

第一部分 C#数据库编程

朱运乔



第一讲 ADO.NET

一、主要内容：

- 1、ADO.NET对象模型概述
- 2、主要数据对象应用举例

二、目的及要求

- 1、掌握ADO.NET结构组及其关系
- 2、掌握数据提供程序及各部分作用
- 3、初步掌握Connection对象的配置及使用
- 4、掌握Command对象常用方法成员及应用
- 5、掌握DataReader对象常用方法成员及应用



ADO.NET

ADO.NET是 .NET提供的、为访问各种数据源提供统一接口和方法的类。

ADO.NET包括两大部分：数据提供程序和数据集（DataSet）。

数据提供程序负责与物理数据库的连接，数据集代表的数据。

1、.NET数据提供程序

根据将要访问的数据库类型，.NET框架提供了不同的数据提供程序，常用的如：

- （1）SQL Server .NET数据提供程序 用以访问SQL Server数据库
- （2）OLE DB .NET数据提供程序 用以访问任何与OLE DB兼容的数据库

每个数据提供程序都实现了以下的类，构成了提供程序的核心对象：

| | |
|-------------|------------------|
| Connection | 建立对物理数据库的连接 |
| Command | 用于执行数据库操作命令 |
| DataReader | 用于访问一个只读、向前的数据流 |
| DataAdapter | 用以负责数据集同物理数据源的通信 |

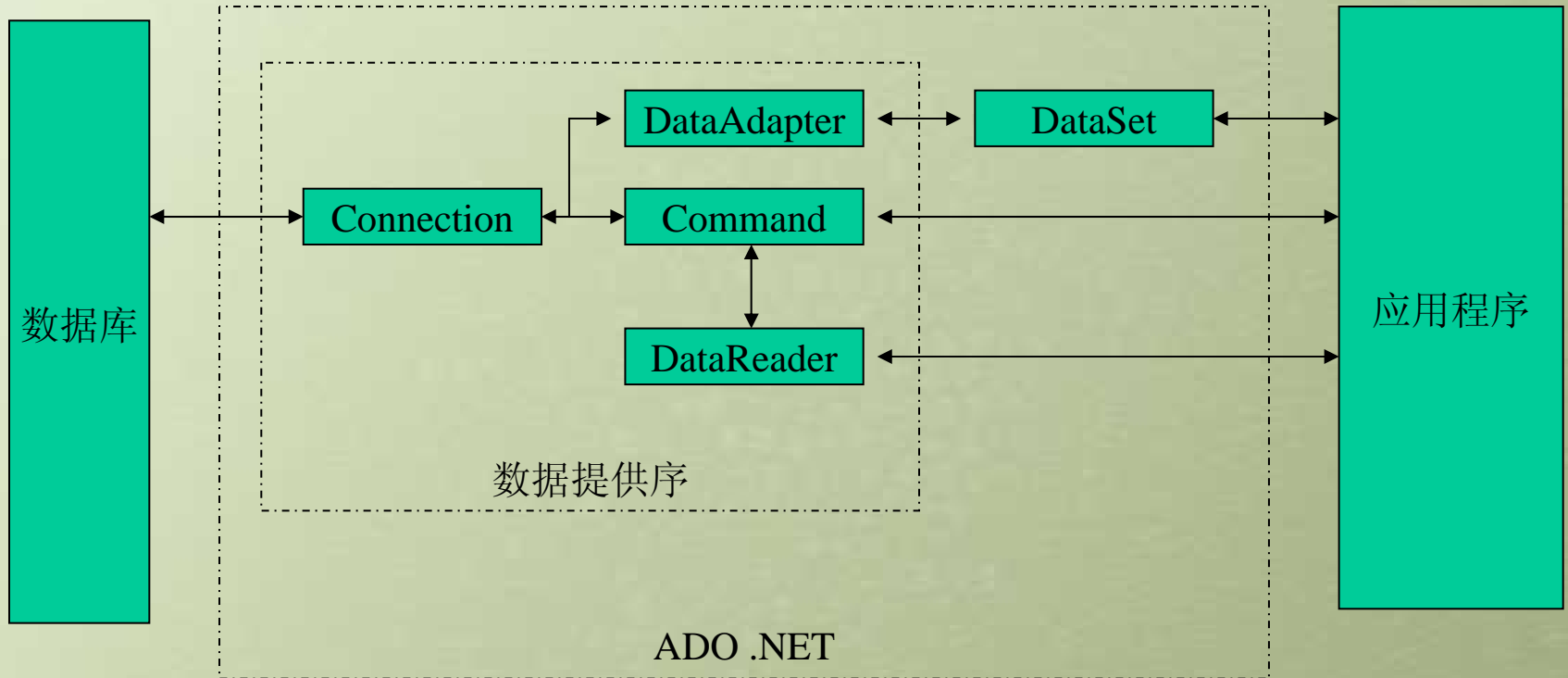
不同数据提供程序的核心对象命名不同，但内容几乎一样：

| 核心对象 | OLE DB .NET | SQL Server .NET |
|-------------|------------------|-----------------|
| Connection | OleDbConnection | SqlConnection |
| Command | OleDbCommand | SqlCommand |
| DataReader | OleDbDataReader | SqlDataReader |
| DataAdapter | OleDbDataAdapter | SqlDataAdapter |



ADO.NET

ADO .NET对象模型如下图所示：



另外，不同的数据提供程序所在的名字空间也不一样，如：

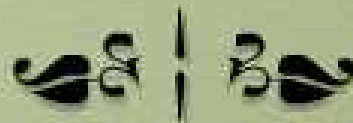
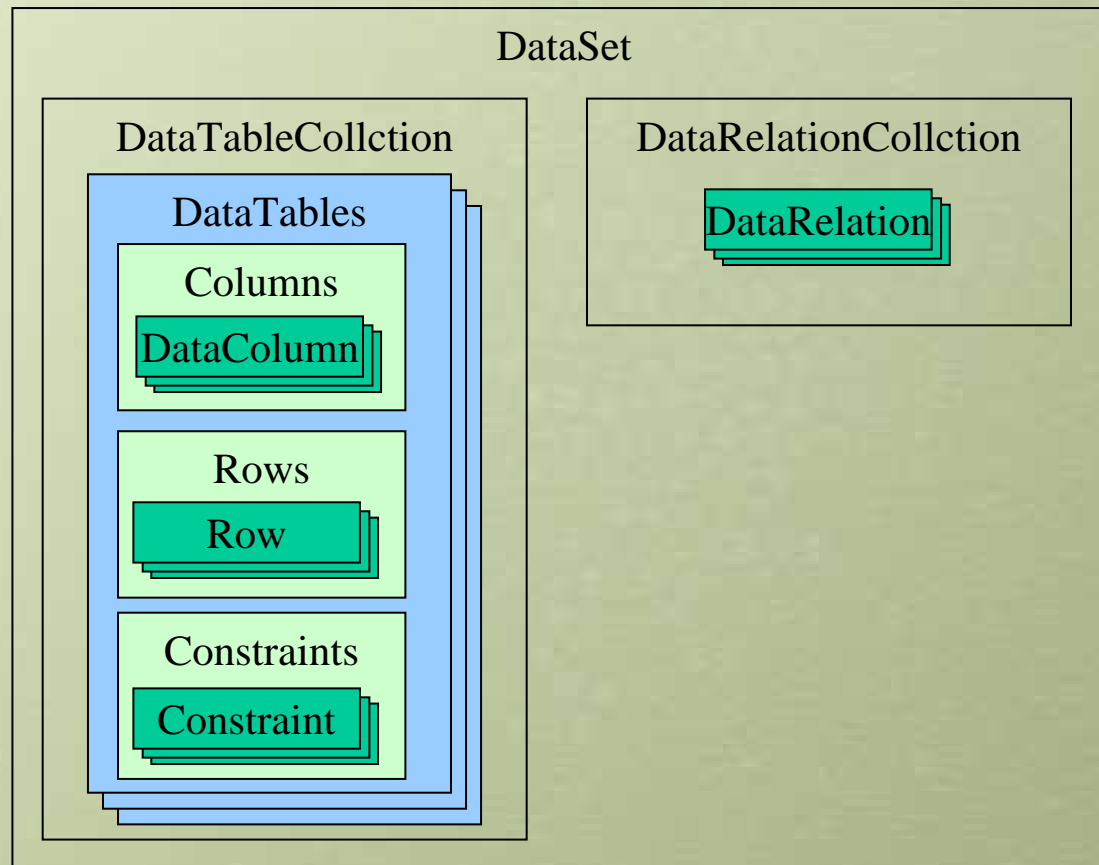
OLE DB .NET System.Data.OleDb

SQL Server .NET System.Data.SqlClient



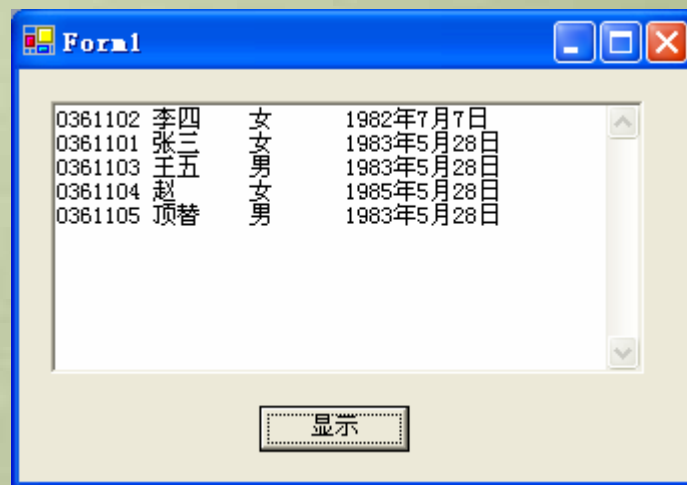
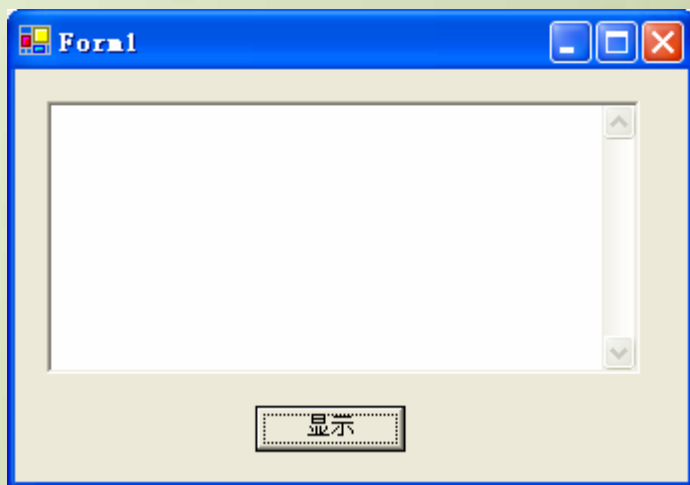
ADO.NET

数据集（DataSet）是记录在内存中的数据，类似一个简化的关系数据库，包含表及表这间的关系。



ADO.NET组件对象的使用

[项目一]利用DataReader对象读取数据，本例为一ADO.NET对象的简单应用：通过.NET提供的ADO.NET数据控件提供连接并读取数据记录。程序主界面如下图一，单击显示后如下图二。



1、制作过程

1) 生成Access数据库stu.mdb，并添加一个表tb1，表模式为：

tb1(xh(文本，10)，xm(文本，4)，xb(文本，1)，csrq(日期/时间))

主键：xh

对应表在Access中的设计界面为下图三；生成表tb1结构后并输入记录，见下图四。



ADO.NET组件对象的使用

tb1 : 表

| 字段名称 | 数据类型 | 说明 |
|------|-------|----|
| xh | 文本 | |
| xm | 文本 | |
| xb | 文本 | |
| csrq | 日期/时间 | |

字段属性

常规 查阅



| | |
|---------------|--------|
| 字段大小 | 10 |
| 格式 | |
| 输入掩码 | |
| 标题 | |
| 默认值 | |
| 有效性规则 | |
| 有效性文本 | |
| 必填字段 | 否 |
| 允许空字符串 | 是 |
| 索引 | 有(无重复) |
| Unicode 压缩 | 是 |
| 输入法模式 | 开启 |
| IME 语句模式(仅日文) | 无转化 |
| 智能标记 | |

tb1 : 表

| xh | xm | xb | csrq |
|---------|----|----|-----------|
| 0361101 | 张三 | 女 | 1983-5-28 |
| 0361102 | 李四 | 女 | 1982-7-7 |
| 0361103 | 王五 | 男 | 1983-5-28 |
| 0361104 | 赵六 | 女 | 1985-5-28 |
| 0361105 | 顶替 | 男 | 1983-5-28 |

记录: 1 共有记录数: 5

2) 生成应用程序主界面，添加以下控件：

- 一个文本框textBox1和一个按钮button1
- 在工具箱的数据分类标签中找到连接控件  OleDbConnection 并添加到窗体
- 在工具箱的数据分类标签中找到命令控件  OleDbCommand 并添加到窗体

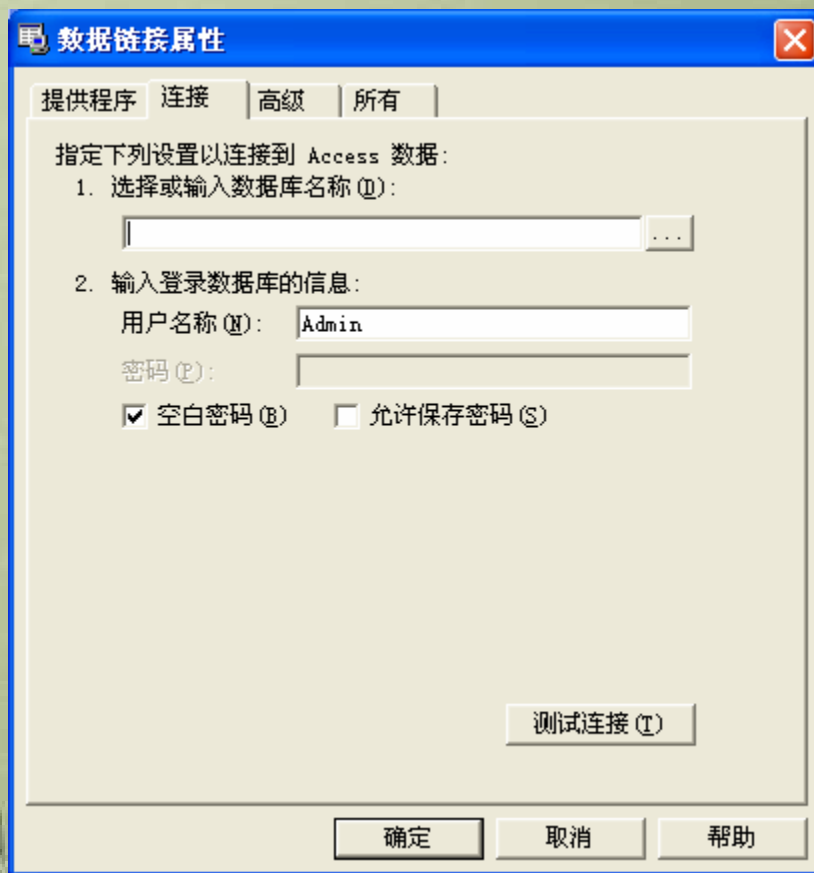


ADO.NET组件对象的使用

3) 属性修改

(1) 连接控件oleDbConnection1的ConnectionString设置:

点击属性右边的设置按钮，出现下图五；选择第一项后单下一步，出现图六，然指定位置选择刚建好的数据库名stu.mdb，然后测试连接，如显示成功则表示正确。



ADO.NET组件对象的使用

(2) 命令数据控件oleDbCommand1的属性设置:

Connection属性: 选择上面配置好的oleDbConnection1。

CommandText属性设置, 点击右边的设置按钮, 出现下图八: 按要求点击右键并添加表tb1, 出现下图九, 并在其中勾选“所有列”。即可生成所需的SQL命令。



ADO.NET组件对象的使用

4) 代码实现

(1) 本例中所操作数据库为access数据库，故所用的数据提供程序为OLE DB .NET（本例中也可用ODBC .NET），所以窗体代码前应导入相应的名字空间：

```
using System.Data.OleDb;
```

(2) 添加显示按钮Click事件处理方法：

```
private void button1_Click(object sender, System.EventArgs e)
{
    string str="";           //声明临时字符串
    OleDbConnection1.Open(); //打开数据库连接
    OleDbDataReader rd =OleDbCommand1.ExecuteReader();//执行SQL命令，返只读数据集
    while(rd.Read())
    {
        //只读数据集的Read()方法功能为下移一行并读取数据行
        str+=rd.GetString(0) +"\t";           //读该行第一列字符串
        str+=rd.GetString(1) +"\t";           //读取第二列数据
        str+=rd.GetString(2) +"\t";           //读取第三列数据
        str+=rd.GetDateTime(3).ToLongDateString(); //读取第四列日期时间型数据
        str+= "\r\n";           //回车换行
    }
    rd.Close();           //关闭只读、向前数据集
    OleDbConnection1.Close(); //关闭连接
    textBox1.Text=str;
}
```



ADO.NET组件对象的使用

2、相关知识

1)OleDbConnection连接对象

该对象为OLE DB .NET数据提供程序的核心对象之一。

其最重的属性：

ConnectionString 主要用以指定连接数据源，设置方法有两种：

- ✓如果连接对象是通过控件工具箱添加，则该属性的设置可通过属性窗口辅助生成（如本例）
- ✓如果在代码中利用new 创建该对象，则需手工添加（也可通过其它辅手段），只不同数据源，连接字符串设置格式不一样，故手工设置比较麻烦。

主要方法：

Open() 打开并正式建立连接

Close() 关闭连接

2) OleDbCommand对象

该对象为OLE DB .NET数据提供程序的核心对象之一。

其最重的属性：

CommandType属性 命令类型，三种可选：存储过程、表、SQL命令文本

CommandText属性 命令文本

Connection属性 对OleDbConnection的引用。

重要方法：



ADO.NET组件对象的使用

ExecuteNonQuery方法

[格式]: public int ExecuteNonQuery();

执行不返回记录集的命令

ExecuteReader方法

[格式]: public OleDbDataReader ExecuteReader();

执行返回记录集的命令如select 命令。

返回类型为OleDbDataReader 记录集

3) OleDbDataReader对象

该对象为OLE DB .NET数据提供程序的核心对象之一。

DataReader对象的常用方法

Read方法

[格式]: public bool Read();

读取下一行，如返回真则还有数据，否则已结束。

Close方法

[格式]: public void Close();

记录读完后和此方法关闭数据集。

Get×××方法

[格式]: public ××× Get××× (int ordinal);

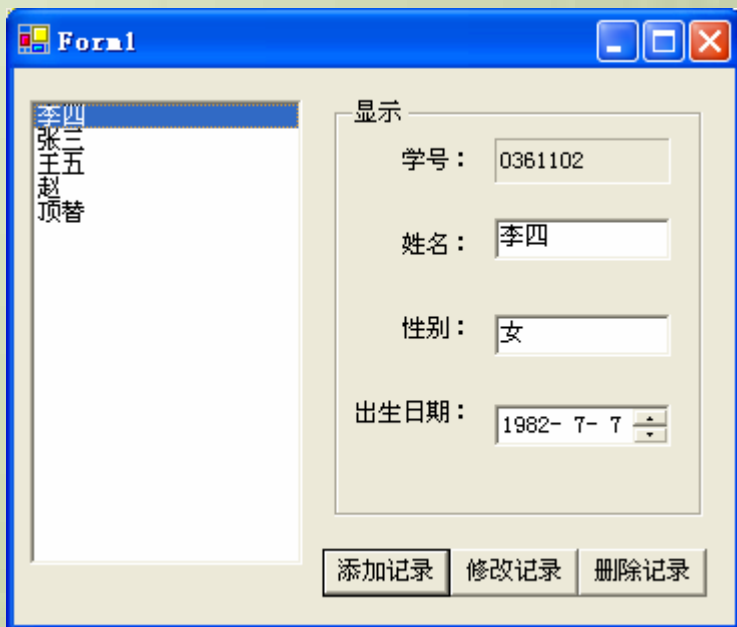
如读取列为字符，则使用GetString(列序号)



ADO.NET组件对象的使用

[项目二]利用OLE DB .NET 数据提供程序基本对象制作一个学生信息管理程序：

程序主界面为下图一：在列表框中选中某同学后，右边显示出该生信息；在右边显示区修改相关信息后，单击“修改记录”可修改记录；如果单击“删除”按钮，则可删除选定记录；如果单击“添加记录”，则出现窗口如图二：输入内容后，点击添加即可返回主窗口。



Form1 is the main window of the student information management program. It features a list box on the left containing the names of students: 李四, 张三, 王五, 赵六, and 顶替. To the right of the list box is a section titled "显示" (Display) which shows the details of the selected student. The details include: 学号 (Student ID) 0361102, 姓名 (Name) 李四, 性别 (Gender) 女, and 出生日期 (Date of Birth) 1982- 7- 7. At the bottom of the window are three buttons: 添加记录 (Add Record), 修改记录 (Modify Record), and 删除记录 (Delete Record).



Form2 is the "Add Record" window. It is titled "添加记录" (Add Record). It contains four input fields: 学号 (Student ID), 姓名 (Name), 性别 (Gender), and 出生日期 (Date of Birth). The 出生日期 field is a date picker showing 2006- 5-28. At the bottom of the window is a button labeled 添加 (Add).

