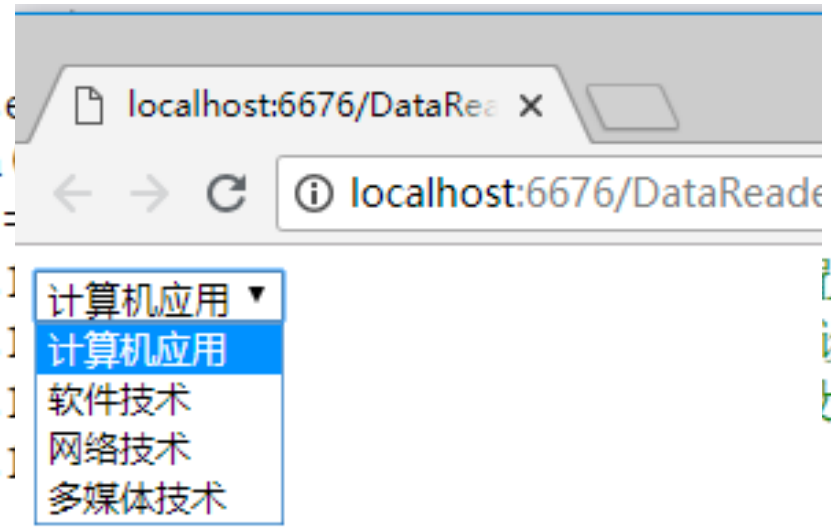
Visual Studio 2010提供了大量列表绑定控件，如DropDownList、ListBox和GridView控件等，可以直接将SqlDataReader对象绑定到这些控件来显示查询结果。

与控件绑定时，主要设置控件的以下属性和方法：

- **DataSource属性：**设置控件的数据源，可以是SqlDataReader对象，也可以是DataSet对象。
- **DataTextField属性：**对于绑定DropDownList、ListBox等控件时，设置显示数据的字段名称。
- **DataValueField属性：**对于绑定DropDownList、ListBox等控件时，设置隐藏值的字段名称。
- **DataBind方法：**设置完控件的绑定属性后，调用该方法将数据绑定到控件上。

【例8-8】演示如何将SqlDataReder对象与DropDownList控件绑定。本示例主要在DropDownList控件中显示Major表的记录。

```
cmd.CommandText = "select * from Major";
SqlDataReader MajorReader = null;
try
{
    if (cnn.State
        cnn.Open
    MajorReader =
    DropDownList1
    DropDownList1
    DropDownList1
    DropDownList1
}
```



DataReader_DataBind.aspx.cs

- **SqlDataReader对象查询数据库的一般步骤：**
  1. 创建SqlConnection对象，设置连接字符串；
  2. 创建SqlCommand对象，设置它的Connection和CommandText属性，分别表示数据库连接和需要执行的SQL命令。
  3. 打开与数据库连接；
  4. 使用SqlCommand对象的ExecuteReader方法执行CommandText中的命令；并把返回的结果放在SqlDataReader对象中。
  5. 通过循环，处理数据库查询结果。
  6. 关闭与数据库连接；

- 使用SqlDataReader对象时，应注意以下几点：
  - ① 读取数据时，SqlConnection对象必须处于打开状态。
  - ② 必须通过SqlCommand对象的ExecuteReader()方法，产生SqlDataReader对象的实例。
  - ③ 只能按向下的顺序逐条读取记录，不能随机读取。且无法直接获知读取记录的总数。
  - ④ SqlDataReader对象管理的查询结果是只读的，不能修改。