ADO.NET组件对象的使用

1、制作过程

1) 利用Access创一数据库db1.mdb，库结构及内容同项目一。

2)生成应用程序主界面，添加下以控件：

✓一个列表框listBox1

✓一个组框groupBox1

✓在组框中添加五个标签：label1—label4 作为显示标题，label5用来显示学号

✓在组框中添加两个文本框textBox1--2，分别显示姓名和性别

✓在组框中添加一个日期时间设置控件dateTimePicker1，该控件在工具箱上为 DateTimePicker

✓在窗体中添加三个按钮button1—3

✓添加一连接控件oleDbConnection1

3)主窗体中主要控件属性设置

(1) 显示姓名和性别的两文本框的MaxLength属性值分别4和1个字符（同数据表的xm字段和xb字段长度相一致）。

(2) 日期选择控件属性设置：

| 属性名 | 属性值 | 说明 |
|------------|-------|----------|
| Format | Short | 短日期格式 |
| ShowUpDown | true | 显示上下调节按钮 |



ADO.NET组件对象的使用

(3) OleDbConnection1控件属性

在属性窗口选择属性ConnectionString并对其设置，设置方法见本讲项目一。
其它属性设置参考主窗口界面。

4) 添加子窗体Form2 并添加如下控件：

- ✓ 一个组框
- ✓ 在组框中添加四个标签label1—4，分别用以显示四个标题
- ✓ 在组框中添加三个文本框textBox1—3，分别输入学号值、姓名值、性别值
- ✓ 在组框中添加一个日期时间选择控件dateTimePicker1，用来输入日期
- ✓ 在窗体添加一按钮button1

5) 子窗体Form2中各控件主要属性设置：

- (1) 三个文本框MaxLength属性设置，设置最多字符数分别设为10、4、1（同xh,xm,xs字段宽度一致）
- (2) dateTimePicker1属性设置，同主窗体相应控件相同设置
- (3) 按钮button1 标题设为“添加”； DialogResult属性设为OK

其它属性设置略。

6) 代码实现

- (1) 在主窗体代码前加上OLE DB .NET 数据提供程序的名字空间：
using System.Data.OleDb;



ADO.NET组件对象的使用

(2) 在项目中添加一学生类Student，添加操作见第八讲的项目八

(3) 在学生类中添加四个字段，同数据表中的四个数据列一致：

```
public string xh;  
public string xm;  
public string xb;  
public DateTime csrq;
```

(添加步骤见第八讲的项目八)

(4) 在学生类中重写虚拟方法ToString():

```
public override string ToString()  
{  
  
    return xm;  
  
}
```

(操作步骤见第八讲的项目八)

(5) 在主窗体中添加一个自定义方法void FillList():

```
public void FillList()  
{  
    listBox1.Items.Clear();           //清除列表框
```



ADO.NET组件对象的使用

```
OleDbCommand cmd= new OleDbCommand();  
cmd.CommandType=CommandType.Text;  
cmd.Connection=oleDbConnection1;  
cmd.CommandText="select * from tb1";  
oleDbConnection1.Open();  
OleDbDataReader rd = cmd.ExecuteReader();  
while(rd.Read()) {  
    Student stu = new Student();  
    stu.xh=rd.GetString(0);  
    stu.xm=rd.GetString(1);  
    stu.xb=rd.GetString(2);  
    stu.csrq=rd.GetDateTime(3);  
    listBox1.Items.Add(stu);  
}  
rd.Close();  
oleDbConnection1.Close();  
if(listBox1.Items.Count!=0) listBox1.SetSelected(0,true);  
}
```

(6) 添加窗体的Load事件处理方法:

```
private void Form1_Load(object sender, System.EventArgs e)  
{  
    this.FillList();  
}
```



ADO.NET组件对象的使用

(7) 添加列表框SelectedIndexChanged事件处理方法:

```
private void listBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1) {
        Student stu = (Student)(listBox1.SelectedItem);
        label5.Text=stu.xh;
        textBox1.Text=stu.xm;
        textBox2.Text=stu.xb;
        dateTimePicker1.Value=stu.csrq;
    }else
    {
        label5.Text="";
        textBox1.Text="";
        textBox2.Text="";
        dateTimePicker1.Text="";
    }
}
```

(8) 添加“删除”按钮button3的Click事件处理方法:

```
private void button3_Click(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1)
    {
```



ADO.NET组件对象的使用

```
Student stu =(Student)listBox1.SelectedItem;  
OleDbCommand cmd= new OleDbCommand();  
cmd.CommandType=CommandType.Text;  
cmd.Connection=oleDbConnection1;  
cmd.CommandText="delete from tb1 where xh=\"";  
cmd.CommandText+=stu.xh+"\"";  
oleDbConnection1.Open();  
cmd.ExecuteNonQuery();  
oleDbConnection1.Close();  
this.FillList();
```

```
}
```

```
}
```

(9) 添加“修改”按钮button2的Click事件处理方法:

```
private void button3_Click(object sender, System.EventArgs e)
```

```
{
```

```
    if(listBox1.SelectedIndex!=-1)
```

```
    {
```

```
        int index=listBox1.SelectedIndex;  
        OleDbCommand cmd= new OleDbCommand();  
        cmd.CommandType=CommandType.Text;  
        cmd.Connection=oleDbConnection1;
```



ADO.NET组件对象的使用

```
cmd.CommandText="update tb1 set xm=\"'+textBox1.Text+'\",xb=\"'";  
cmd.CommandText+=textBox2.Text+"\",csrq=\"'+dateTimePicker1.Text+' \";  
cmd.CommandText+=" where xh=\"'+label5.Text+'\"";  
oleDbConnection1.Open();  
cmd.ExecuteNonQuery();  
oleDbConnection1.Close();  
this.FillList();  
listBox1.SelectedIndex=index;  
}  
}
```

(10) 在窗体Form2中添加四个字段:

```
public string xh;  
public string xm;  
public string xb;  
public string csrq;
```

(11) 添加窗体Form2的”确认添加”按钮的Click事件处理方法:

```
private void button1_Click(object sender, System.EventArgs e)  
{  
    if(textBox1.Text=="" ||textBox2.Text=="") {  
        MessageBox.Show("学号和姓名不能为空");  
        return;  
    }  
}
```



ADO.NET组件对象的使用

```
xh=textBox1.Text;  
xm=textBox2.Text;  
xb=textBox3.Text;  
csrq=datetimePicker1.Text;
```

```
}
```

(12) 最后添加主窗体Form1的“添加记录”按钮button1的Click事件处理方法:

```
private void button2_Click(object sender, System.EventArgs e)
```

```
{
```

```
    Form2 dlg = new Form2();
```

```
    if(dlg.ShowDialog()==DialogResult.OK) {
```

```
        OleDbCommand cmd= new OleDbCommand();
```

```
        cmd.CommandType=CommandType.Text;
```

```
        cmd.Connection=oleDbConnection1;
```

```
        cmd.CommandText="insert into tb1(xh,xm,xb,csrq) values (\";
```

```
        cmd.CommandText+=dlg.xh+"\", \"'+dlg.xm+"\", \"'+dlg.xb+"\", \"\";
```

```
        cmd.CommandText+=dlg.csrq+"\"");
```

```
        oleDbConnection1.Open();
```

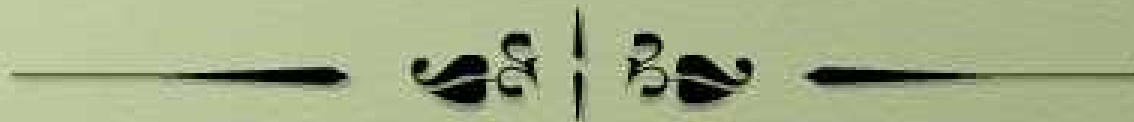
```
        cmd.ExecuteNonQuery();
```

```
        oleDbConnection1.Close();
```

```
        this.FillList();
```

```
    }
```

```
}
```



本课作业

- 1、ADO.NET数据提供程序有四种，请说出其中的两种，并指出在.NET中引用它们的名字空间。
- 2、数据提供程序的四个核心对象是什么？分别说出它们的作用。
- 3、简述利用DataReader对象读取数据表记录的过程或步骤。



第二讲 数据集

一、主要内容：

- 1、数据集常用操作
- 2、数据表的常用操作

二、目的和要求：

- 1、掌握数据集的常用方法
- 2、掌握对数据表进行添加、编辑、删除记录行的各种操作。



范例介绍

[项目三]数据集应用范例。将项目二改用数据集实现，一个学生信息管理程序：

程序主界面为下图一：在列表框中选中某同学后，右边显示出该生信息；在右边显示区修改相关信息后，单击“修改记录”可修改记录；如果单击“删除”按钮，则可删除选定记录；如果单击“添加记录”，则出现窗口如图二：输入内容后，点击添加即可返回主窗口；单击“还原所有”，可一次恢复所有的操作；单击“保存所有”后，以前所作的操作将保存到数据库中。



Form1 is the main interface of the student information management program. It features a list box on the left containing the names '张三' (Zhang San) and '李四' (Li Si). To the right of the list box is a '显示' (Display) section with four input fields: '学号' (Student ID) with the value '0584101', '姓名' (Name) with '张三', '性别' (Gender) with '男' (Male), and '出生日期' (Date of Birth) with '1984- 9-11'. Below these fields are four buttons: '添加记录' (Add Record), '修改记录' (Modify Record), '删除记录' (Delete Record), and '保存所有' (Save All). At the bottom right, there is a '还原所有' (Restore All) button.



Form2 is the '添加记录' (Add Record) dialog box. It contains four input fields: '学号' (Student ID), '姓名' (Name), '性别' (Gender), and '出生日期' (Date of Birth) with the value '2006- 9-11'. A '添加' (Add) button is located at the bottom center of the form.

范例介绍

1、制作过程

1) 利用Access创一数据库db1.mdb，库结构及内容同项目一。

2)生成应用程序主界面，添加下以控件：

✓一个列表框listBox1

✓一个组框groupBox1

✓在组框中添加五个标签：label1—label4 作为显示标题，label5用来显示学号

✓在组框中添加两个文本框textBox1--2，分别显示姓名和性别

✓在组框中添加一个日期时间设置控件dateTimePicker1，该控件在工具箱上为 DateTimePicker

✓在窗体中添加五个按钮button1—5

3)主窗体中主要控件属性设置

(1) 显示姓名和性别的两文本框的MaxLength属性值分别4和1个字符（同数据表的xm字段和xb字段长度相一致）。

(2) 日期选择控件属性设置：

| 属性名 | 属性值 | 说明 |
|------------|-------|----------|
| Format | Short | 短日期格式 |
| ShowUpDown | true | 显示上下调节按钮 |

(3) 数据组件的添加及配置



范例介绍

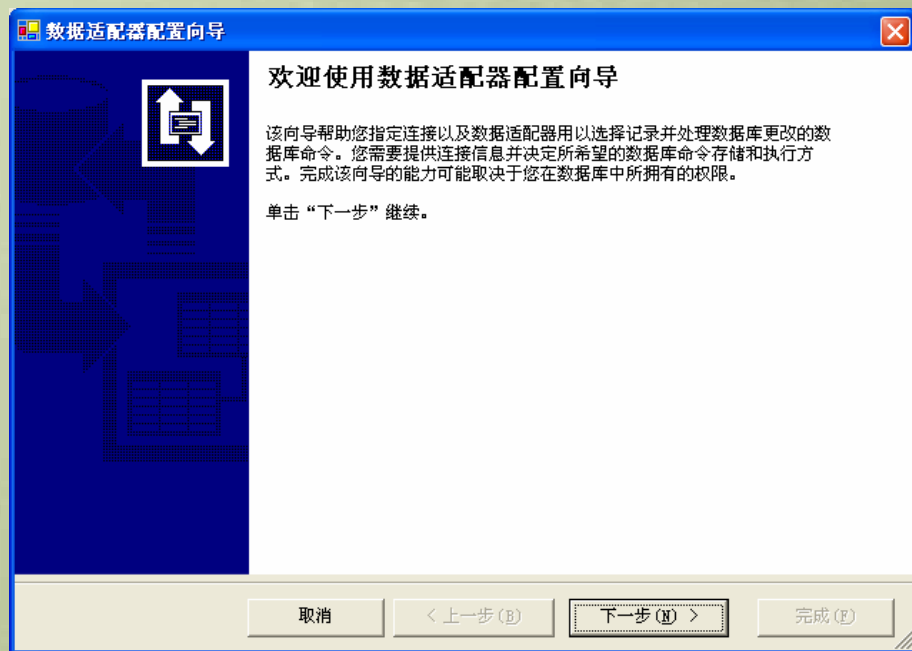
本例将通过数据集来操作数据，所以要添加连接(Connection)、数据适配器(DataAdapter)、数据集(DataSet)三种组件。

这三种组件的添加和配置方法比较灵活，这里只介绍两种常用方法。

方法一：

(I) 先创建并配置连接组件，产生连接对象，默认名oleDbConnection1。具体操作见前面项目（还是连接到上个项目的数据库）。

(II) 在工具栏数据组件栏中选OleDbDataAdapter组件，出现数据适配器的设置向导，第一步：



范例介绍

第二步：在下拉列表中选择刚建立好的连接，或点右边按钮新建一个连接。

下一步：

数据适配器配置向导

选择您的数据连接
数据适配器将使用此连接执行查询以加载和更新数据。

从服务器资源管理器中的当前数据连接列表中进行选择。如果需要的连接未列出，则添加新连接。

数据适配器应使用哪一种数据连接？ (U)

ACCESS: C:\Documents and Settings\ryq\My Documents\db2.mdb Admin

新建连接 (C)...

< 上一步 (B) 下一步 (N) > 完成 (F)

数据适配器配置向导

选择查询类型
数据适配器使用 SQL 语句或存储过程。

数据适配器如何访问数据库？

☒ **使用 SQL 语句 (S)**
指定一个 Select 语句来加载数据，向导将生成 Insert、Update 和 Delete 语句，以保存数据更改。

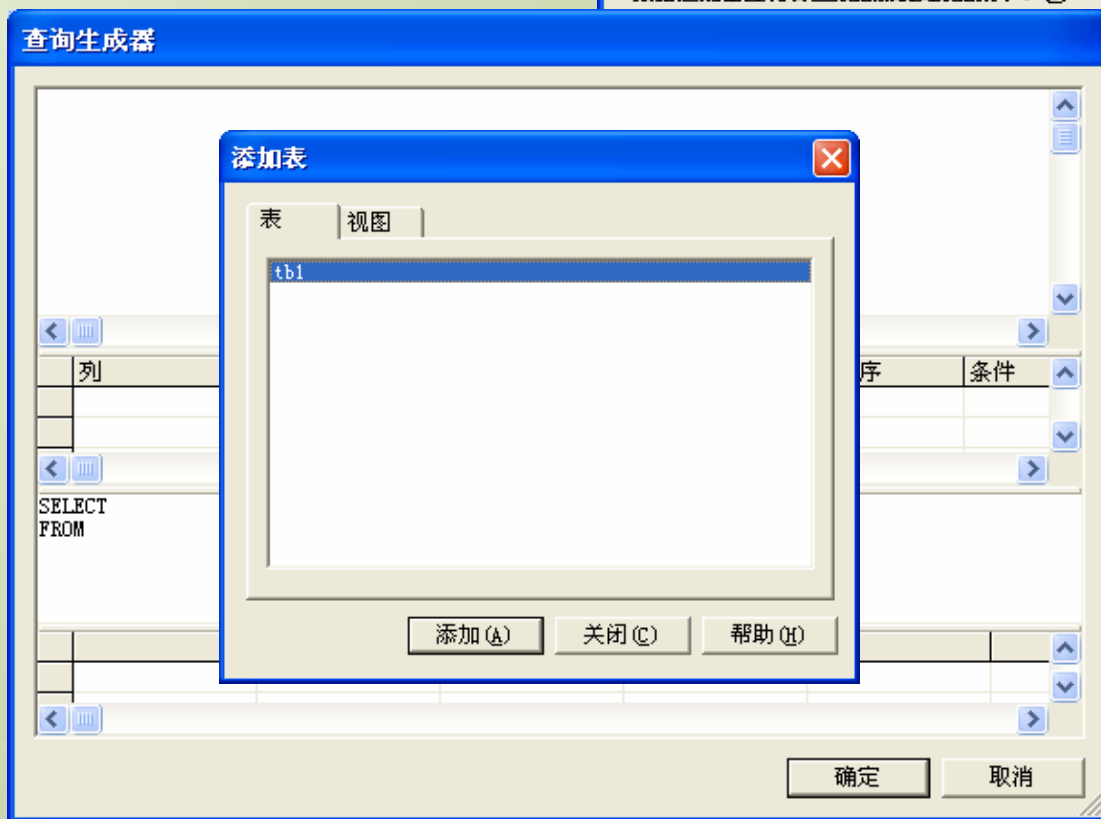
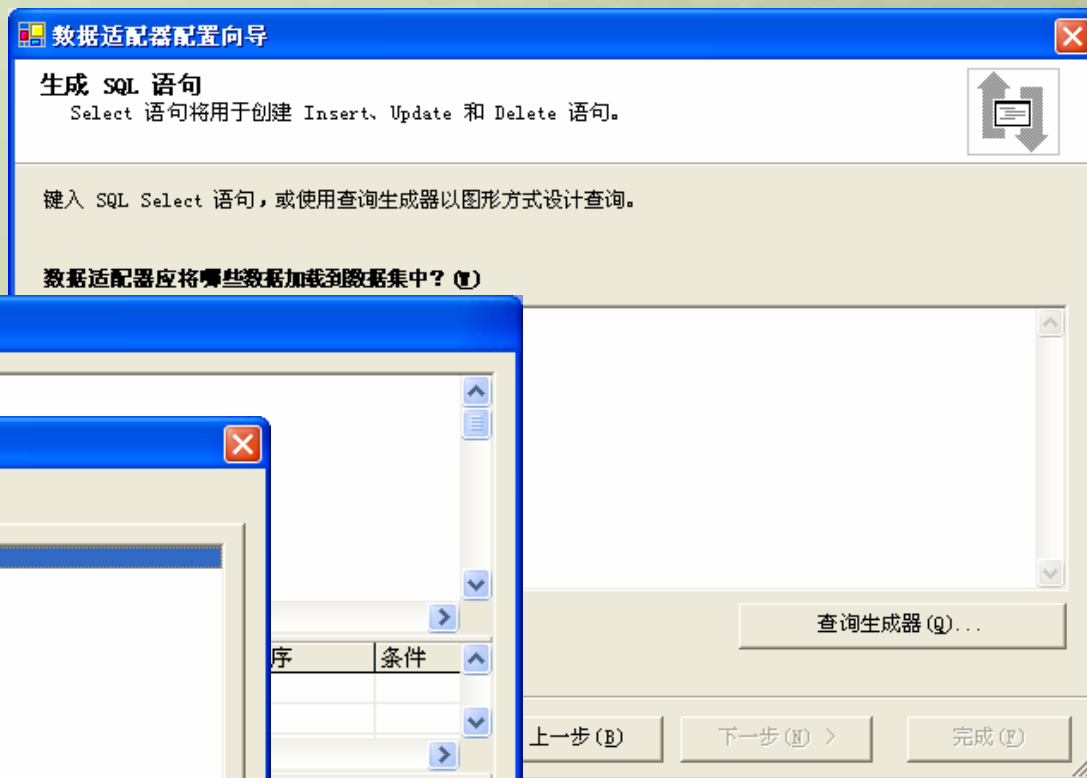
☐ **创建新存储过程 (C)**
指定 Select 语句，向导将生成新的存储过程以选择、插入、更新和删除记录。

☐ **使用现有存储过程 (E)**
为每个操作选择一个现有存储过程 (Select、Insert、Update 和 Delete)。

取消 < 上一步 (B) 下一步 (N) > 完成 (F)

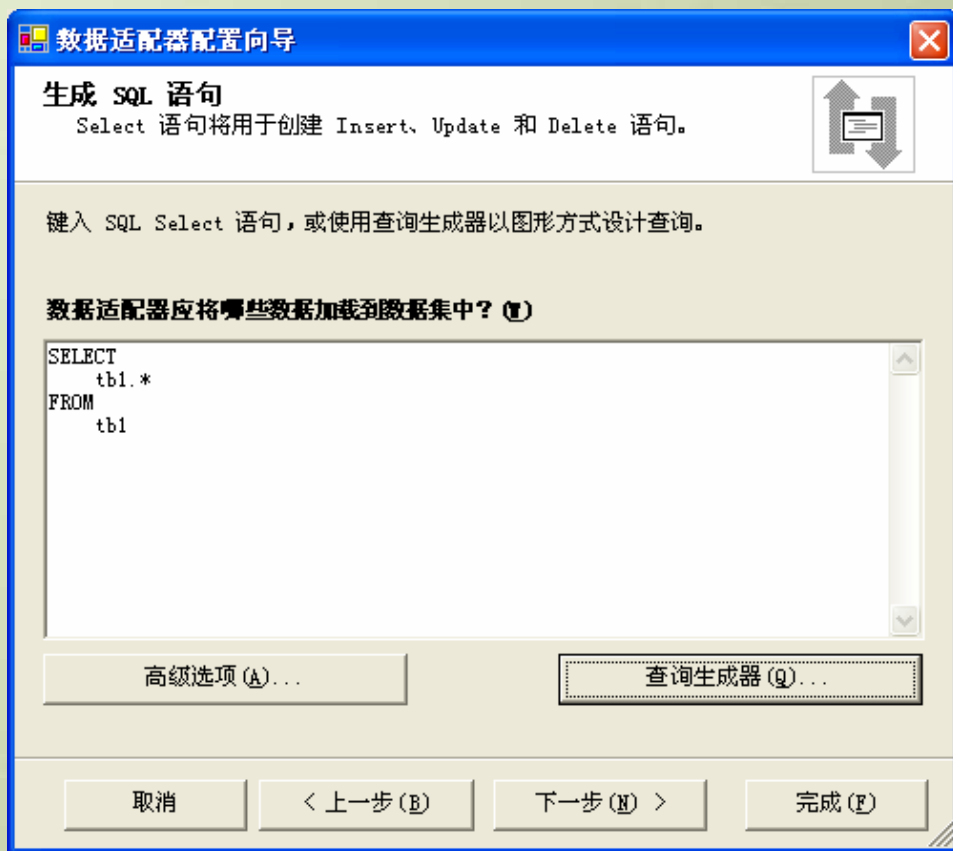
范例介绍

下一步：选择查询生成器。
出现下图：



范例介绍

添加数据库中的表tb1，然后关闭“添加表”窗口，现下图：
在其中选“所有列”项后，确定，回到下面窗口：



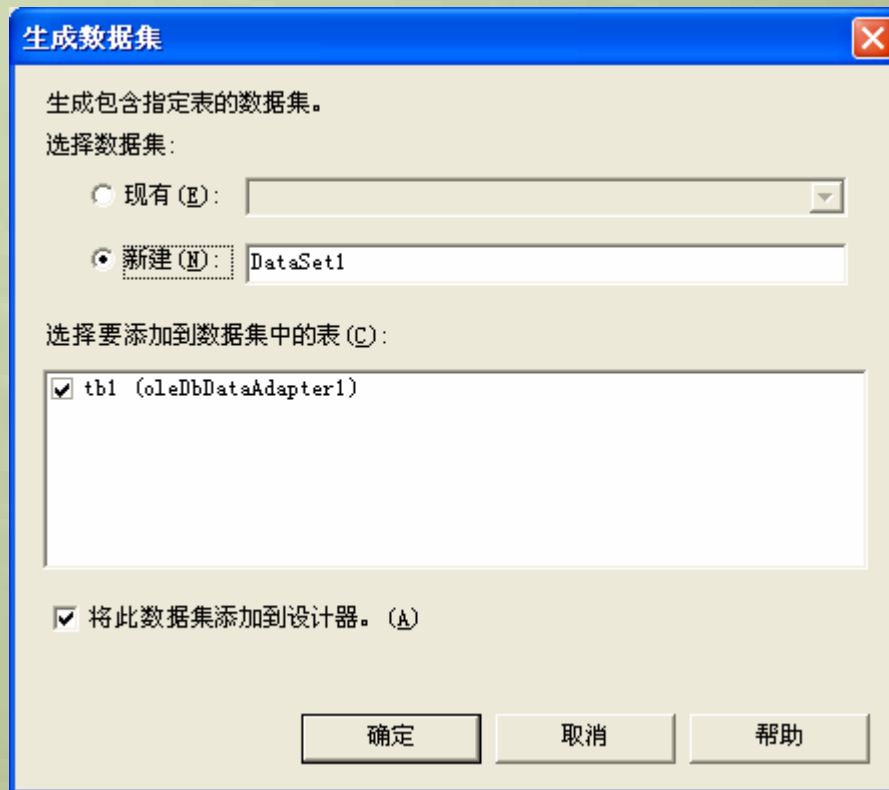
然后点下一步直到完成。此时生成了数据表tb1的数据适配器oleDbDataAdapter1

范例介绍

(III)生成数据集。在窗体下面的组件oleDbDataAdapter1上点右键出现弹出菜单，选择生成类型数据集，出现下面窗口：

在其中选择新建，然后点确定，则生成新的数据集类型及对象，对象名为DataSet1。

方法二：通过服务器资源管理器添加，具体操作略。



生成数据集

生成包含指定表的数据集。

选择数据集:

☐ 现有 (E):

☒ 新建 (N): DataSet1

选择要添加到数据集中的表 (C):

☒ tbl1 (oleDbDataAdapter1)

☒ 将此数据集添加到设计器。 (A)

确定 取消 帮助

范例介绍

4) 添加子窗体Form2 并添加如下控件:

- ✓ 一个组框
- ✓ 在组框中添加四个标签label1—4, 分别用以显示四个标题
- ✓ 在组框中添加三个文本框textBox1—3, 分别输入学号值、姓名值、性别值
- ✓ 在组框中添加一个日期时间选择控件dateTimePicker1, 用来输入日期
- ✓ 在窗体添加一按钮button1

5) 子窗体Form2中各控件主要属性设置:

- (1) 三个文本框MaxLength属性设置, 设置最多字符数分别设为10、4、1
(同xh,xm,xb字段宽度一致)
- (2) dateTimePicker1属性设置, 同主窗体相应控件相同设置
- (3) 按钮button1 标题设为“添加”;

6) 代码实现

(1) 在主窗体代码前加上OLE DB .NET 数据提供程序的名字空间:
using System.Data.OleDb;



范例介绍

(2) 为了能使数据行对象在加入列表框后能在其中显示学生姓名，需要在数据行类中重写虚拟方法ToString():

```
public override string ToString()
{
    return xm;
}
```

(3) 在主窗体中添加一个自定义方法void FillList():

```
public void FillList()
{
    listBox1.Items.Clear();
    foreach (DataRow row in this.dataSet11.tb1.Rows)
    {
        if(row.RowState!=DataRowState.Deleted)
            listBox1.Items.Add(row);
    }
    if(listBox1.Items.Count!=0) listBox1.SetSelected(0,true);
}
```



范例介绍

(3) 添加窗体的Load事件处理方法:

```
private void Form1_Load(object sender, System.EventArgs e)
{
    this.oleDbDataAdapter1.Fill(this.dataSet11);
    this.FillList();
}
```

(4) 添加列表框SelectedIndexChanged事件处理方法:

```
private void listBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1)
    {
        DataRow row = (DataRow)(listBox1.SelectedItem);
        label5.Text=row["xh"].ToString();
        textBox1.Text=row["xm"].ToString();
        textBox2.Text=row["xb"].ToString();
        dateTimePicker1.Text=row["csrq"].ToString();
    }else {
        label5.Text="";
        textBox1.Text="";
        textBox2.Text="";
        dateTimePicker1.Text="";
    }
}
```



范例介绍

(5) 添加“删除”按钮button3的Click事件处理方法:

```
private void button1_Click(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1)
    {
        DataRow row =(DataRow)listBox1.SelectedItem;
        row.Delete();
        this.FillList();
    }
}
```

(6) 添加“修改”按钮button2的Click事件处理方法:

```
private void button3_Click(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1)
    {
        int index=listBox1.SelectedIndex;
        DataRow row=(DataRow)listBox1.SelectedItem;
        row["xm"]=this.textBox1.Text;
        row["xb"]=this.textBox2.Text;
        row["csrq"]=this.dateTimePicker1.Value;
        this.FillList();
        listBox1.SelectedIndex=index;
    }
}
```



范例介绍

(7) 在窗体Form2中添加四个字段:

```
public string xh;  
public string xm;  
public string xb;  
public string csrq;
```

(8) 添加窗体Form2的”确认添加”按钮的Click事件处理方法:

```
private void button1_Click(object sender, System.EventArgs e)  
{  
    if(textBox1.Text==" " ||textBox2.Text==" " )  
    {  
        MessageBox.Show("学号和姓名不能为空");  
        return;  
    }  
    xh=textBox1.Text;  
    xm=textBox2.Text;  
    xb=textBox3.Text;  
    csrq=dateTimePicker1.Text;  
    this.DialogResult=DialogResult.OK;  
}
```



范例介绍

9) 然后添加主窗体Form1的“添加记录”按钮button1的Click事件处理方法:

```
private void button2_Click(object sender, System.EventArgs e)
{
    Form2 dlg = new Form2();
    if(dlg.ShowDialog()==DialogResult.OK)
    {
        DataRow row=this.dataSet11.tb1.NewRow();
        row["xh"]=dlg.xh;
        row["xm"]=dlg.xm;
        row["xb"]=dlg.xb;
        row["csrq"]=DateTime.Parse(dlg.csrq);
        this.dataSet11.tb1.Rows.Add(row);
        this.FillList();
    }
}
```



范例介绍

10) 然后添加主窗体Form1的“保存所有”按钮button4的Click事件处理方法:

```
private void button4_Click(object sender, System.EventArgs e)
{
    this.oleDbDataAdapter1.Update(this.dataSet11);
}
```

11) 最后添加主窗体Form1的“还原所有”按钮button5的Click事件处理方法:

```
private void button5_Click(object sender, System.EventArgs e)
{
    this.dataSet11.RejectChanges();
    this.FillList();
}
```

2.相关知识介绍

◆ DataAdapter 对象及使用

DataAdapter是DataSet对象和数据源之联系的桥梁,主要功能是从数据源中检索数据,填充数据集对象中的表、把用户对数据集DataSet对象做出的更改写入到数据源。关于填充数据集和更新数据源操作请参考本项目窗体Load事件代码及按钮四（保存所有）的有关代码。



相关知识

DataAdapter常用方法：

(1) **Fill方法**：该方法有多种重载格式，其主要作用是从数据源中提取数据以填充数据集。下面是常用调用格式：

Fill(数据集对象)：当数据集中只有一个数据表时。

Fill(数据集对象，数据集中的表名)：填充指定的表，当数据集中有多个表时。

(2) **Update方法**：该方法用于更新数据源，下面是常用的调用格式：

Update (数据集对象)：对数据集中指定的数据表所作的任何修改全部更新到数据源，该方法常用于数据集中只有一个数据表时。

Update (数据集对象，数据表)：该方法用于数据集中有多个数据表时。

DataAdapter常用属性：

SelectCommand

InsertCommand

UpdateCommand

DeleteCommand

这四个属性分用来指向四个命令对象（对数据源查询、插入、更新、删除时将用到），是DataAdapter用来填充数据集和更新数据源的主要依据。

◆ DataSet对象及其使用

数据集对象包含数据表对象的集合和关系的集合



相关知识

数据集的构成见第一章结构图。

数据集对象的访问：

DataSet对象包含数据表的集合**Tables**(元素类型为**DataTable**)，而**DataTable**对象包含数据行的集合**Rows**(元素类型为**DataRow**)、数据列的集合**Columns**(元素类型为**DataColumn**)。

表的常用方法和属性：

NewRow()，其作用是利用当前表的模式产生一新行。

Rows，表示数据行的集合。该集合对象包含对表中所有记录的引用（通过下标引用，如**Rows[0]**代表第一行记录）。

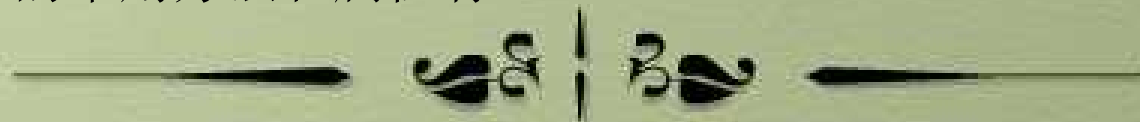
Rows对象的常用属性：

Count：记录条数

ADD（数据行）往数合中添加新的记录。

数据行（**DataRow**），代表表中的一行记录。通过列名称下标或位置下标可访问数据行列(字段)。

数据行的常用方法和属性有：



相关知识

Delete () 删除当前行。

BeginEdit()开始编辑当前行。

EndEdit()结束编辑当前行。



第三讲 数据绑定

一、主要内容：

- 1、数据绑定基本知识
- 2、数据绑定的相关实例

二、目的和要求：

- 1、掌握简单数据绑定及操作
- 2、掌握复杂数据绑定及操作



范例介绍

[项目四]数据绑定应用范例。将项目三改用数据绑定的方法实现：

程序主界面为下图一：在列表框中选中某同学后，右边显示出该生信息；在右边显示区修改相关信息后，可直接修改记录；如果单击“删除”按钮，则可删除选定记录；如果单击“添加记录”，则出现窗口如图二：输入内容后，点击添加即可返回主窗口；单击“还原所有”，可一次恢复所有的操作；单击“保存所有”后，以前所作的操作将保存到数据库中。

同[项目三]要求基本一样。

The screenshot shows a Windows-style window titled "Form1". On the left is a list box containing the names "张三", "王九", "王五", and "周七", with "张三" selected. To the right of the list box is a section titled "显示" (Display). This section contains four input fields: "学号:" (Student ID) with the value "sdfs", "姓名:" (Name) with the value "张三", "性别:" (Gender) with the value "s", and "出生日期:" (Date of Birth) with a date picker showing "2006-10- 5". At the bottom of the window are four buttons: "添加" (Add), "删除" (Delete), "保存" (Save), and "还原" (Reset).

The screenshot shows a Windows-style window titled "Form2". It contains a section titled "添加记录" (Add Record). This section has four input fields: "学号:" (Student ID), "姓名:" (Name), "性别:" (Gender), and "出生日期:" (Date of Birth) with a date picker showing "2006- 9-11". At the bottom of the window is a single button labeled "添加" (Add).

范例介绍

1、制作过程

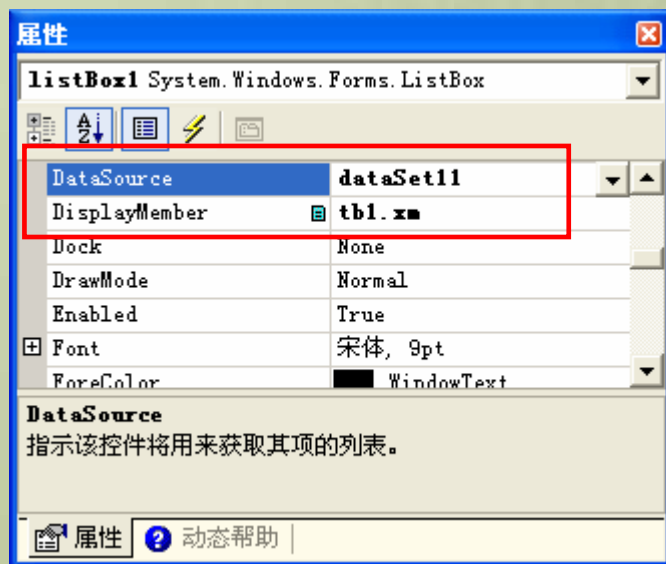
1) 利用Access创一数据库db1.mdb，库结构及内容同项目一。

2)生成应用程序主界面(略)

3)主窗体中主要控件属性设置

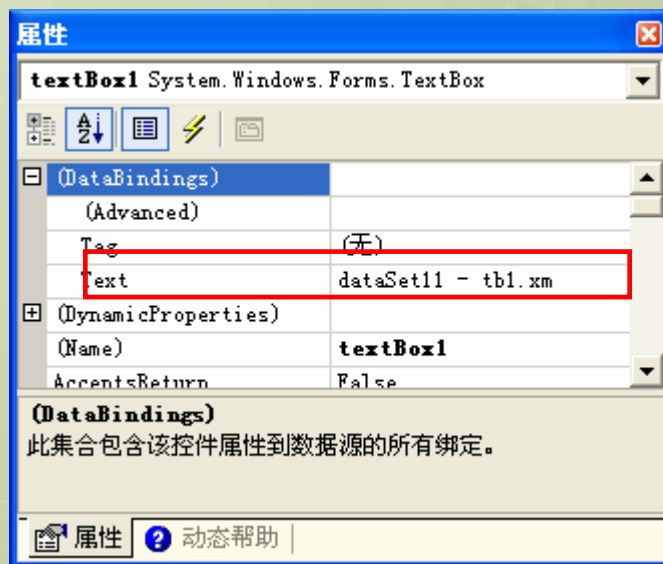
(1) 数据组件的添加及配置(同上项目)

(2) 设置列表框的两个重要属性,如下图(注意，用右边的下拉列表选择):



范例介绍

(3) 设置学号、姓名、性别、出生日期各显示框控件的数据绑定集合属性中的数据绑定，如姓名文本框的数据绑定的设置如下图（表示文本框的TEXT属性将同数据表tb1的xm字段绑定）：



(4) 日期选择控件属性设置：

| 属性名 | 属性值 | 说明 |
|------------|-------|----------|
| Format | Short | 短日期格式 |
| ShowUpDown | true | 显示上下调节按钮 |

4) 用来添加记录的窗体Form2及其控件创建同上个项目，这里从略。



范例介绍

5) 代码实现

(1) 在主窗体代码前加上OLE DB .NET 数据提供程序的名字空间:

```
using System.Data.OleDb;
```

(2) 添加窗体的Load事件处理方法:

```
private void Form1_Load(object sender, System.EventArgs e)
{
    this.oleDbDataAdapter1.Fill(this.dataSet11);
}
```

注意, 到这里为止, 程序可运行并实现基本功能, 但还不能添加和删除记录!

可见, 利用数据绑定的方法要比前面的方法要简单得多。

下面继续添加下面的代码, 使程序更完整。

(3) 添加“删除”按钮的Click事件处理代码:

```
private void button1_Click(object sender, System.EventArgs e)
{
    if(listBox1.SelectedIndex!=-1)
    {
        DataRowView rw=(DataRowView)(listBox1.SelectedItem);
        rw.Delete();
    }
}
```



范例介绍

(3) 添加“添加”按钮的Click事件处理代码:

```
private void button2_Click(object sender, System.EventArgs e)
{
    Form2 dlg=new Form2();
    if(dlg.ShowDialog()==DialogResult.OK)
    {
        DataRow row=this.dataSet11.tb1.NewRow();
        row["xh"]=dlg.xh;
        row["xm"]=dlg.xm;
        row["xb"]=dlg.xb;
        row["csrq"]=DateTime.Parse(dlg.csrq);
        dataSet11.Tables["tb1"].Rows.Add(row);
    }
}
```

(4) 最后，“保存”和“还原”按钮的代码同上个项目完全一样。
这样，花很少的代码即可完成前面范例的所有功能。



相关知识

1) 数据绑定的概念

数据绑定指将控件和数据源捆绑在一起,通过控件来显示或修改数据。

数据绑定有两种类型:简单数据绑定和复杂数据绑定。简单数据绑定是通常是将控件属性绑定到数据表字段的单个值上;复杂绑定通常是把数据集中的某些字段或某个字段中的多行数据绑定到组件的属性上。

支持简单绑定的控件通常有: **TextBox**控件、**Label**控件。

支持复杂绑定的控件通常有: 列表框、组合框、数据表格 (**DataGrid**)

2) 简单数据绑定的方法

A. 通过属性窗口

在属性窗口中打开**DataBindings**属件,从中选择要绑定的属性,在右边输入绑定目标(数据表中的字段)。

B. 通过代码

格式: 控件名.**DataBindings.Add**("属性名", "数据集名", "字段名")

3) 复杂数据绑定的方法

A. 通过属性窗口

先在属性窗口中将**DataSource**属性设置为指定数据源,然后将**DisplayMember**属性设置为相应的表字段(对于列表框或组合框),如果是数据表格控件则设置**DataSource**属性和**DataMember**属性(属性值为相应表)

B. 通过代码

可直在代码中对指定属性进行设定,对于数据表格还可用如下格式:

SetDataBinding(数据集, 数据表)



范例介绍

[项目五]实现数据数导航。程序主界面如下图一，该项目与上一项目导航方式不一样，上一项目是利用列表框选择定位不同的数据行，本项目提供了另一种更自然的记录导航方式，分别通过下面“|<”、“<<”、“>>”、“>|”几个按钮提供“第一条、上一条、下一条、最后一条”记录航导功能。

其它按钮功能同[项目四]一样。



The screenshot shows a Windows application window titled "Form1". Inside the window, there is a form with the following elements:

- 学号: (ID) text box containing "sdfs"
- 姓名: (Name) text box containing "张三"
- 性别: (Gender) text box containing "s"
- 出生日期: (Birth Date) date picker showing "2006-10- 5"
- Navigation buttons: A set of four buttons labeled "|<", "<<", ">>", and ">|". The "|<" button is highlighted with a dashed border.
- Action buttons: A row of four buttons labeled "添加" (Add), "删除" (Delete), "保存" (Save), and "还原" (Restore).

范例介绍

1、制作过程

1) 利用Access创一数据库db1.mdb, 库结构及内容同项目一。

2)生成应用程序主界面(略)

3)主窗体中主要控件属性设置

(1) 数据组件(连接、数据匹配器、数据集)的添加及配置(同上项目)

(2) 设置学号、姓名、性别、出生日期各显示框控件的数据绑定属性, 使以上各控件的TEXT属性同数据表的相应字段绑定起来。

(4) 日期选择控件属性设置:

| 属性名 | 属性值 | 说明 |
|------------|-------|----------|
| Format | Short | 短日期格式 |
| ShowUpDown | true | 显示上下调节按钮 |

4) 用来添加记录的窗体Form2及其控件创建同上个项目, 这里从略。

5) 代码实现

(1) 在主窗体代码前加上OLE DB .NET 数据提供程序的名字空间:

using System.Data.OleDb;

(2) 在类中添加一成员字段:

private System.Windows.Forms.BindingManagerBase bmb;

(3) 添加窗体的Load事件处理方法:



范例介绍

```
private void Form1_Load(object sender, System.EventArgs e)
{
    this.oleDbDataAdapter1.Fill(this.dataSet11);
    bmb=this.BindingContext[this.dataSet11,"tb1"];
}
```

(4)添加“|<”（第一条） 导航按钮的鼠标单击处理方法：

```
private void button1_Click(object sender, System.EventArgs e)
{
    if(bmb.Count!=0)
        bmb.Position=0;
}
```

（5） 添加“<”（上一条） 导航按钮的鼠标单击处理方法：

```
private void button2_Click(object sender, System.EventArgs e)
{
    if(bmb.Count!=0&& bmb.Position>0)
        bmb.Position--;
}
```

（6） 添加“>”（下一条） 导航按钮的鼠标单击处理方法：



范例介绍

```
private void button3_Click(object sender, System.EventArgs e)
{
    if(bmb.Count!=0&&bmb.Position<bmb.Count-1)
        bmb.Position++;
}
```

（6）添加“>|“（最后一条）导航按钮的鼠标单击处理方法：

```
private void button4_Click(object sender, System.EventArgs e)
{
    if(bmb.Count!=0)
        bmb.Position=bmb.Count-1;
}
```

（7）添加窗体（Form2）的“添加”按钮鼠标单击外理方法

```
private void button1_Click(object sender, System.EventArgs e)
{
    if(textBox1.Text==""){
        MessageBox.Show("学号不能为空！");    return;    }
    xh=textBox1.Text;
    xm=textBox2.Text;
    xb=textBox3.Text;
    csrq=dateTimePicker1.Text;
    this.DialogResult=DialogResult.OK;
}
```



范例介绍

(8) “添加记录”按钮鼠标单击处理方法:

```
private void button5_Click(object sender, System.EventArgs e)
{
    Form2 dlg= new Form2();
    if(dlg.ShowDialog()==DialogResult.OK)
    {
        DataRow dr=this.dataSet1.tb1.NewRow();
        dr["xh"]=dlg.xh;
        dr["xm"]=dlg.xm;
        dr["xb"]=dlg.xb;
        dr["csrq"]=dlg.csrq;
        this.dataSet1.tb1.Rows.Add(dr);
    }
}
```

(9) “删除”按钮的鼠标单击处理方法:

```
private void button6_Click(object sender, System.EventArgs e)
{
    if(bmb.Position!=-1) {
        DataRowView drv = (DataRowView)(bmb.Current);
        drv.Delete();
    }
}
```

(10) 最后保存和还原按钮相关代码可参考上面项目。



相关知识

BindingManagerBase对象

该对象可称为绑定管理器对象，该对象对应于窗体引用中的每一个数据源，通过该对象可实现对相应表的记录导航

(1) 获取跟窗体相关的BindingManagerBase对象：

BindingContext[数据集,数据表];

(2) BindingManagerBase对象主要属性：

Count绑定源的行数

Current当前活动行

Postion当前位置



第四讲 关系

一、主要内容：

1、关系建立

2、多表操作

二、目的和要求：

1、掌握关系建立及使用方法



范例介绍

[项目六]利用关系实现多表操作。程序主界面如下：本项目在项目五基础上进行了扩充，显示每位学生的基本情况外，同时显示每位学生的各科成绩（在下面的列表视图控件中同步显示），通过下面的相关编辑控件可添加该生的单科成绩，下面的删除按钮可删除列表视图中选定的成绩；上面有关学生信息的基本操作同项目五。

The screenshot shows a Windows application window titled "Form1" with a tab labeled "学生信息". The form contains the following fields and controls:

- 学号: sdf5
- 姓名: 张三
- 性别: s
- 出生日期: 2006-10- 5

Below the form are navigation buttons: |< << 添加 删除 >> >|.

Below the navigation buttons is a table with the following data:

| 课程 | 成绩 |
|----|----|
| 语文 | 67 |
| 数学 | 90 |
| 英语 | 67 |
| | |
| | |
| | |

At the bottom of the window are controls for adding new grades:

课程: 语文 成绩: 添加 删除 保存 还原

范例介绍

1、制作过程

1) 利用Access创一数据库db1.mdb，库结构及内容在项目一基础上增加两个表sc、kc，如下图。

其中sc保存学生相应单科成绩：id(关键字)唯一标识每条记录，xh为学号，来自表stu中的xh（7个字符），kh为课程号，来自表kc中的关键字kh，score为某生某单科成绩，类型为单精度数字型。

kc表保存课程号和课名对照表，kh为课号（关键字），长度为3的文本型，km为课名，长度不限。

| 字段名称 | 数据类型 |
|-------|------|
| id | 自动编号 |
| xh | 文本 |
| kh | 文本 |
| score | 数字 |

字段属性

常规 | 查阅

| | |
|------|--------|
| 字段大小 | 长整型 |
| 新值 | 递增 |
| 格式 | |
| 标题 | |
| 索引 | 有(无重复) |
| 智能标记 | |

| 字段名称 | 数据类型 |
|------|------|
| kh | 文本 |
| km | 文本 |

字段属性

常规 | 查阅

| | |
|---------------|--------|
| 字段大小 | 3 |
| 格式 | |
| 输入掩码 | |
| 标题 | |
| 默认值 | |
| 有效性规则 | |
| 有效性文本 | |
| 必填字段 | 否 |
| 允许空字符串 | 是 |
| 索引 | 有(无重复) |
| Unicode 压缩 | 是 |
| 输入法模式 | 开启 |
| IME 语句模式(仅日文) | 无转化 |
| 智能标记 | |

范例介绍

然后在kc表中输入下面类似内容：



| | kch | km |
|---|-----|----|
| ▶ | k01 | 语文 |
| | k02 | 数学 |
| | k03 | 英语 |
| * | | |

记录: 1 共有记录数: 3

2)生成应用程序主界面

主要控件基本上同上个项目，除此之个，再在下面添加一个ListView（列表视图）控件、一个ComboBox（组合框）。

3)主窗体中主要控件属性设置

（1）数据组件的添加及配置，因本项目需对三个表操作，所以需三个数据匹配器，可依次如前面项目类似添加可用服务器资源管理器更快：



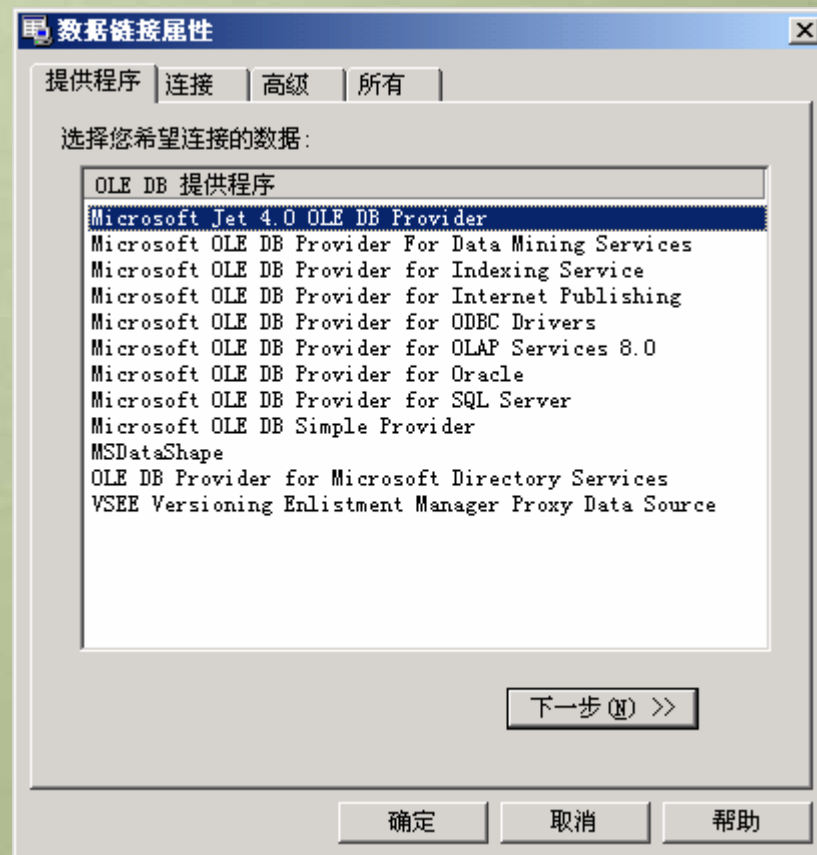
范例介绍

首先，打开服务器资源管理器（下图）：在数据连接上击右键，打开快捷菜单，选择“添加连接.....”。



然后弹出右图：

通过该窗口添加连接到服务器。



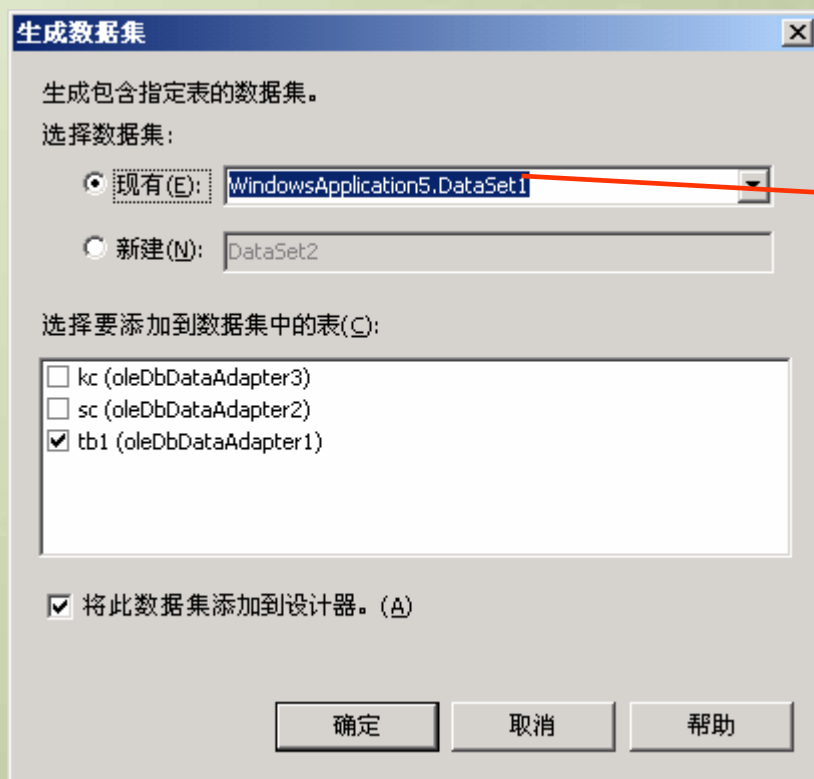
范例介绍

然后在服务器资源管理器中出现了新加的连接（右图）

再然后依次用鼠标将其中的三个表拖到设计窗体，于是在设计窗体的组件栏中出现了三个数据匹配器：

oleDbDataAdapter1、oleDbDataAdapter2、oleDbDataAdapter3。

再分别在各数据匹配器上点击右键，选择“建立数据集”。



第一次选择新建，后两次选现在数据集

范例介绍

(2) 列表视图listView1属性设置:

| 属性名 | 属性值 | 说明 |
|---------------|---------|--------|
| View | Details | 细节列表视图 |
| MultiSelect | false | 不多行选择 |
| GridLines | true | 有网格线 |
| FullRowSelect | true | 可整行选中 |

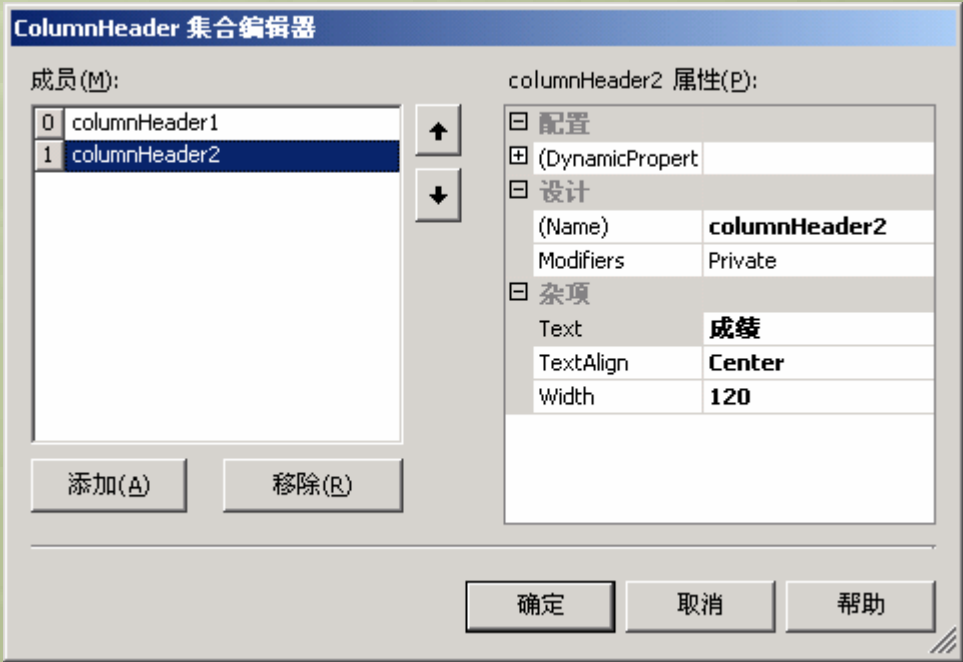
listView1打开Columns属性集合编辑器（下图）：添加两项，表示列表视图的两列，分别设置它们的属性：

columnHeader1（第一列）

| 属性名 | 属性值 |
|-------|-----|
| Text | 课程 |
| Width | 120 |

columnHeader2（第二列）

| 属性名 | 属性值 |
|-----------|--------|
| Text | 成绩 |
| TextAlign | Center |
| Width | 120 |



范例介绍

(3) 组合框ComboBox1属性设置

| 属性名 | 属性值 | 说明 |
|---------------|----------|-----------------------------|
| DataSource | 课程表 (sc) | 数据来源 |
| DisplayMember | km | 将sc中的课名显示在组合框中供选择 |
| ValueMember | kh | 将对应课号作为选择项的SelectedValue属性值 |

(4) 其它属性设置参考项目五。

4) 用来添加记录的窗体Form2及其控件创建同上个项目，这里从略。

5) 代码实现

(1) 在主窗体代码前加上OLE DB .NET 数据提供程序的名字空间：

```
using System.Data.OleDb;
```

(2) 在类中添加一成员字段：

```
private System.Windows.Forms.BindingManagerBase bmb;
```

(3) 添加窗体的Load事件处理方法：

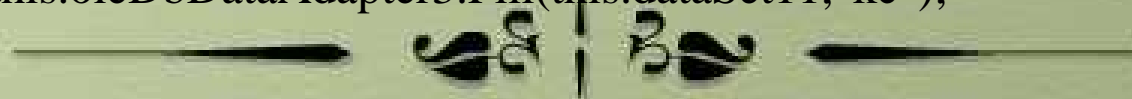
```
private void Form1_Load(object sender, System.EventArgs e)
```

```
{
```

```
    this.oleDbDataAdapter1.Fill(this.dataSet11);
```

```
    this.oleDbDataAdapter2.Fill(this.dataSet11,"sc");
```

```
    this.oleDbDataAdapter3.Fill(this.dataSet11,"kc");
```



范例介绍

```
dataSet11.Relations.Add("re1",dataSet11.stu.xhColumn,dataSet11.sc.xhColumn);  
dataSet11.Relations.Add("re2",dataSet11.kc.khColumn,dataSet11.sc.khColumn);  
bmb=this.BindingContext[this.dataSet11,"stu"];  
bmb.Position=0;  
FillLv();  
bmb.CurrentChanged+=new EventHandler(bmb_CurrentChanged);
```

```
}
```

(4) 其中以面代码中的自定义方法FillLv()作用是填充列表视框控件:

```
private void FillLv()
```

```
{
```

```
    listView1.Items.Clear();
```

```
    DataRowView drv =(DataRowView)(bmb.Current);
```

```
    DataRow[] drarray= drv.Row.GetChildRows("re1");
```

```
    foreach(DataRow dr in drarray)
```

```
    {
```

```
        DataRow temp=dr.GetParentRow("re2");
```

```
        ListViewItem lvi=listView1.Items.Add(temp["km"].ToString());
```

```
        lvi.Tag=dr["id"];
```

```
        lvi.SubItems.Add(dr["score"].ToString());
```

```
    }
```

```
}
```

(5) Form1_Load方法中最后一条代码是为数据绑定管理器添加当前记录改变事件CurrentChanged的处理方法bmb_CurrentChanged:



范例介绍

```
private void bmb_CurrentChanged(object sender, EventArgs e)
{
    FillLv();
}
```

即每次当前记录改变后重新填充列表视图。

(6) 四个记录导航按钮的事件处理方法参考项目五。

(7) 添加窗体 (Form2) 的“添加”按钮鼠标单击外理方法

```
private void button1_Click(object sender, System.EventArgs e)
{
    if(textBox1.Text==""){
        MessageBox.Show("学号不能为空！ ");    return;    }
    xh=textBox1.Text;
    xm=textBox2.Text;
    xb=textBox3.Text;
    csrq=dateTimePicker1.Text;
    this.DialogResult=DialogResult.OK;
}
```

(8) 添加和删除学生信息按钮代码参考项目五。

(9) “添加成绩”按钮事件处理方法：

```
private void button9_Click(object sender, System.EventArgs e)
{
    if(bmb.Position!=-1&&comboBox1.SelectedIndex!=-1)
```



范例介绍

```
{  
    DataRow dr=this.dataSet11.sc.NewRow();  
    dr["xh"]=label5.Text;           //学号显示标签  
    dr["kh"]=comboBox1.SelectedValue;  
    dr["score"]=Convert.ToSingle(textBox3.Text);  
    this.dataSet11.sc.Rows.Add(dr);  
    FillLv();  
}
```

(10) “删除成绩”按钮事件处理方法:

```
private void button10_Click(object sender, System.EventArgs e)  
{  
    if(listView1.SelectedItems.Count!=0)  
    {  
        object id=listView1.SelectedItems[0].Tag;  
        DataRow dr=this.dataSet11.sc.Rows.Find(id);  
        dr.Delete();  
        FillLv();  
    }  
}
```

(11) 保存按钮的事件处理方法:



范例介绍

```
private void button8_Click(object sender, System.EventArgs e)
{
    this.oleDbDataAdapter1.Update(this.dataSet11);
    this.oleDbDataAdapter2.Update(this.dataSet11);
}
```

因为学生表（stu）和成绩表（sc）需要更新，所要依次要调用两个数据匹配器的更新方法。

（12）还原按钮的事件处理方法：

```
private void button7_Click(object sender, System.EventArgs e)
{
    this.dataSet11.RejectChanges();
    FillLv();
}
```



相关知识

1、关系（Relations）集合

数据集的关系集合**Relations**定义数据表之间的关系。

关系集合**Relations**可包含零个或多个数据关系对象**DataRelation**，每个对象表示两个表之间的关系。

数据关系对象**DataRelation**可以在两个表之间建立起行与行的对应关系，使我们在父行和子行之间轻松地移动，给定一个父行，就可以找到与之相关的所有子行；反之，找到子行后，也可找到与之相关的父行。

（1）添加关系

调用关系集合的Add方法：

数据集.**Relations.Add**(“关系名”，父列对象，子列对象)

如上面代码：

```
dataSet11.Relations.Add("re1",dataSet11.stu.xhColumn,dataSet11.sc.xhColumn);
```

表示在数据集的关系集合中添加一个关系名为“re1”的关系。

（2）由主行取得子行集

调用数据行的**GetChildRows**方法：

数据行.**GetChildRows**（“关系名”）

作用：通过指定的关系名取得数据行的子行集合。

如上面的代码：

```
DataRow[] drarray= drv.Row.GetChildRows("re1");
```

（3）由子行取得父行

调用数据行的**GetParentRow**方法：

数据行.**GetParentRow**（“关系名”）。如本项目中的代码：

```
DataRow temp=dr.GetParentRow("re2");
```



相关知识

2、列表视图控件（ListView）使用

外观类似数据网格控件，但功能更强，应用更广。

常用属性：

| | |
|---------------|---------|
| View | 外观风格样式 |
| MultiSelect | 是否多行选择 |
| GridLines | 是否有网格线 |
| FullRowSelect | 是否可整行选中 |
| Items | 所有项目行集合 |
| SelectedItems | 被选项目行集合 |

其中：

Items 是所有列表视图项（ ListViewItem 类对象）的集合。

列表视图项的常用属性有：

| | |
|----------|---------------|
| SubItems | 当前项的所有子项的集合 |
| Tag | 可设置与当前项相关联的数据 |

其中，

列表视图项通过Items集合属性的Add方法添加。

如本项目代码：

```
ListViewItem lvi=listView1.Items.Add(temp["km"].ToString());
```

而当前项的子项通过当前项的子项集合属性SubItems的Add方法添加，如本项目代码
lvi.SubItems.Add(dr["score"].ToString());

如果ListView的 View 属性值为Details，则所有项的标题显示在第一列，对应子项的标题依次显示在所续列。



第二部分 Web数据库编程

朱运乔



第五讲 ASP.NET数据访问方法

一、主要内容

- 建立数据库连接
- 数据集访问
- 直接命令访问

二、目的和要求：

掌握Web环境下数据访问的两种主要方式---
数据集访问和直接执行命令



在Visual Studio.NET中可视化创建数据库

1 使用Connection组件创建数据库连接

- 添加SqlConnection组件SqlConnection1
- 然后设置其ConnectionString属性来指定如何连接数据库
- 选择<新建连接...>后打开数据连接属性
- 在“数据链接属性”对话框中正确设置数据库连接信息后，Visual Studio.NET会自动设置合适的ConnectionString属性值

2 通过服务器资源管理器创建数据库连接

- 创建程序对整个数据源的连接，可从“服务器资源管理器”里拖动某个数据源放进窗体里。
- 连接到数据源的某个特定的表或者视图，可从“服务器资源管理器”里拖动表名称或者视图名称到窗体里。
- 要在程序中使用一个存储过程，拖动存储过程名到窗体里。



编写代码创建数据库连接

可视化地创建数据连接虽然比较便捷，但是灵活性和效率不高，为了用好Connection对象，还是必须掌握如何通过编写代码来创建数据库连接。

1 使用SQL Server.NET数据提供程序连接SQL Server 2000

例如：

```
SqlConnection conn1= new SqlConnection("Server=(local);  
Database=stu; Integrated Security=SSPI;");  
SqlConnection conn2 = new SqlConnection();  
conn2.ConnectionString = "Server=(local); Database= stu;User ID=sa;  
Password=abcd“;
```

2 使用OLEDB.NET数据提供程序连接Access

```
OleDbConnection cnAccess = New OleDbConnection();  
cnAccess.ConnectionString = "Provider= Microsoft.Jet.OLEDB.4.0;Data  
source=C:\WebDB\StudentMS.mdb"
```



编写代码创建数据库连接

3、使用Web.Config文件定义数据连接字符串

- 在Web.Config文件中添加如下节点代码：

```
<appSettings>
```

```
    <add key="DBConnStr"
```

```
        value="Server=(local);
```

```
        Database=stu;
```

```
        User ID=sa;
```

```
        Password=sql" />
```

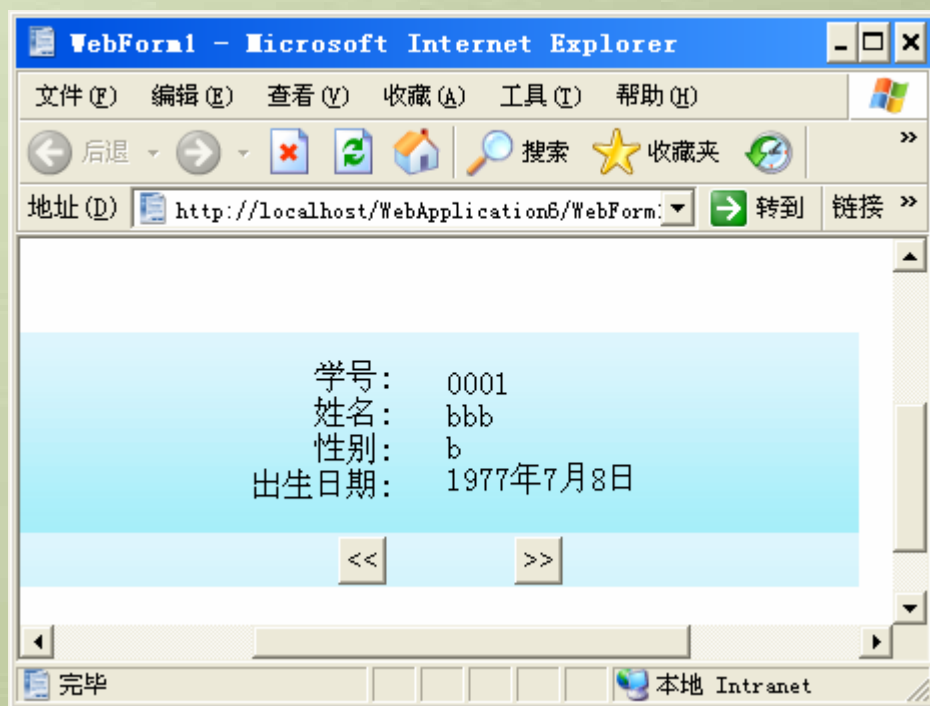
```
</appSettings>
```

- 在程序代码中，通过
System.Configuration.ConfigurationSettings.AppSettings(“键”)来读取Web.config中定义的字符串：
String connStr = System.
Configuration.ConfigurationSettings.AppSettings("DBConnStr");
SqlConnection cnSqlServer = New SqlConnection(connStr);
cnSqlServer.Open();



实例分析

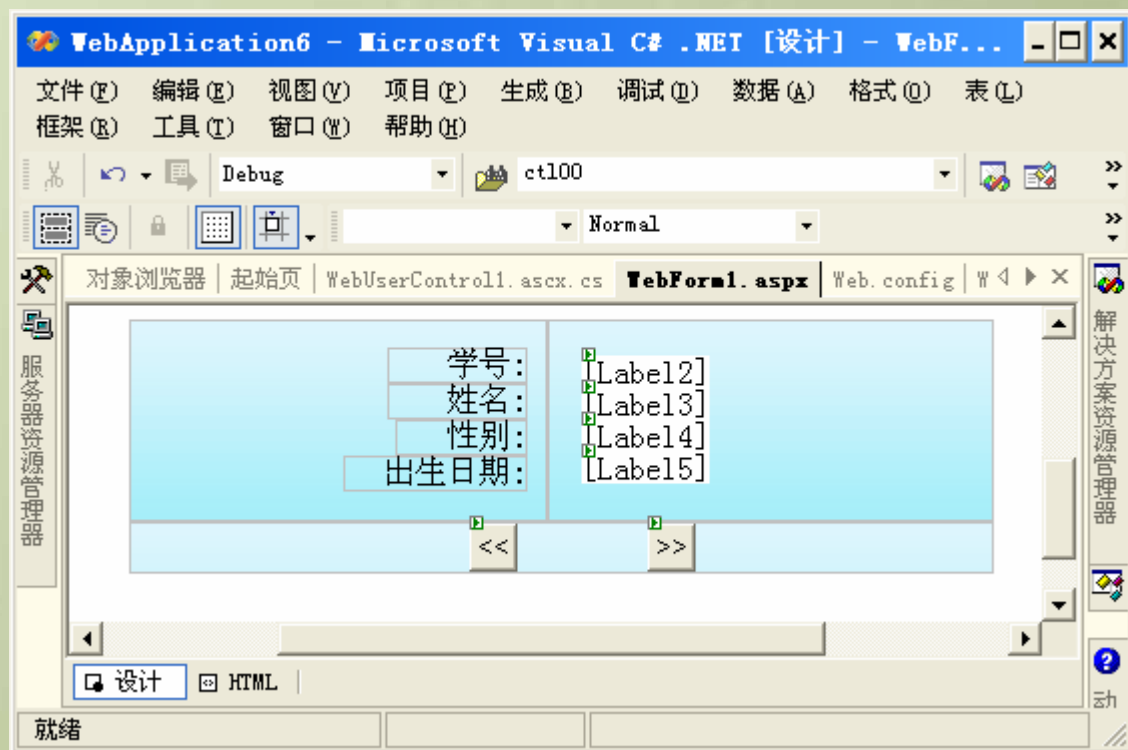
[项目一]利用数据集在网页上逐一显示学生记录，界面如下：通过导航按钮可显示不同记录。



实例分析

1、制作过程

1) 生成ASP应用程序，在WebForm1.aspx上生成主界面，设计主界面如下：



实例分析

添加以下主要控件：

| | |
|----------------|-----------------|
| HTML 表格控件 | 布局界面 |
| 四个HTML Label控件 | 显示字段标题(如学号、姓名等) |
| 四个WEB Label控件 | 显示字段值 |
| 两个WEB Button控件 | 导航按钮 |

各控件属性值参考主界面。

2) 打开web.config文件，在节点<configuration>添加如下配置节点<appSettings>，代表一个连接字符串：

```
<appSettings>
```

```
<add key="DBConn" value="Server=localhost;DataBase=stu;User ID  
= sa" />
```

```
</appSettings>
```

3) 添加WebForm1.aspx的后台代码：

(1) 在WebForm1.aspx的设计窗口下按F7功能键，打开后台代码文件WebForm1.aspx.cs，在类WebForm1中添加如下变量：



实例分析

```
private SqlConnection sqlConnection1=new  
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings[  
"DBConn"]);
```

```
private DataSet dataSet11;
```

```
private SqlDataAdapter sqlDataAdapter1;
```

（2）在WebForm1的Load事件处理方法中下加入如下代码：

```
private void Page_Load(object sender, System.EventArgs e)  
{  
    if(!this.IsPostBack)  
    {  
        sqlDataAdapter1=new SqlDataAdapter("select * from  
stu",this.sqlConnection1);  
        this.dataSet11=new DataSet();  
        sqlDataAdapter1.Fill(dataSet11,"stu");  
        ViewState["postion"]=0;  
        this.DataBind();  
    }  
}
```



实例分析

(3) 添加WEB窗体WebForm1的DataBinding事件处理方法（在WebForm1的设计窗口下，打开属性窗口的下拉列表框，选中WebForm1，然后在属性窗口中选中事件选项，在事件列表中找到DataBinding事件，在右栏双击即可）。

```
private void WebForm1_DataBinding(object sender, System.EventArgs e)
{
    sqlDataAdapter1=new SqlDataAdapter("select * from
stu",this.sqlConnection1);
    this.dataSet11=new DataSet();
    sqlDataAdapter1.Fill(dataSet11,"stu");
    int ind=(int)ViewState["postion"];
    if(ind<0) ViewState["postion"]=ind=0;
    else if(ind>=this.dataSet11.Tables["stu"].Rows.Count)
    ViewState["postion"]=ind=this.dataSet11.Tables["stu"].Rows.Count-1;
    Label2.Text=
DataBinder.Eval(this.dataSet11.Tables["stu"].DefaultView[ind],"xh").ToString();
    Label3.Text=
```



实例分析

```
DataBinder.Eval(this.dataSet11.Tables["stu"].DefaultView[ind],"xm").ToString();
Label4.Text=DataBinder.Eval(this.dataSet11.Tables["stu"].DefaultView[ind],"xb").ToString();
Label5.Text=DataBinder.Eval(this.dataSet11.Tables["stu"].DefaultView[ind],"csrq","{0:D}").ToString();
}
```

（4）生成按钮<<的Click事件处理方法（双击按钮即可）。

```
private void Button1_Click(object sender, System.EventArgs e)
{
    int ind=(int)ViewState["postion"];
    ViewState["postion"]/--ind;
    this.DataBind();
}
```

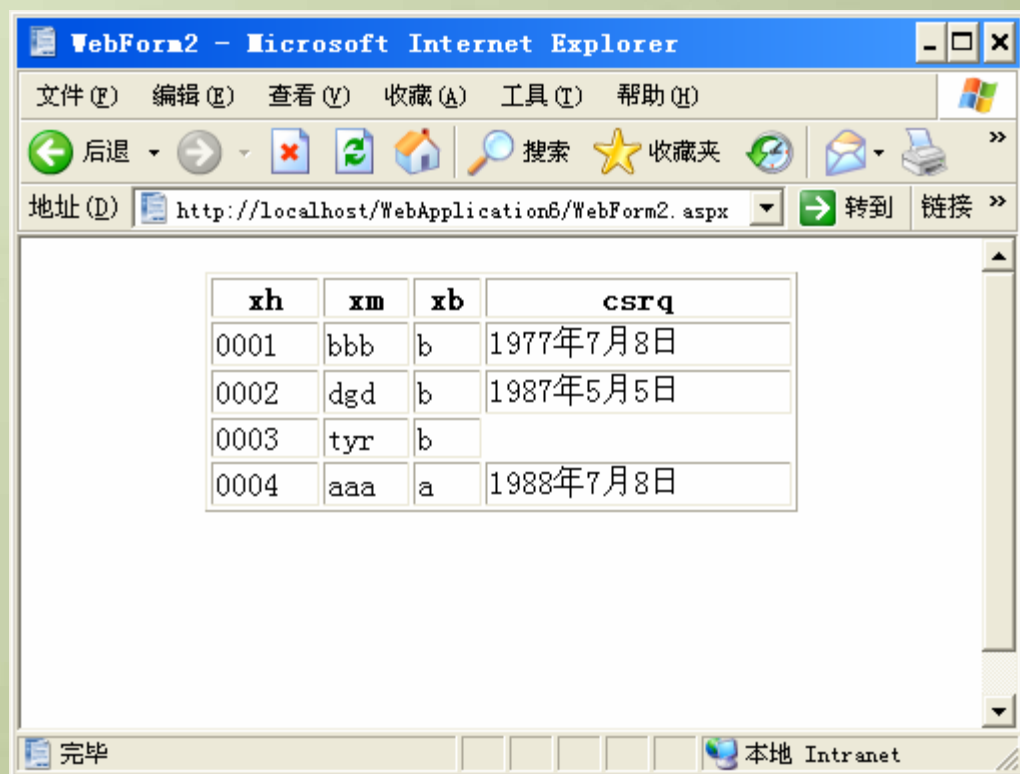
（5）生成按钮>>的Click事件处理方法（双击按钮即可）。

```
private void Button1_Click(object sender, System.EventArgs e)
{
    int ind=(int)ViewState["postion"];
    ViewState["postion"]=++ind;
    this.DataBind();
}
```



实例分析

[项目一]利用DataReader对象在网页上显示所有学生记录，界面如下：



实例分析

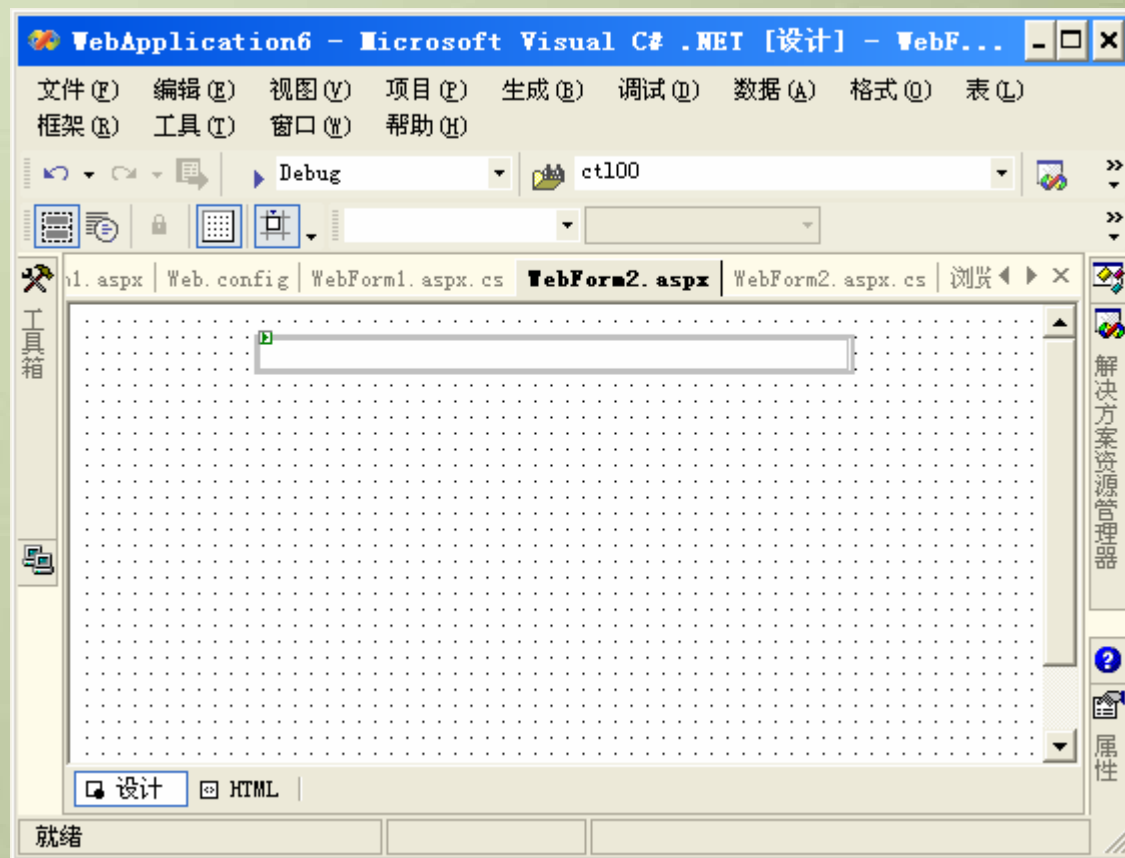
制作过程:

1) 设计界面

本界面较简单:

- 添加一个HTML表格，设置居中，起布局作用，且只留一个单元格。
- 在单元单中添加一个HTML 标签（Lable），点右键，选“作为服务器控件运行”选项。

这时将会在后台代码类中自动生成同该控件相对应的名为DIV1的变量。



实例分析

2) 添加代码

(1) 在设计窗体中按F7打开后台代码，在WEB窗体类中加入变量定义，定义一个数据库连接：

```
private SqlConnection sqlConnection1 = new  
    SqlConnection(System.Configuration.ConfigurationSettings.AppSettings["DBConn"]);
```

(2) 进入窗体的Load事件处理方法，加入事件代码，该方法代码如下：

```
private void Page_Load(object sender, System.EventArgs e)  
{  
    if(!this.IsPostBack)  
    {  
        SqlCommand cmd = new System.Data.SqlClient.SqlCommand(  
            "select * from stu",this.sqlConnection1);  
        this.sqlConnection1.Open();  
        SqlDataReader rd= cmd.ExecuteReader(  
            System.Data.CommandBehavior.CloseConnection);  
        string str="";
```



实例分析

```
str+="<table align=center width=100% border=1>";
str+="<tr>";
for(int i=0;i<rd.FieldCount;i++)
    str+="<th>"+rd.GetName(i)+"</th>";
str+="</tr>";
while(rd.Read())
{
    str+="<tr>";
    str+="<td>"+String.Format("{0}",rd["xh"])+ "</td>";
    str+="<td>"+String.Format("{0}",rd["xm"])+ "</td>";
    str+="<td>"+String.Format("{0}",rd["xb"])+ "</td>";
    str+="<td>"+String.Format("{0:D}",rd["csrq"])+ "</td>";
    str+="</tr>";
}
str+="</table>";
rd.Close();
DIV1.InnerHtml=str;
}
```



第六讲 WEB窗体数据绑定

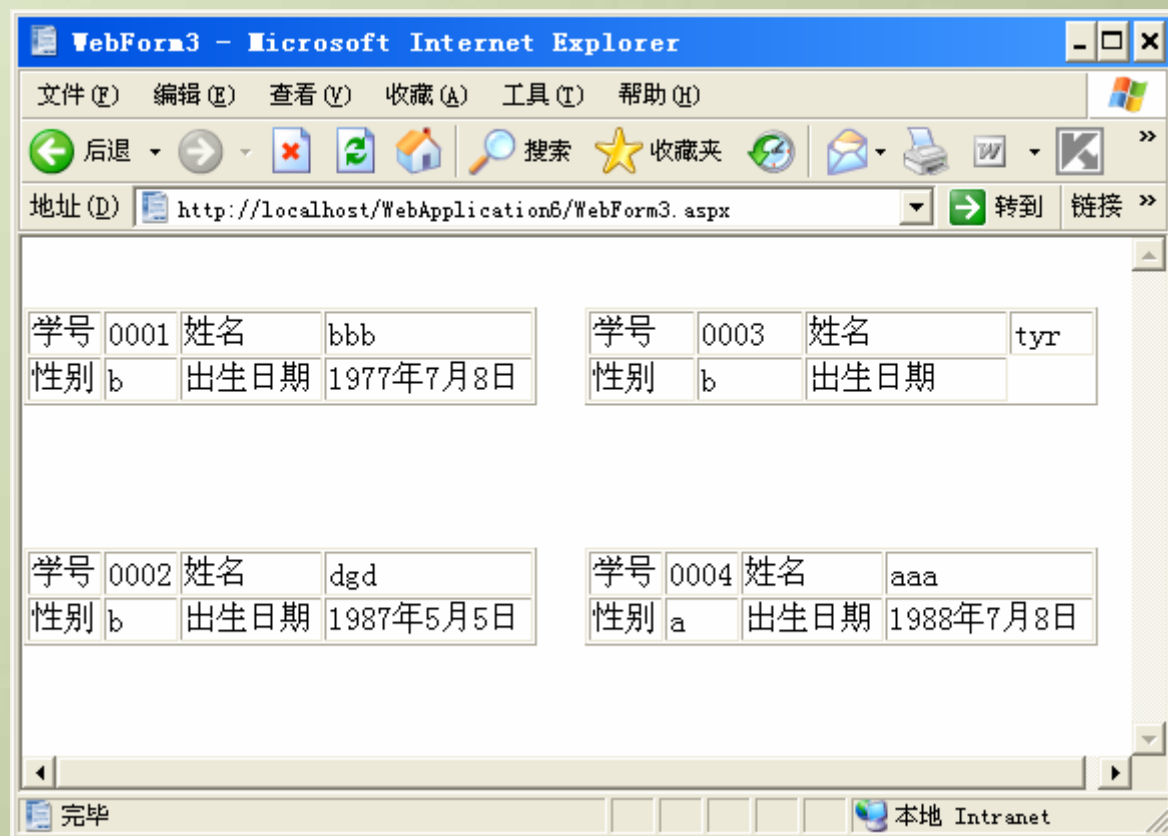
主要内容：

- 将数据绑定到DataList控件
- 将数据绑定到DataGrid控件
- 通过绑定控件对被绑定数据编辑



将数据绑定到DataList控件

[项目三]将数据表stu显示到DataList控件,显示界面如下:



The screenshot shows a Microsoft Internet Explorer window titled "WebForm3 - Microsoft Internet Explorer". The address bar displays "http://localhost/WebApplication8/WebForm3.aspx". The main content area contains four student records, each represented by a table with two rows: "学号" (Student ID) and "姓名" (Name) in the first row, and "性别" (Gender) and "出生日期" (Date of Birth) in the second row.

| | | | |
|----|------|------|-----------|
| 学号 | 0001 | 姓名 | bbb |
| 性别 | b | 出生日期 | 1977年7月8日 |

| | | | |
|----|------|------|-----|
| 学号 | 0003 | 姓名 | tyr |
| 性别 | b | 出生日期 | |

| | | | |
|----|------|------|-----------|
| 学号 | 0002 | 姓名 | dgd |
| 性别 | b | 出生日期 | 1987年5月5日 |

| | | | |
|----|------|------|-----------|
| 学号 | 0004 | 姓名 | aaa |
| 性别 | a | 出生日期 | 1988年7月8日 |

The status bar at the bottom shows "完毕" (Completed) and "本地 Intranet" (Local Intranet).

将数据绑定到DataList控件

操作过程:

1) 设计界面。

(1) 在WEB窗体中加入DataList控件。

(2) 在DataList控件点击右键，选择“编辑模板→项模板”。

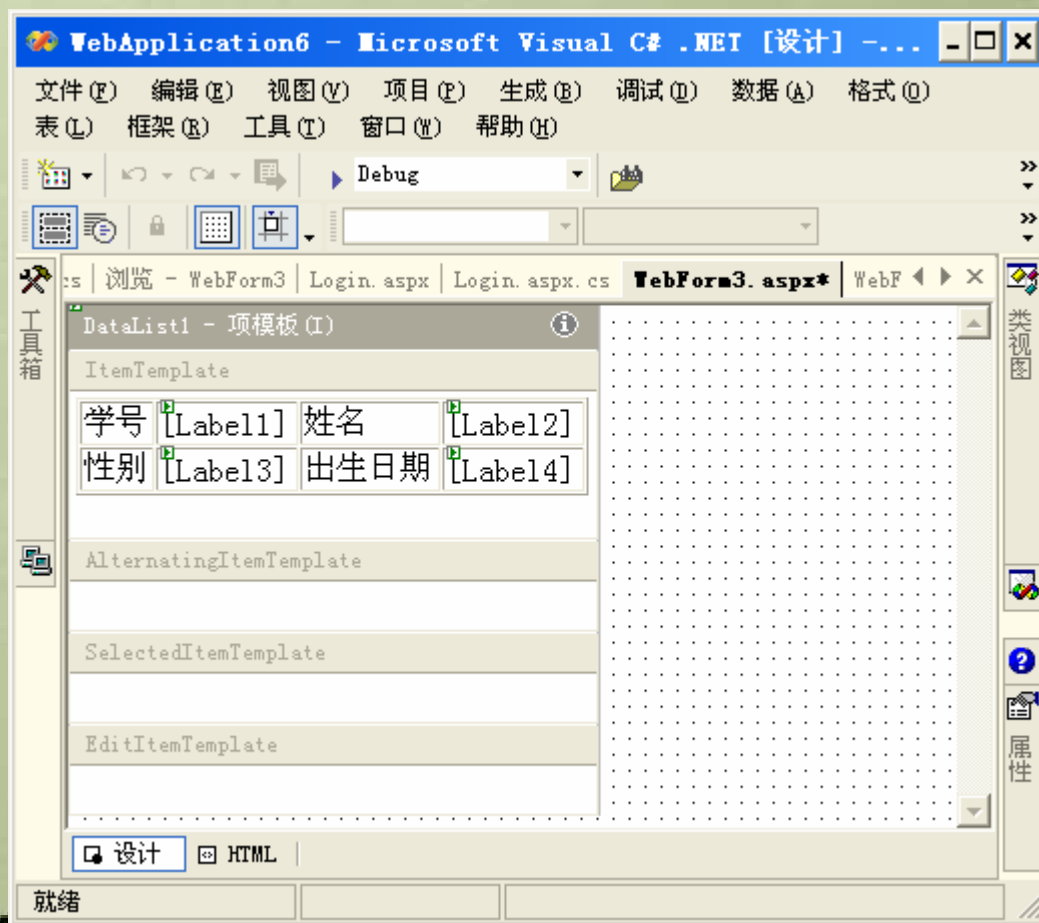
(3) 然后在项模板第一栏中加入右面内容:

- 先加入HTML表格，设为二行四列。

- 在指定位置直接输入标题（如学号等）。

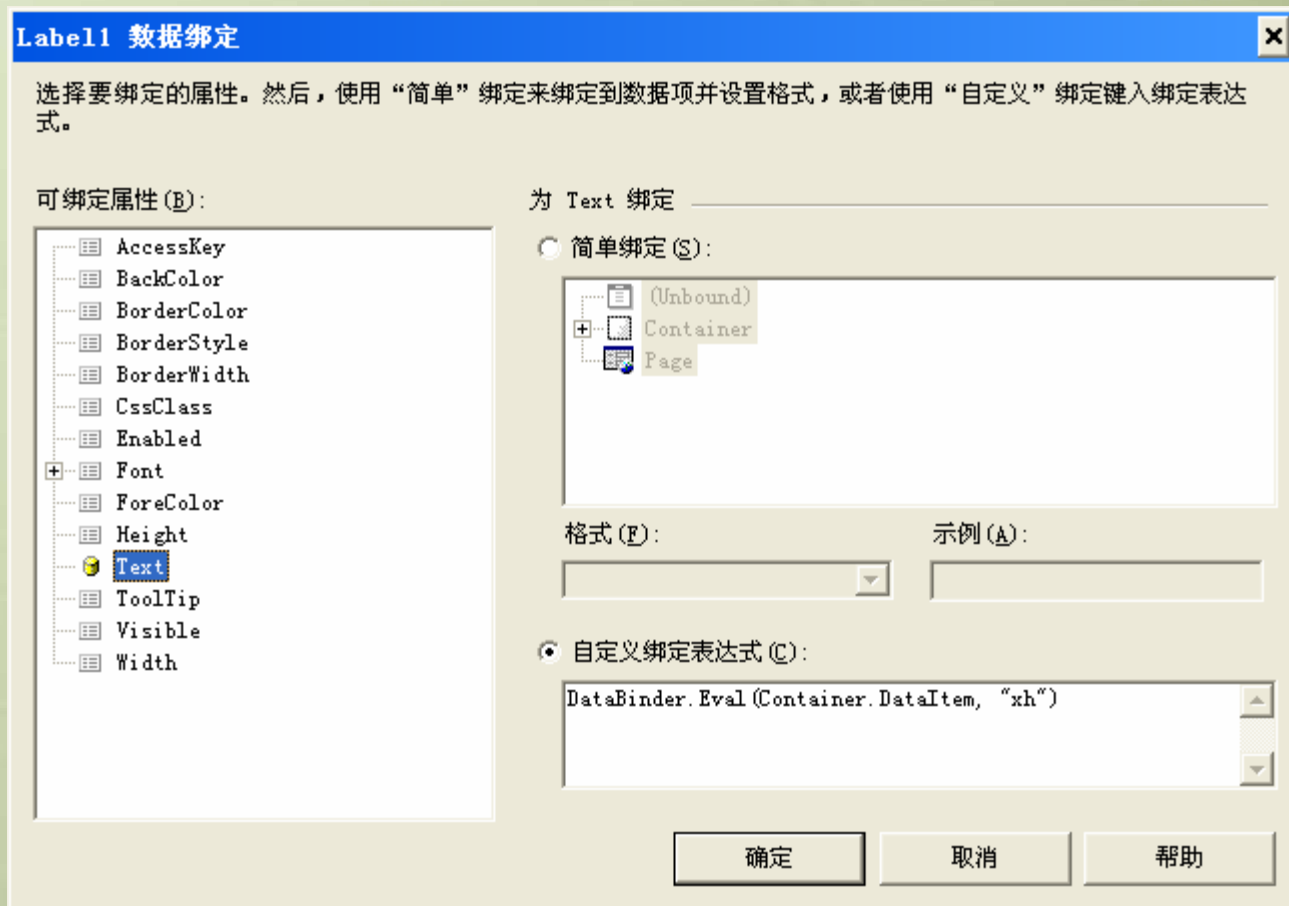
- 分别在指定位置输入四个WEB标签控件

设置四个Label控件的DataBindings属性，如下：



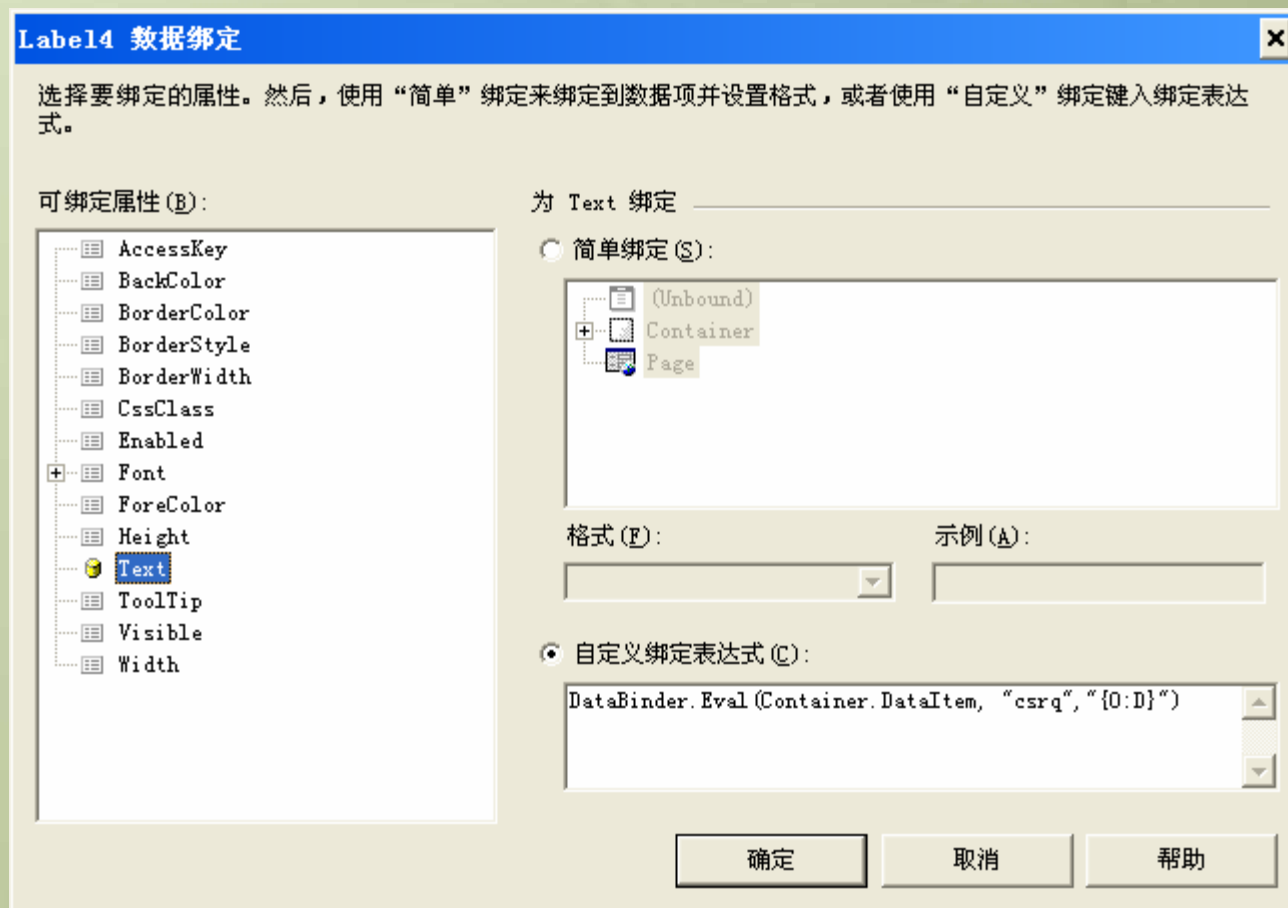
将数据绑定到DataList控件

学号对应的Label1设置：先选择左边的Text属性，然后在右面设置自定义表达式：`DataBinder.Eval(Container.DataItem, "xh")`



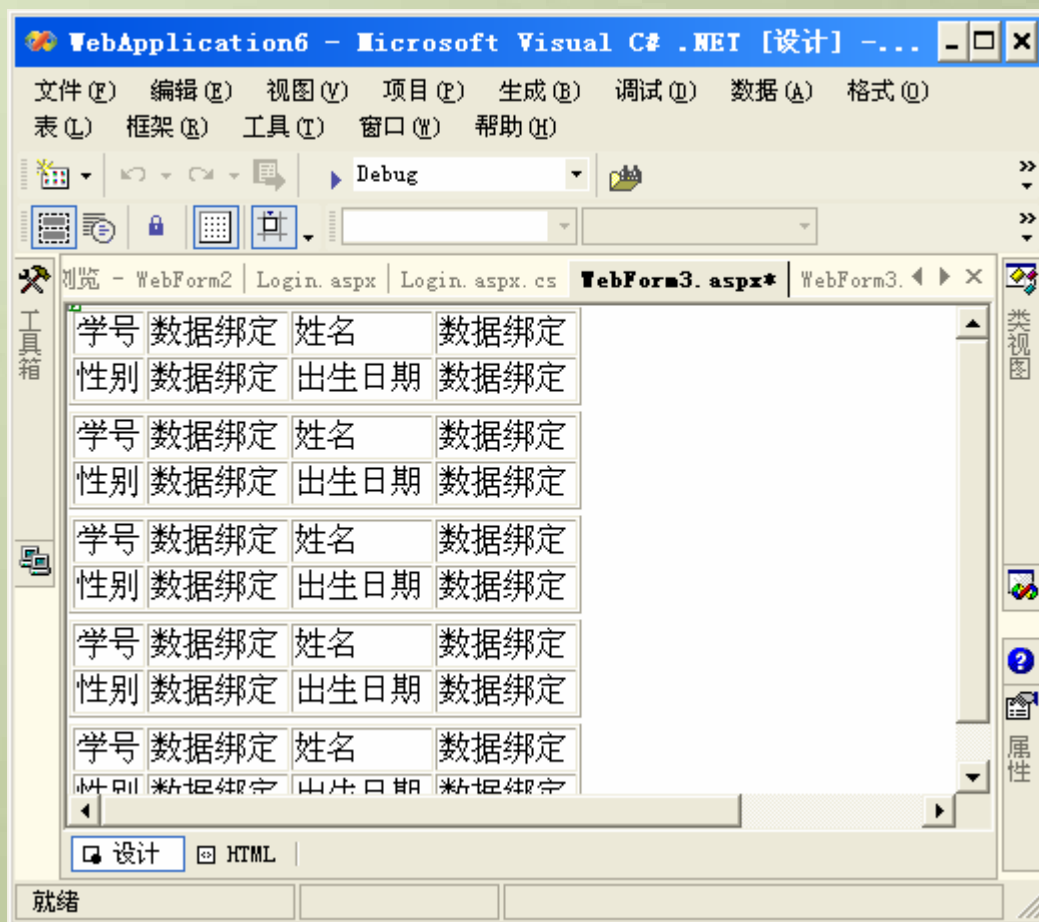
将数据绑定到DataList控件

其它Label绑定属性设置方法类似，只是修改绑定到的字段名，对于出生日期字段，可设置数据格式：`DataBinder.Eval(Container.DataItem, "csrq", "{0:D}")`其中{0:D}表示以长日期格式显示。



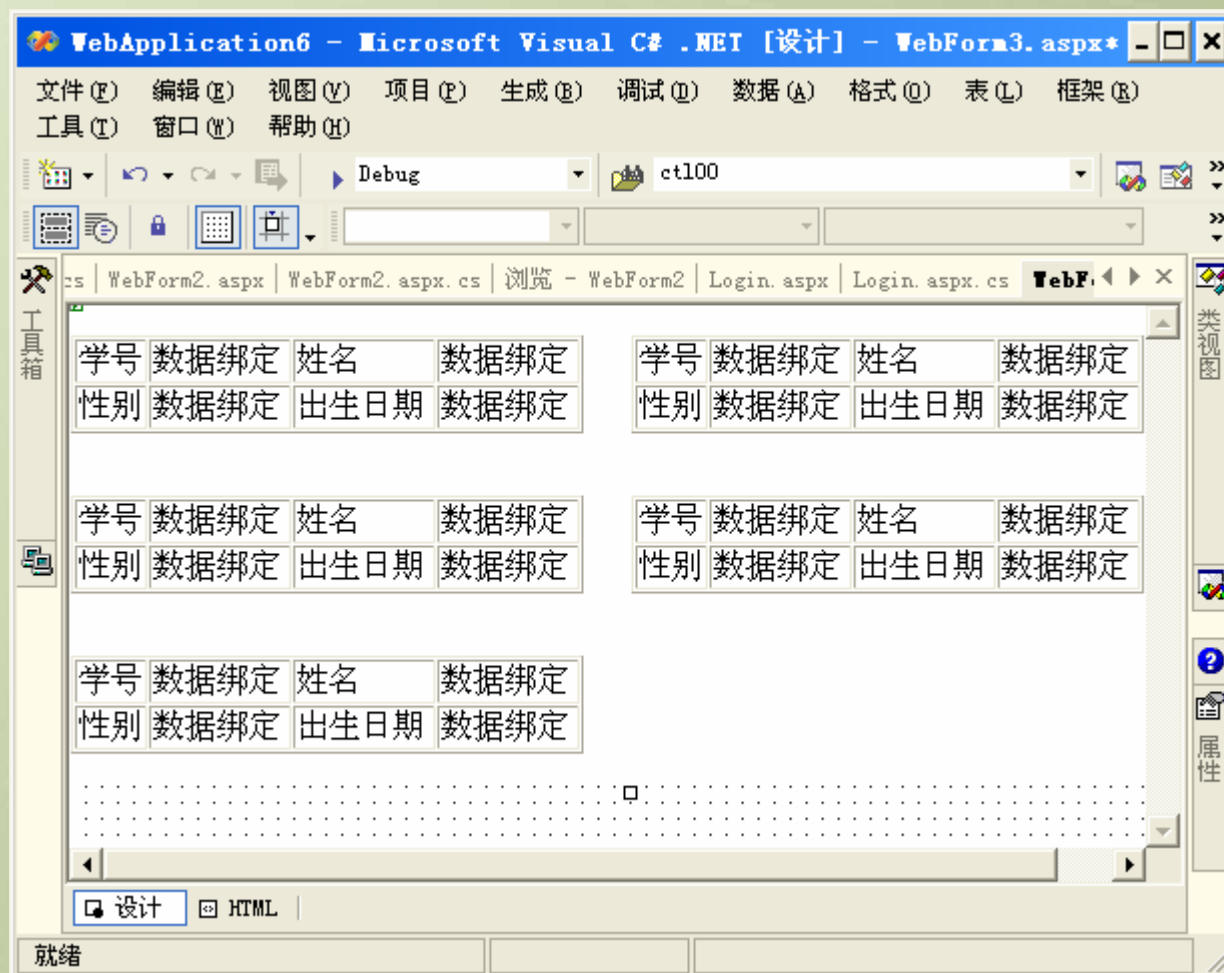
将数据绑定到DataList控件

然后回到项模板设计界面，点右建选择“结束模板编辑”，显示如下界面。



将数据绑定到DataList控件

设置DataList控件的RepeatColumn属性值为2列。则界面如下：



将数据绑定到DataList控件

2) 添加代码

(1) 在WEB窗体按F7打开后台代码，在窗体类中添加数据连接对象并初始化：

```
private SqlConnection sqlConnection1 = new  
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings["DBConn"]);
```

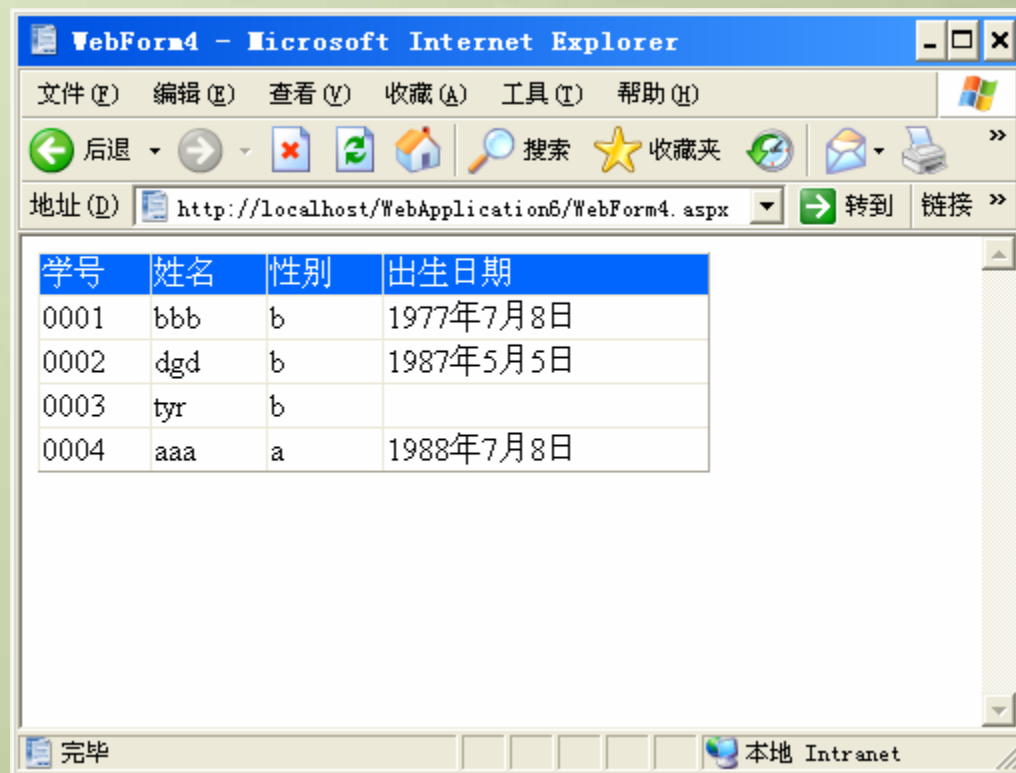
(2) 在窗体类中添加Load事件代码：

```
private void Page_Load(object sender, EventArgs e)  
{  
    if(!this.IsPostBack)  
    {  
        SqlCommand cmd = new System.Data.SqlClient.SqlCommand("select  
* from stu",this.sqlConnection1);  
        this.sqlConnection1.Open();  
        SqlDataReader rd=  
cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);  
        DataList1.DataSource=rd;  
        DataList1.DataBind();  
        rd.Close();  
    }  
}
```



将数据绑定到DataGrid控件

[项目四]将stu表绑定到DataGrid控件显示，效果如下：

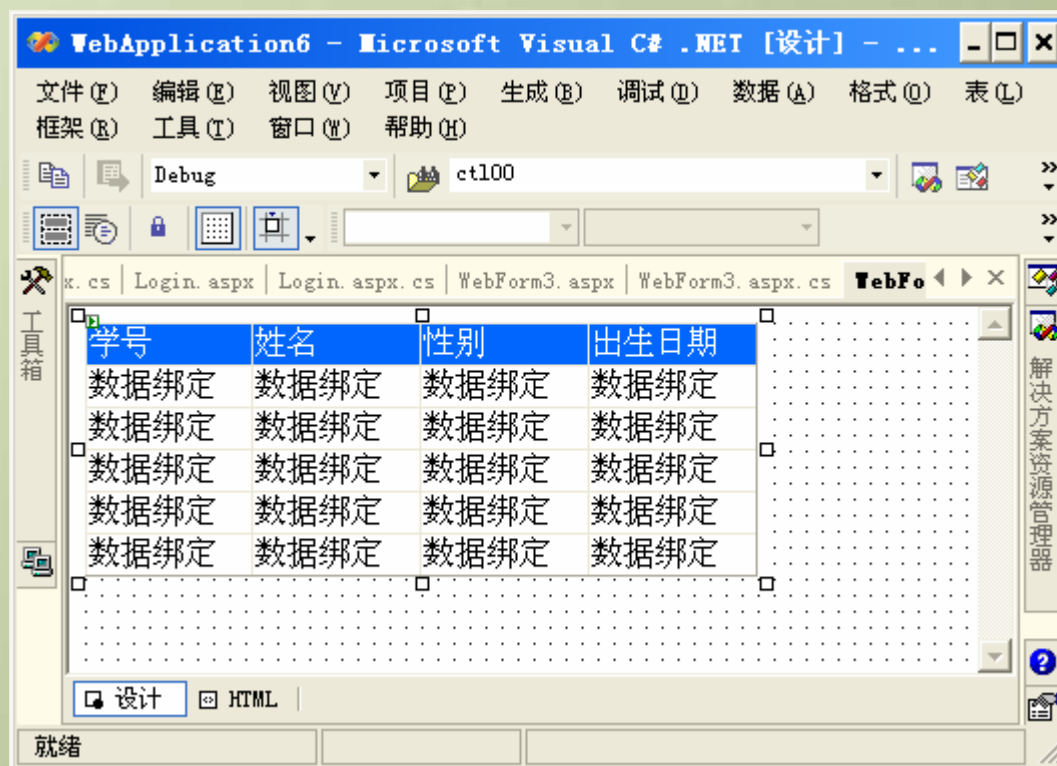


将数据绑定到DataGrid控件

制作过程:

1) 界面设计

(1) 在设计窗口添加一个DataGrid控件DataGrid1, 设置属性后如下图所示:

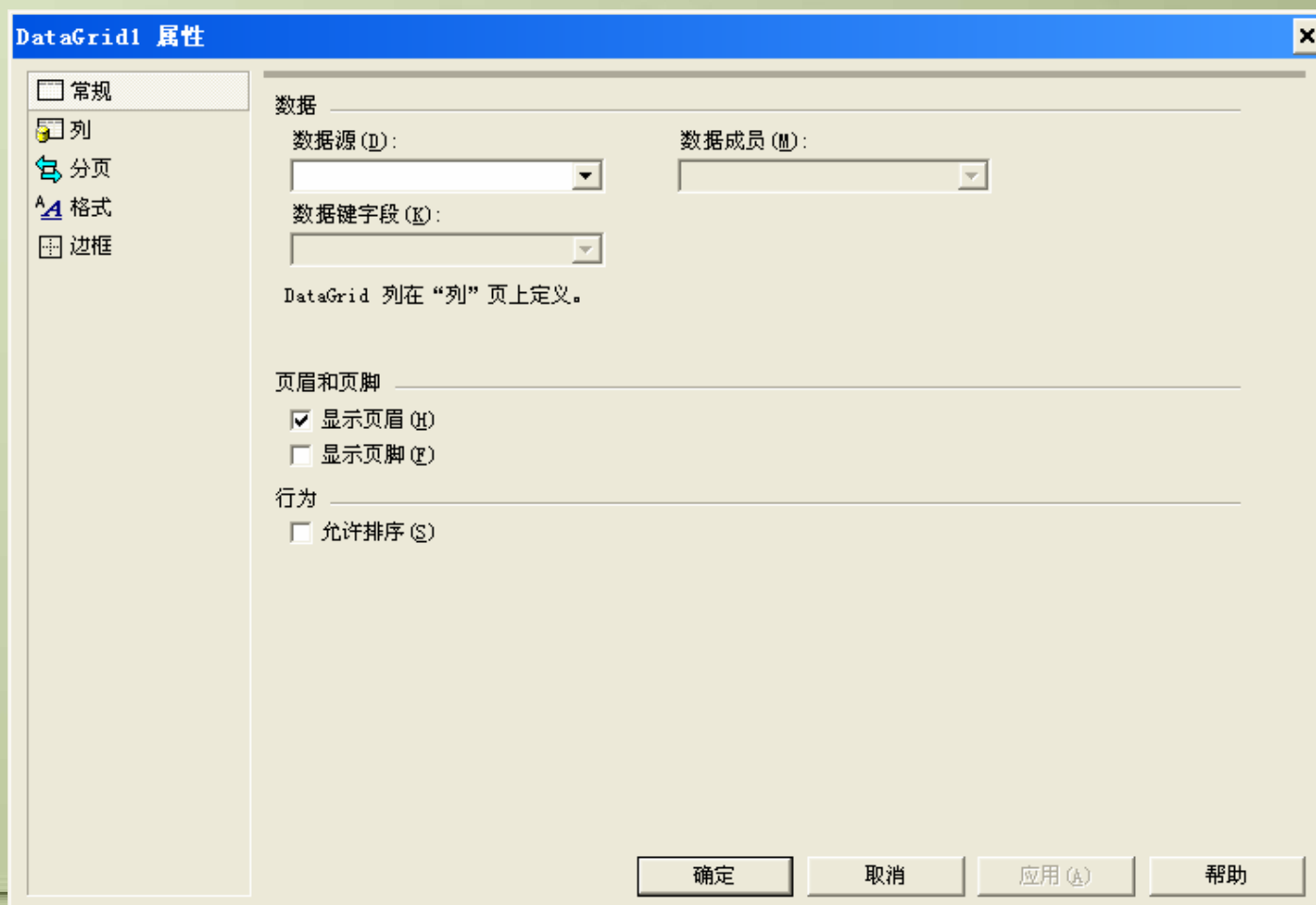


将数据绑定到DataGrid控件

DataGrid控件属性设置过程：

在新添加的DataGrid控件上点右键，在快捷菜单上选取“属性生成器”。

如下所示：



DataGrid1 属性

数据

数据源 (D):

数据成员 (M):

数据键字段 (K):

DataGrid 列在“列”页上定义。

页眉和页脚

☒ 显示页眉 (H)

☐ 显示页脚 (F)

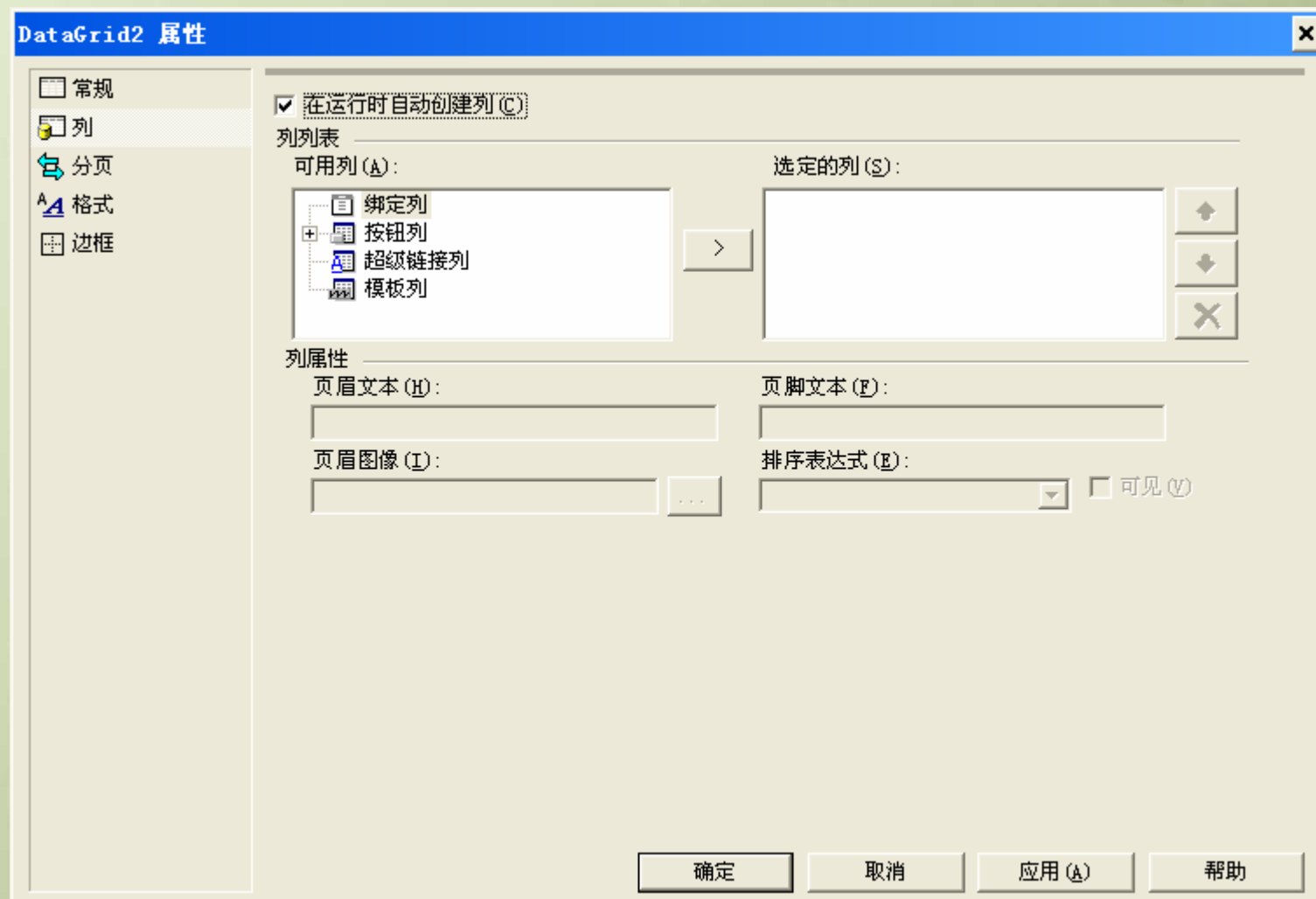
行为

☐ 允许排序 (S)

确定 取消 应用 (A) 帮助

将数据绑定到DataGrid控件

在左边的导航框中选择列：



将数据绑定到DataGrid控件

清除上面“运行时自动创建列”的勾选，并加入四个绑定列，如下图：

DataGrid2 属性

☐ 在运行时自动创建列 (C)

列列表

可用列 (A):

- 绑定列
- 按钮列
- 超级链接列
- 模板列

选定的列 (S):

- 绑定列
- 绑定列
- 绑定列
- 绑定列

BoundColumn 属性

页眉文本 (H):

页眉图像 (I):

数据字段 (D):

数据格式设置表达式 (T):

☐ 只读 (R)

页脚文本 (F):

排序表达式 (E):

☒ 可见 (V)

[将此列转换为模板列](#)

确定 取消 应用 (A) 帮助

将数据绑定到DataGrid控件

修改每个绑定列的属性，主要设置“页眉文本”和“数据字段”，以学号为例：

将数据绑定到DataGrid控件

对于出生日期，因为是日期时间型数据，所以最好指定显示格式，设置如下：

DataGrid1 属性

☐ 在运行时自动创建列 (C)

列列表

可用列 (A):

- ☐ 绑定列
- ☐ 按钮列
- ☐ 超级链接列
- ☐ 模板列

选定的列 (S):

- ☐ 学号
- ☐ 姓名
- ☐ 性别
- ☒ 出生日期

BoundColumn 属性

页眉文本 (H):

出生日期

页眉图像 (I):

...

数据字段 (D):

csrq

数据格式设置表达式 (T):

{0:D}

☐ 只读 (R)

页脚文本 (F):

排序表达式 (E):

可见 (V) ☒

[将此列转换为模板列](#)

确定 取消 应用 (A) 帮助

将数据绑定到DataGrid控件


2) 添加代码

(1) 在设计窗体按F7打开后台代码，在WEB窗体类中添加数据连接字段并初始化：

```
private SqlConnection sqlConnection1 = new  
SqlConnection(System.Configuration.ConfigurationSettings.AppSettings["DBConn"]  
);
```

(2) 打开窗体Load事件处理方法，添加代码如下：

```
private void Page_Load(object sender, System.EventArgs e)  
{ if(!this.IsPostBack)  
{ SqlCommand cmd = new System.Data.SqlClient.SqlCommand("select * from  
stu",this.sqlConnection1);  
this.sqlConnection1.Open();  
SqlDataReader rd= cmd.ExecuteReader(  
System.Data.CommandBehavior.CloseConnection);  
DataGrid1.DataSource=rd;  
DataGrid1.DataBind();  
rd.Close();}  
}
```



通过绑定控件对被绑定数据编辑

[项目五]通过DataList控件实现对数据表的修改,主界面如下,在项目三基础上另加链接按钮LinkButton, 分别可进行编辑和删除操作, 当单击编辑链接时, 出

出编辑界面（下页）。
当单击删除链接时,
删除指定记录。

WebForm3 - Microsoft Internet Explorer

文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)

后退 前进 停止 刷新 主页 搜索 收藏夹 打印 邮件 地址(D) http://localhost/WebApplication8/WebForm3.aspx 转到 链接 >>

| | | | |
|----|------|--------------------|--------------------|
| 学号 | 0001 | 姓名 | aaaa |
| 性别 | b | 出生日期 | 1966年8月9日 |
| | | 编辑 | 删除 |

| | | | |
|----|------|--------------------|--------------------|
| 学号 | 0002 | 姓名 | bbbb |
| 性别 | b | 出生日期 | 1999年9月9日 |
| | | 编辑 | 删除 |

| | | | |
|----|------|--------------------|--------------------|
| 学号 | 0003 | 姓名 | cccc |
| 性别 | a | 出生日期 | 1987年9月9日 |
| | | 编辑 | 删除 |

| | | | |
|----|------|--------------------|--------------------|
| 学号 | 0004 | 姓名 | dddd |
| 性别 | b | 出生日期 | 1978年6月6日 |
| | | 编辑 | 删除 |

本地 Intranet

通过绑定控件对被绑定数据编辑

下图为编辑界面，编辑记录行下的链接按钮变为取消和更新，记录行的段字显示标签变为可输入数据的编辑框，单击更按钮可以更新修改后的内容。

WebForm3 - Microsoft Internet Explorer

文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)

后退 前进 停止 刷新 搜索 收藏夹 打印 邮件 地址(D) http://localhost/WebApplication6/WebForm3.aspx 转到 链接

| | | | | | | | |
|---------------------------------------|------|------|----------|---------------------------------------|------|------|-----------|
| 学号 | 0001 | 姓名 | aaaa | 学号 | 0002 | 姓名 | bbbb |
| 性别 | b | 出生日期 | 1966-8-9 | 性别 | b | 出生日期 | 1999年9月9日 |
| 取消 更新 | | | | 编辑 删除 | | | |

| | | | | | | | |
|---------------------------------------|------|------|-----------|---------------------------------------|------|------|-----------|
| 学号 | 0003 | 姓名 | cccc | 学号 | 0004 | 姓名 | dddd |
| 性别 | a | 出生日期 | 1987年9月9日 | 性别 | b | 出生日期 | 1978年6月6日 |
| 编辑 删除 | | | | 编辑 删除 | | | |

本地 Intranet

通过绑定控件对被绑定数据编辑

项目制作过程：

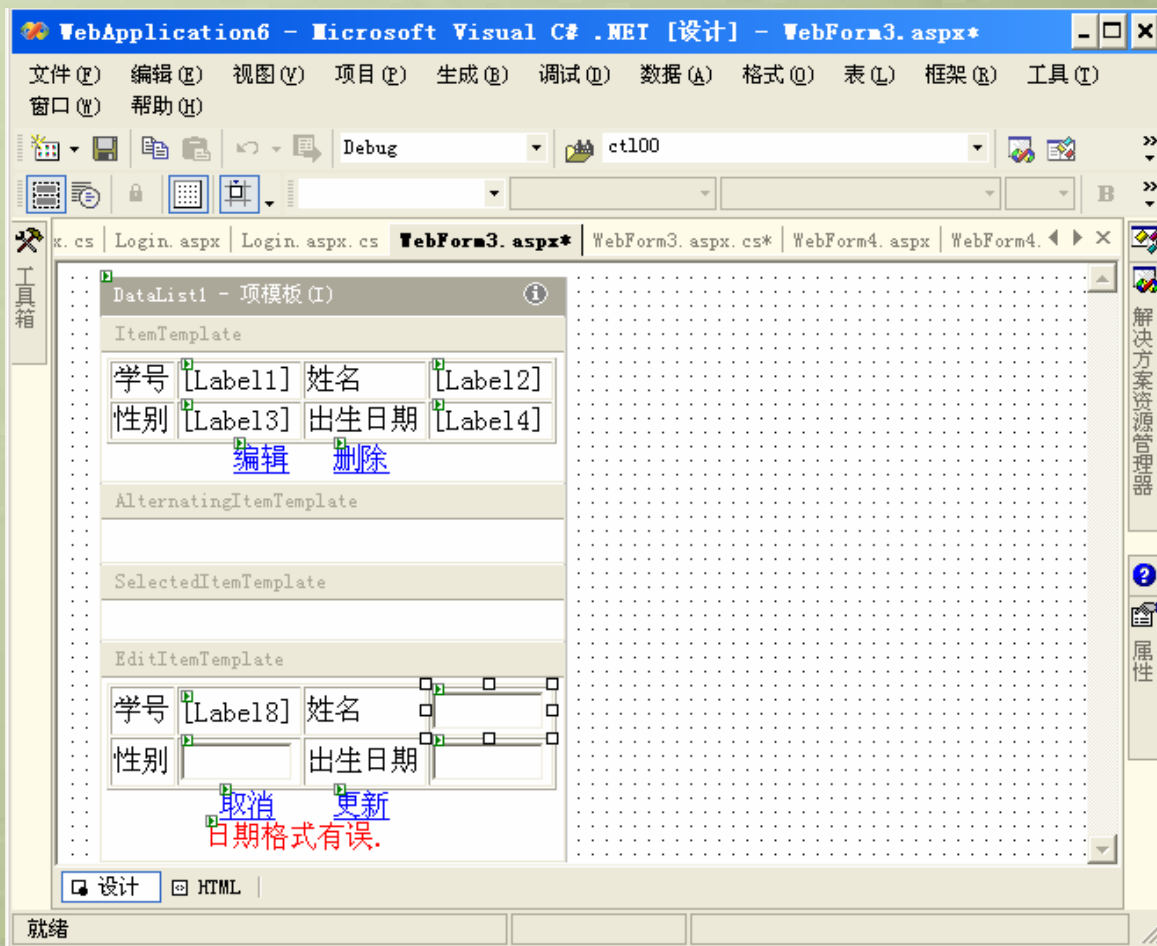
本项目在项目三界面基础上作继续如下操作：

1、界面修改

1) 打开项模板，从工具箱中添加两个

LinkButton控件到原ItemTemplate栏中记录行下，分别将标题改为编辑、删除（如右图）。

2) 将ItemTemplate栏中所有内容拷贝到EditItemTemplate栏中，并将除显示学号外的三个Web标签改为TextBox，将链接按钮标题改为取消和更新。



通过绑定控件对被绑定数据编辑

3) 在取消和更新链接按钮下添加一个Web比较验证控件CompareValidator1。

2、新添加控件的属性修改

| 控件 | 属性 | 属性值 | 说明 |
|----------|-------------------|---|-------------|
| 编辑链接按钮 | CommandName | edit | 发出编辑命令 |
| 删除链接按钮 | CommandName | delete | 发出删除命令 |
| 取消链接按钮 | CommandName | cancel | 发出取消命令 |
| 更新链接按钮 | CommandName | update | 发出取消命令 |
| 姓名编辑框 | 自定义绑定表达式 | DataBinder.Eval(Container.DataItem, "xm") | |
| 性别编辑框 | 自定义绑定表达式 | DataBinder.Eval(Container.DataItem, "xb") | |
| 出生日期编辑框 | 自定义绑定表达式 | DataBinder.Eval(Container.DataItem, "csrq","{0:d}") | |
| 比较验证控件 | ControlToValidate | TextBox3 (出生日期编辑框) | |
| | Display | Dynamic | 动态显示 |
| | ErrorMessage | 日期格式有误 | 日期验证不通过时的提示 |
| | Operator | DataTypeCheck | 数据类型比较验证操作 |
| | Type | Date | 验证目标类型为日期型 |
| DataList | DataKeyField | xh | 用xh字段真充键集 |

注意：上面各编辑框的自定义绑定表达式皆选择绑定到Text属性。



通过绑定控件对被绑定数据编辑

3、代码添加

1) 添加一个自定义方法，用以将数据绑定到DataList控件：

```
private void BindList()
{
    SqlCommand cmd = new System.Data.SqlClient.SqlCommand("select * from
stu",this.sqlConnection1);
    this.sqlConnection1.Open();
    SqlDataReader rd=
cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    DataList1.DataSource=rd;
    DataList1.DataBind();
    rd.Close();
}
```

2) 将项目三中Load事件代码改为：

```
private void Page_Load(object sender, System.EventArgs e)
{
    if(!this.IsPostBack)        this.BindList();
}
```



通过绑定控件对被绑定数据编辑

3、代码添加

3) 选择DataList控件，在属性窗口的事件列表中选择EditCommand（编辑命令）事件，编辑事件处理方法：

```
private void DataList1_EditCommand(object source,  
System.Web.UI.WebControls.DataListCommandEventArgs e)  
{  
    DataList1.EditItemIndex=e.Item.ItemIndex;  
    this.BindList()    ;  
}
```

4) 用上面的操作办法，生成DataList控件的CancelCommand（取消命令）事件处理方法：

```
private void DataList1_CancelCommand(object source,  
System.Web.UI.WebControls.DataListCommandEventArgs e)  
{  
    DataList1.EditItemIndex=-1;  
    this.BindList();  
}
```



通过绑定控件对被绑定数据编辑

3、代码添加

5) 生成DataList控件的UpdateCommand（更新命令）事件处理方法：

```
private void DataList1_UpdateCommand(object source,  
System.Web.UI.WebControls.DataListCommandEventArgs e)  
{  
    TextBox txt=e.Item.FindControl("TextBox1") as TextBox;  
    string xm=txt.Text;  
    txt=e.Item.FindControl("TextBox2") as TextBox;  
    string xb=txt.Text;  
    txt=e.Item.FindControl("TextBox3") as TextBox;  
    string csrq=txt.Text;  
    string str=@"update stu set xm='{0}',xb='{1}',csrq='{2}' where xh='{3}'";  
    str = string.Format(str,xm,xb,csrq,DataList1.DataKeys[e.Item.ItemIndex]);  
    SqlCommand cmd = new System.Data.SqlClient.SqlCommand(str,this.sqlConnection1);  
    this.sqlConnection1.Open();  
    cmd.ExecuteNonQuery();  
    this.sqlConnection1.Close();  
    DataList1.EditItemIndex=-1;  
    this.BindList();  
}
```



通过绑定控件对被绑定数据编辑

3、代码添加

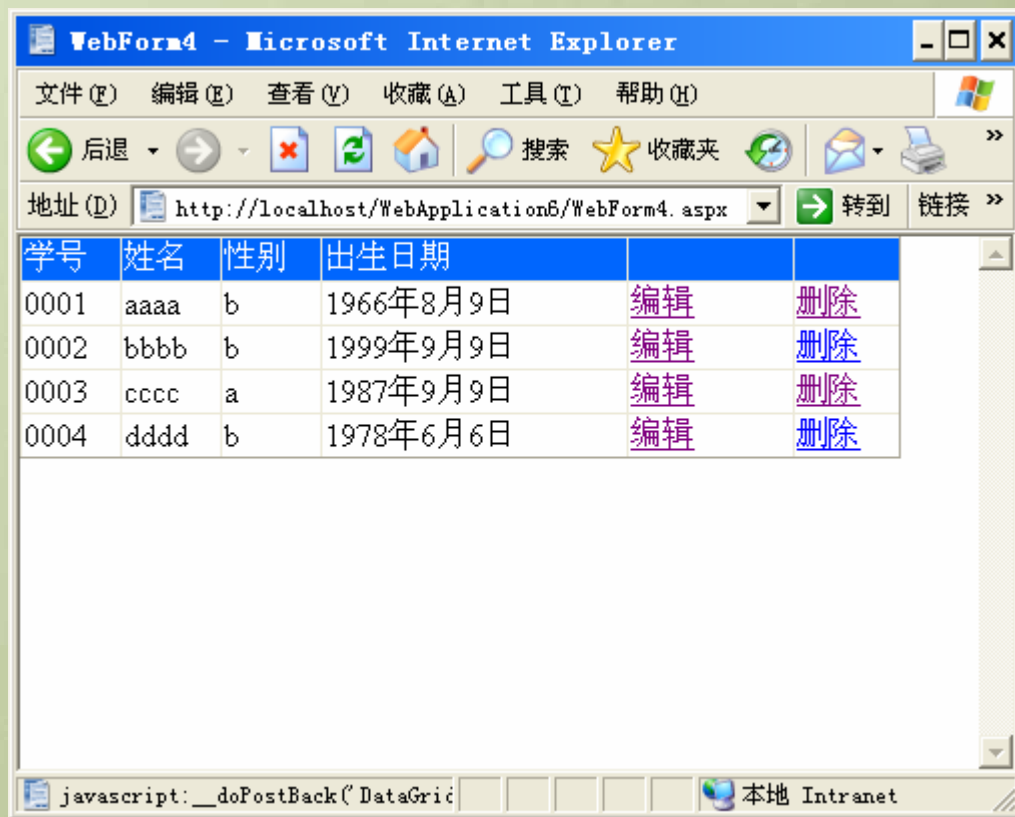
6) 生成DataList控件的DeleteCommand（删除命令）事件处理方法：

```
private void DataList1_DeleteCommand(object source,  
System.Web.UI.WebControls.DataListCommandEventArgs e)  
{  
    string str=@"delete from stu where xh='{0}'";  
    str = string.Format(str,DataList1.DataKeys[e.Item.ItemIndex]);  
    SqlCommand cmd = new  
System.Data.SqlClient.SqlCommand(str,this.sqlConnection1);  
    this.sqlConnection1.Open();  
    cmd.ExecuteNonQuery();  
    this.sqlConnection1.Close();  
    this.BindList();  
}
```



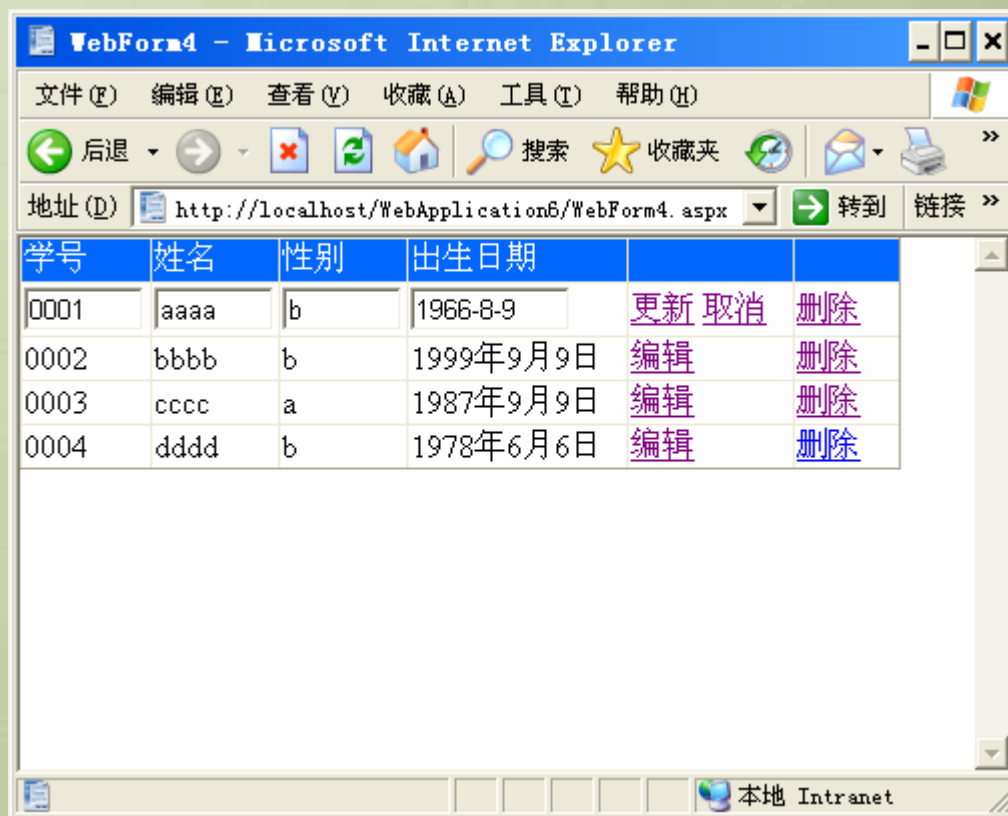
通过绑定控件对被绑定数据编辑

[项目五]通过DataGrid控件实现对数据表的修改,主界面如下,在项目四基础上另外添加两操作列:编辑和删除,单击某行编辑链接后出现编辑界面(下页),单删除链接后删除该行。



通过绑定控件对被绑定数据编辑

下图为编辑界面，编辑记录行下的链接按钮变为取消和更新，记录行的段字显示标签变为可输入数据的编辑框，单击更按钮可以更新修改后的内容。



通过绑定控件对被绑定数据编辑

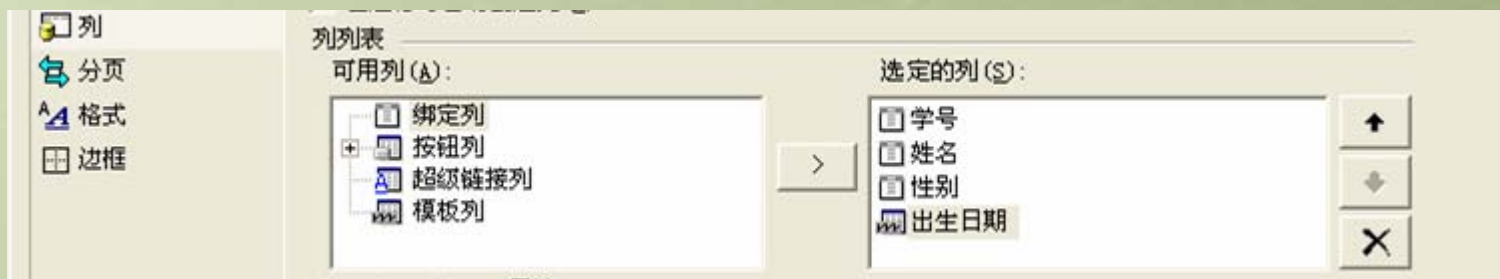
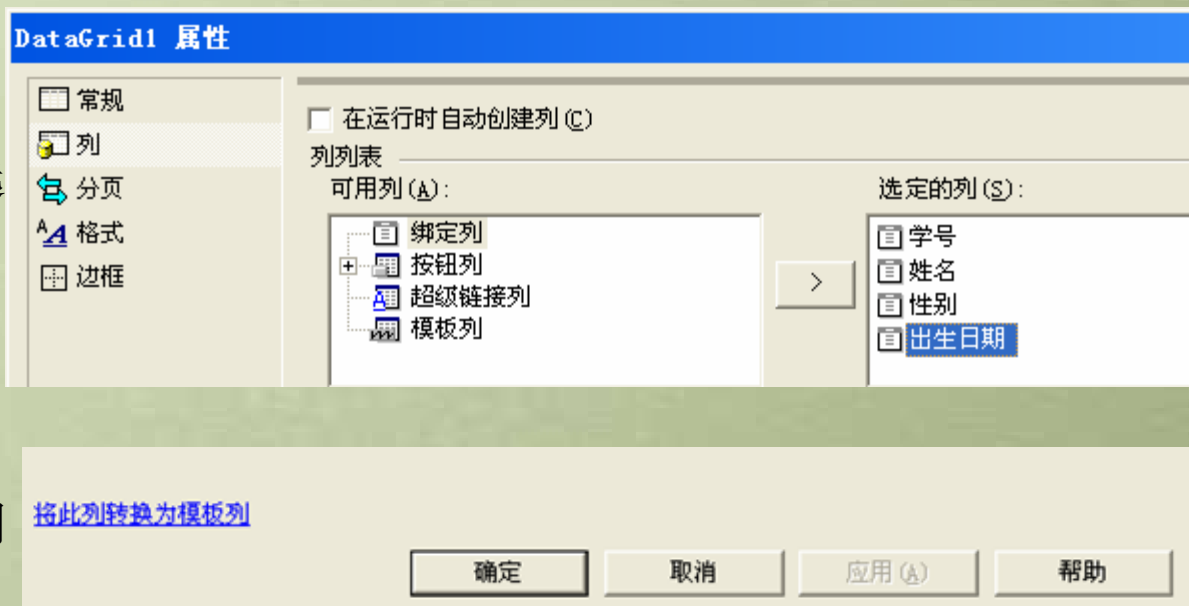
项目制作过程：

本项目在项目三界面基础上作继续如下操作：

1、界面修改

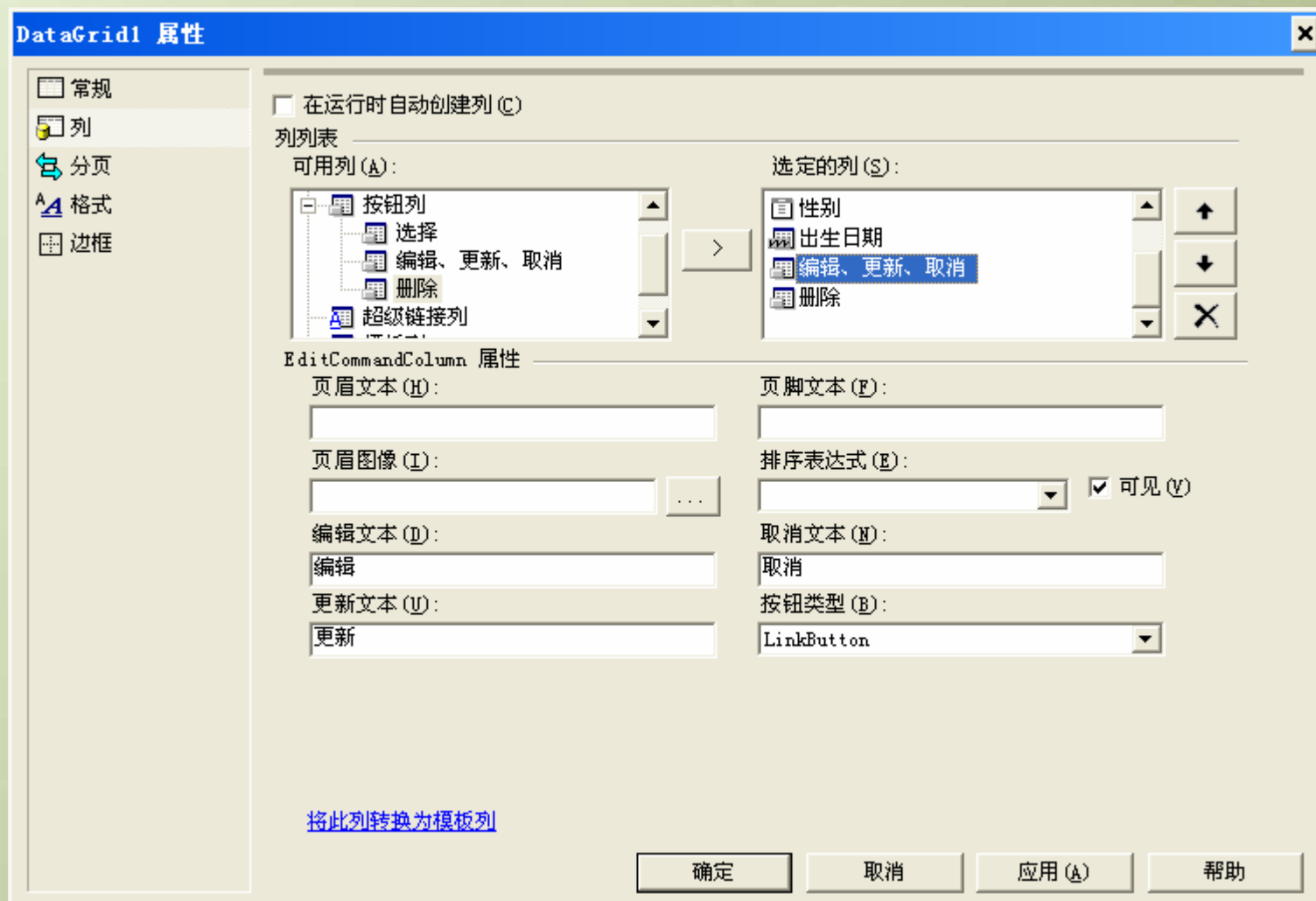
1) 在项目三界面基础上打开属性生成器，选择出生日期列（右），单击对话框下端的“将此列转化为模板列”提示项。

则对话框上的出生日期列显示为下图：



通过绑定控件对被绑定数据编辑

2) 再添加两个按钮列，如下图：



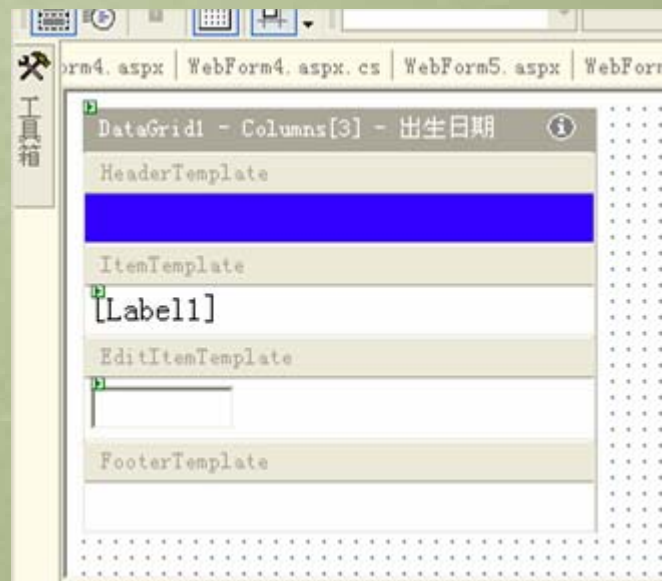
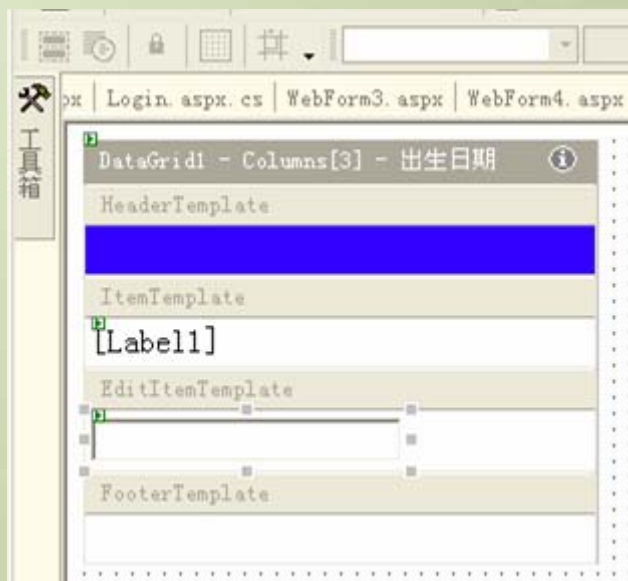
通过绑定控件对被绑定数据编辑

经过上面两步，界面变为右图所示。

3) 在网格控件弹出捷菜单，选择“编辑模板→出生日期”对出生日期模板列编辑，如下图所示。

然后将EditItemTemplate下的文本框适当调短，如下图。

| 学号 | 姓名 | 性别 | 出生日期 | | |
|------|------|------|------|--------------------|--------------------|
| 数据绑定 | 数据绑定 | 数据绑定 | 数据绑定 | 编辑 | 删除 |
| 数据绑定 | 数据绑定 | 数据绑定 | 数据绑定 | 编辑 | 删除 |
| 数据绑定 | 数据绑定 | 数据绑定 | 数据绑定 | 编辑 | 删除 |
| 数据绑定 | 数据绑定 | 数据绑定 | 数据绑定 | 编辑 | 删除 |
| 数据绑定 | 数据绑定 | 数据绑定 | 数据绑定 | 编辑 | 删除 |



通过绑定控件对被绑定数据编辑

4) 选择文本框, 通过数据绑定窗口将绑定日期格式设为短日期, 如下图所示:



5) 点右键结束模板编辑, 选择数据网格控件, 在属性窗口将DataKeyField设为xh。

2、编写代码

1) 定义自定义方法BindToGrid, 用以绑定网格控件:

```
void BindToGrid()
```

```
{  
    SqlDataAdapter ad = new SqlDataAdapter("select * from stu",this.sqlConnection1);  
    DataSet ds = new DataSet();  
    ad.Fill(ds,"stu");  
    DataGrid1.DataSource = ds;  
    DataGrid1.DataBind();  
}
```



通过绑定控件对被绑定数据编辑

2) 修改窗体Load事件处理方法:

```
private void Page_Load(object sender, System.EventArgs e)
{
    if(!this.IsPostBack)
        BindToGrid();
}
```

3) 添加网格控件的EditCommand事件处理方法:

```
private void DataGrid1_EditCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
    DataGrid1.EditItemIndex = e.Item.ItemIndex;
    this.BindToGrid();
}
```

4) 添加网格控件的CancelCommand事件处理方法:

```
private void DataGrid1_CancelCommand(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
    DataGrid1.EditItemIndex = -1;
    this.BindToGrid();
}
```



通过绑定控件对被绑定数据编辑

此时，可运行程序，如右图：
当点编辑按钮时，出现编辑
界面如下：

| 学号 | 姓名 | 性别 | 出生日期 | | |
|------|------|----|-----------|--------------------|--------------------|
| 0001 | aaaa | b | 1966年8月9日 | 编辑 | 删除 |
| 0002 | bbbb | b | 1999年9月9日 | 编辑 | 删除 |
| 0003 | cccc | a | 1987年9月9日 | 编辑 | 删除 |
| 0004 | dddd | b | 1978年6月6日 | 编辑 | 删除 |

| 学号 | 姓名 | 性别 | 出生日期 | | |
|------|------|----|-----------|--------------------|--------------------|
| 0001 | aaaa | b | 1966-8-9 | 更新 | 删除 |
| 0002 | bbbb | b | 1999年9月9日 | 编辑 | 删除 |
| 0003 | cccc | a | 1987年9月9日 | 编辑 | 删除 |
| 0004 | dddd | b | 1978年6月6日 | 编辑 | 删除 |

发现默认的文本编辑框太长，可用下面第五步添加网格控件的ItemCreated事件处理方法调整：

5) 添加网格控件的ItemCreated事件处理方法：



通过绑定控件对被绑定数据编辑

```
private void DataGrid1_ItemCreated(object sender,  
System.Web.UI.WebControls.DataGridItemEventArgs e)  
{  
    if(e.Item.ItemType==ListItemType.EditItem)  
    {  
        for(int i=0;i<3;i++)  
            (e.Item.Cells[i].Controls[0] as TextBox).Width=50;  
    }  
}
```

6) 添加更新命令UpdateCommand事件处理方法:

```
private void DataGrid1_UpdateCommand(object source,  
System.Web.UI.WebControls.DataGridCommandEventArgs e)  
{  
    SqlCommand cmd = new SqlCommand("update stu set  
xm=@xm,xb=@xb,csrq=@csrq where xh=@xh",sqlConnection1);  
    string xm=(e.Item.Cells[1].Controls[0] as TextBox).Text;  
    string xb=(e.Item.Cells[2].Controls[0] as TextBox).Text;  
    string csrq=(e.Item.FindControl("TextBox1") as TextBox).Text;  
    string xh = DataGrid1.DataKeys[e.Item.ItemIndex].ToString();
```



通过绑定控件对被绑定数据编辑

```
cmd.Parameters.Add("@xm",xm);  
cmd.Parameters.Add("@xb",xb);  
cmd.Parameters.Add("@csrq",csrq);  
cmd.Parameters.Add("@xh",xh);  
sqlConnection1.Open();      cmd.ExecuteNonQuery();  
sqlConnection1.Close();  
DataGrid1.EditItemIndex =-1;      this.BindToGrid();
```

```
}
```

7) 最后添加删除命令DeleteCommand事件处理方法:

```
private void DataGrid1_DeleteCommand(object source,  
System.Web.UI.WebControls.DataGridCommandEventArgs e)  
{  
    SqlCommand cmd = new SqlCommand("delete from stu where  
xh=@xh",sqlConnection1);  
    string xh = DataGrid1.DataKeys[e.Item.ItemIndex].ToString();  
    cmd.Parameters.Add("@xh",xh);  
    sqlConnection1.Open();      cmd.ExecuteNonQuery();  
    sqlConnection1.Close();  
    DataGrid1.EditItemIndex =-1;      this.BindToGrid();
```

```
}
```



第三部分 Socket网络编程

朱运乔



第六讲 C#套接字 (Socket)

编程基础

一、主要内容

- Socket简介
- Socket编程原理

二、目的和要求:

- 掌握C#面向连接套接字编程方法
- 掌握C#无连接套接字编程方法



Socket简介

一、什么是Socket

Socket在计算机中提供了一个通信接口，可以通过这个接口与任何一个具有Socket接口的计算机通信。应用程序在网络上传输，接收的信息都通过这个Socket接口来实现。

一旦socket配置完成，应用程序就可以：

- 把数据传给socket，从而进行网络传输
- 从socket接收数据(其他主机通过网络发送过来的)

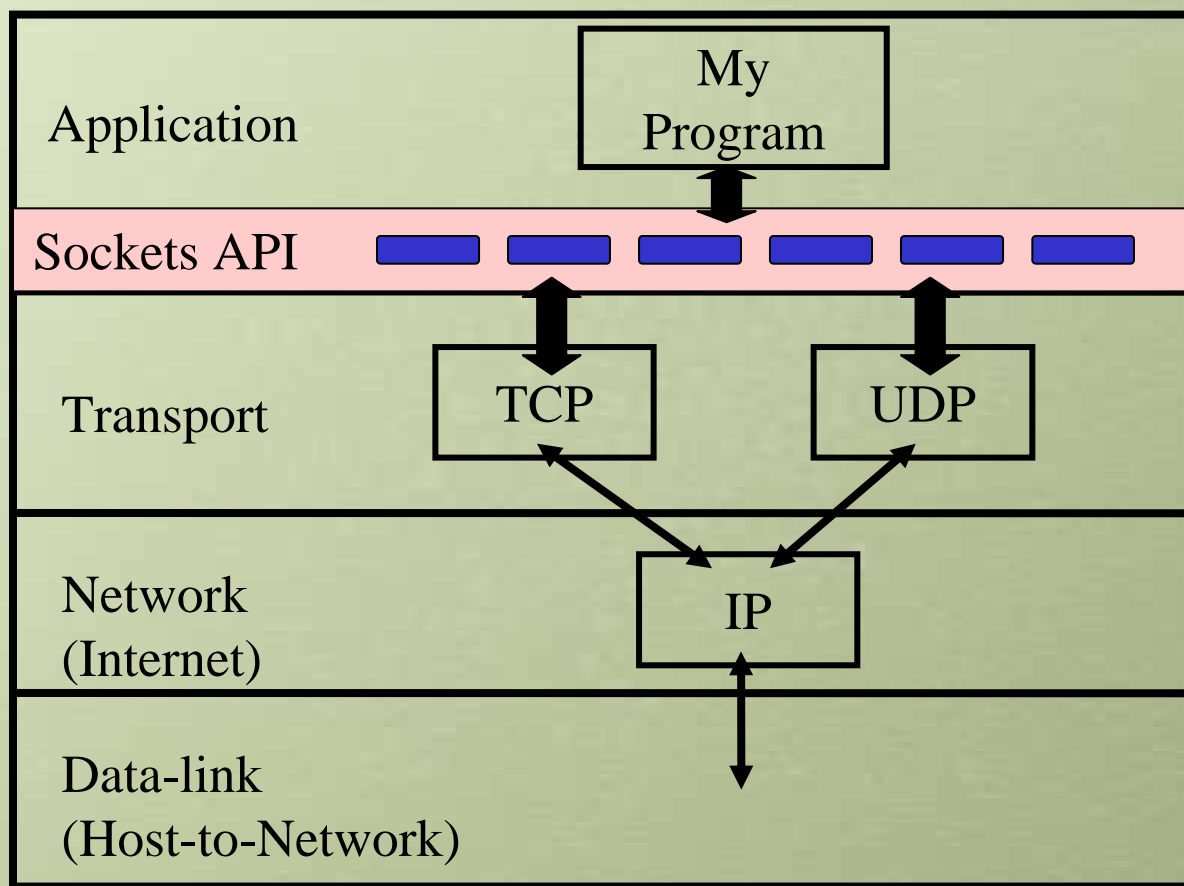
socket 类型决定了通信的类型：

- 可靠的 vs. 不可的、尽最大努力的
- 面向连接的 vs. 无连接的



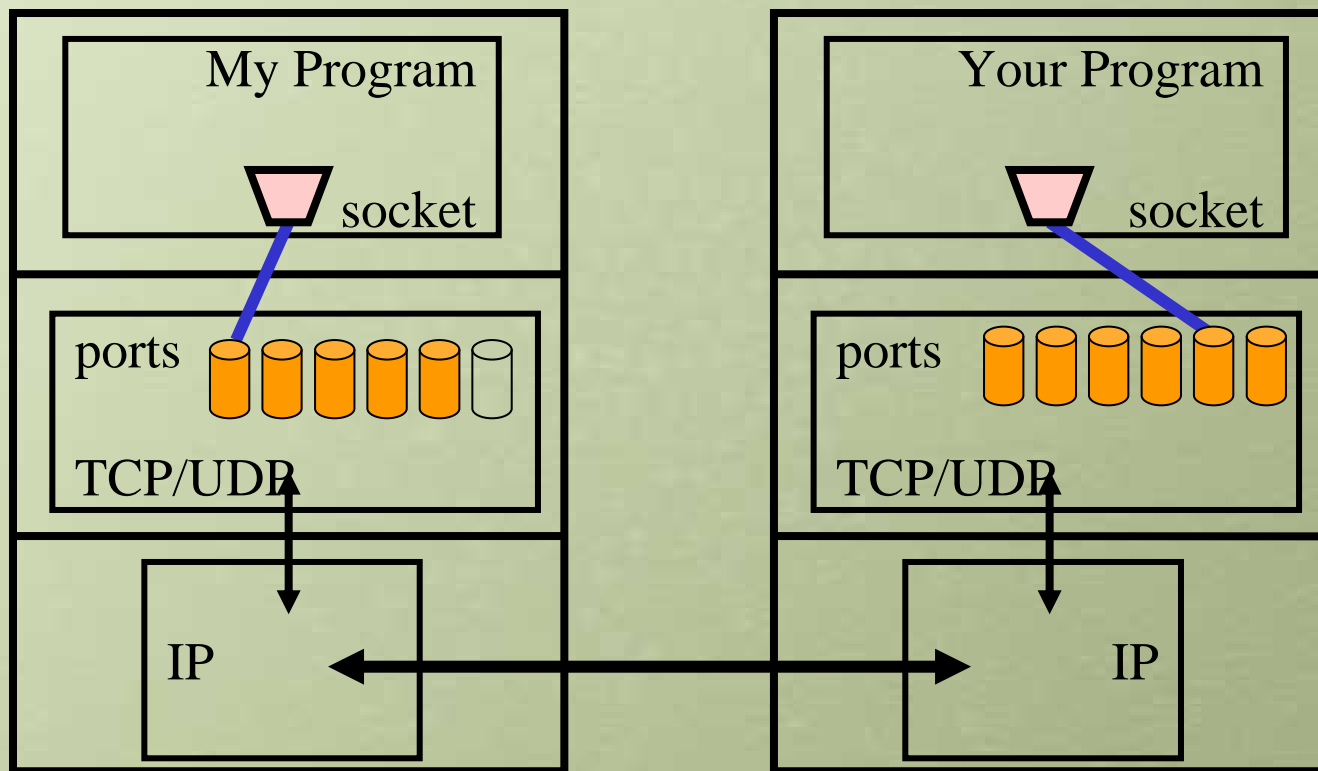
Socket简介

二、Socket在协议栈中的位置



Socket简介

三、Socket到Socket的通信



Socket简介

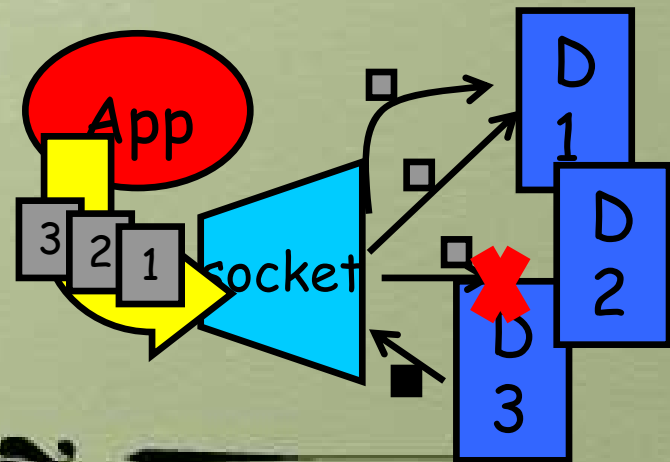
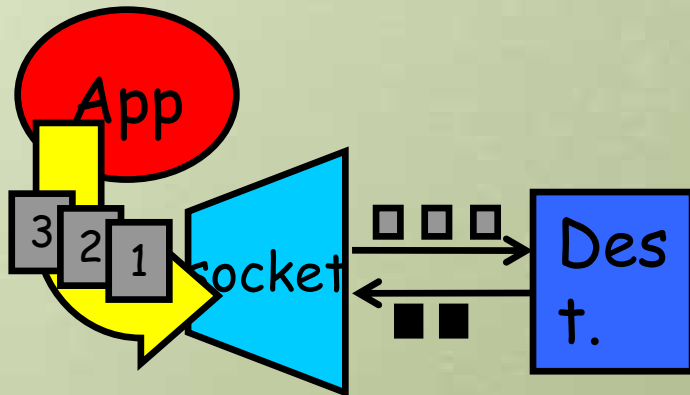
四、Socket的两种基本类型

- SOCK_STREAM

- TCP
- 可靠传输
- 保证顺序
- 面向连接
- 双向

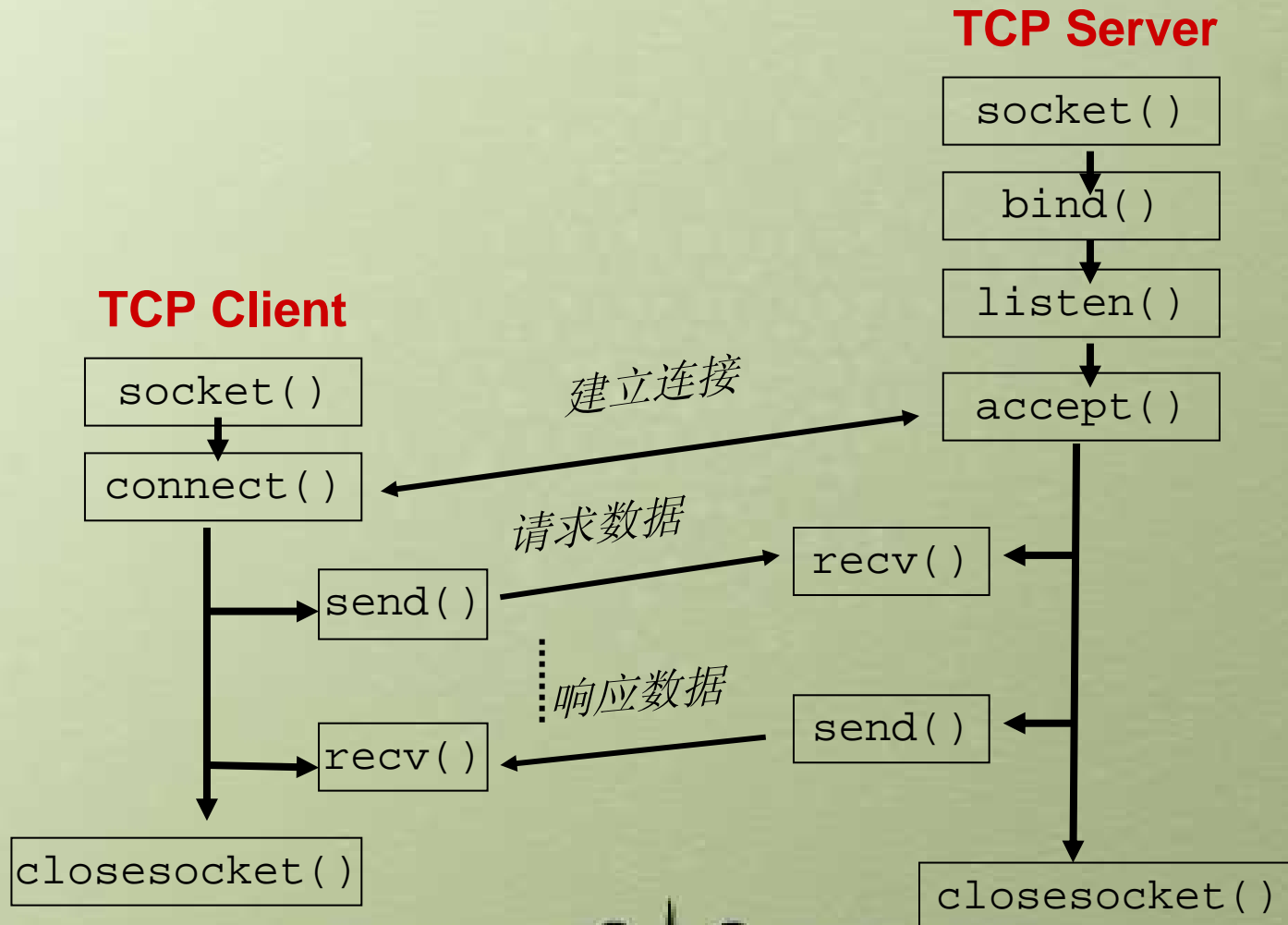
- SOCK_DGRAM

- UDP
- 不可靠传输
- 无顺序保证
- 无“连接”概念 - 应用程序为每个包指定目的地
- 可以发送或接收



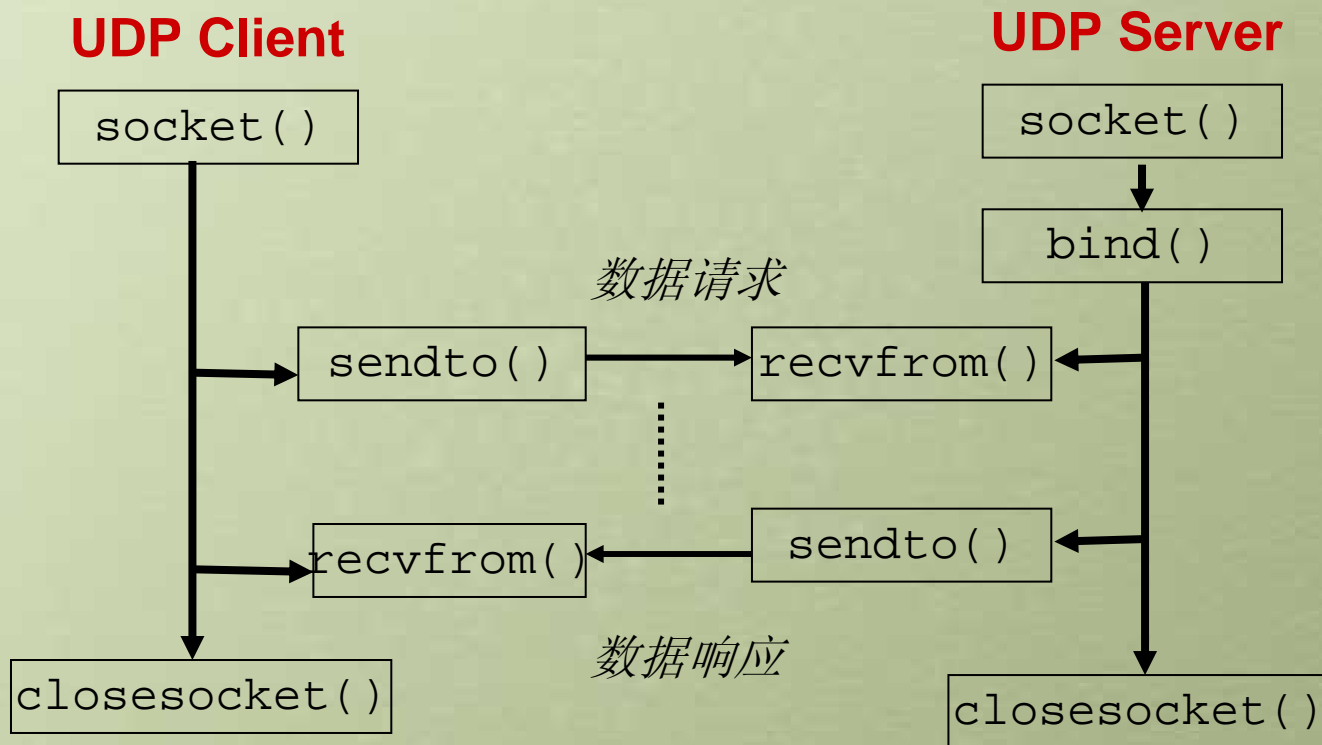
Socket简介

五、TCP Client-Server交互流程



Socket简介

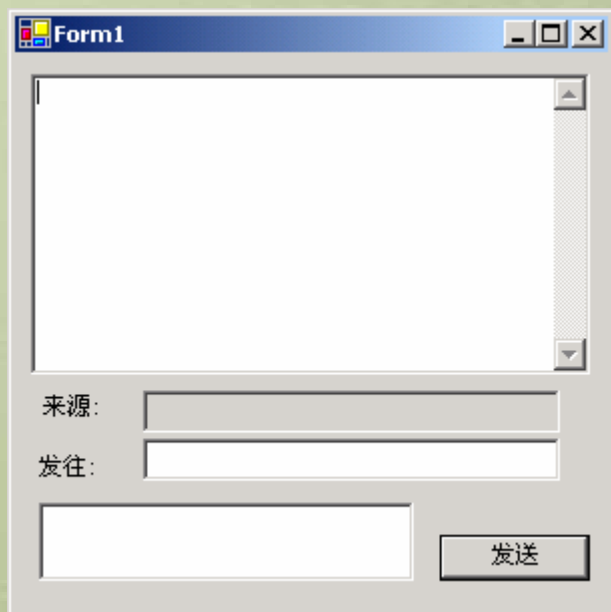
二、UDP Client-Server交互流程



范例介绍

[项目七]利用无连接Socket编写一简单的聊天程序，该程序主界面如下图。

该程序要在两台或多台以上需要相互通讯的机器上运行，先在“发往”编辑框中输入通讯对方法IP地址，然后在下面的信息发送框输入信息，单击发送，则对方的信息接收框中可得到发送的信息，对方的“来源”框可显示信息来源地的IP地址；对方可根据来源地址跟您通讯，在你的信息接收框可看到对方发来的信息，对方也可跟其它运行有该程序的机器通讯。



范例介绍

1、制作过程

1) 生成应用程序主界面，添加以下控件：

| | |
|----------|--------------|
| textBox1 | 接收信息编辑框 |
| textBox2 | 信息来源地的IP地址 |
| textBox3 | 发往目的地的目标IP地址 |
| textBox4 | 信息发送输入框 |
| button1 | 信息发送按钮 |

界面元素的属性设置参考主界面图，这里从略。

2) 代码生成

(1) 在主窗体代码前面加入相关名字空间引用：

| | |
|---------------------------|------------------|
| using System.Net; | //网络相关基本类的所在名字空间 |
| using System.Net.Sockets; | //套接字所在名字空间 |
| using System.Threading; | //线程相关类所在名字空间 |
| using System.Text; | //文字相关类所在名字空间 |

(2) 在主窗体类中加入下面字段并初始化。



范例介绍

```
Socket udp = new  
Socket(AddressFamily.InterNetwork,SocketType.Dgram,ProtocolType.Udp);
```

//构造函数的三个参数的意义:

//第一个参数代表地址族为 InterNetwork

//第二个参数代表套接字类型为Dgram

//第三个参数代表套接字所用协议为Udp

```
Thread th;           //定义一个线程用接收到达的信息
```

(3) 添加一个方法void f(), 作为线程工作时的代码:

```
void f()
```

```
{
```

```
    IPHostEntry ipinfo = Dns.Resolve(Dns.GetHostName());
```

//Dns的方法GetHostName()获取本地主机名

//Resolve()方法由主机名或IP地址得到主机IP列表(IPHostEntry)对象ipinfo

```
    IPAddress addr = ipinfo.AddressList[0];
```

//由ipinfo中取得第一个IP地址 (IPAddress) 对象

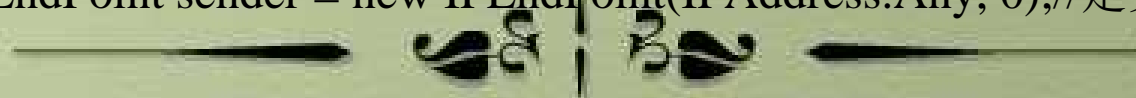
```
    IPEndPoint endp = new IPEndPoint(addr,11000);
```

//由IP地址和端口号生成本地通信终端结点对象

```
    udp.Bind(endp);
```

//将本终端结点对象同Socket对象绑定起来

```
    IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0); //定义任意远程端点
```



范例介绍

```
Byte[] msg = new Byte[256];           //定义用以接收数据的字节数组
while(true)
{
    EndPoint senderRemote = (EndPoint)sender;
    int c = udp.ReceiveFrom(msg,ref senderRemote);
    string str1 = Encoding.Unicode.GetString(msg,0,c)+"\r\n";
    string str2 = ((IPEndPoint)senderRemote).Address.ToString();
    settext(str1,str2);
}
```

//其中调用的方法void settext(string, string)为自定义方法，用来将收到地信息写到窗体的相应位置。

```
void settext(string str1,string str2)
{
    textBox1.Text+=str1;
    textBox2.Text=str2;
    textBox3.Text= str2;
}
```



范例介绍

(4) 主窗体Load事件处理方法:

```
private void Form1_Load(object sender, EventArgs e)
{
    th = new Thread(new ThreadStart( f));
    //利用方法f创建接收进程
    th.Start();
    //进程开始工作，等待接收信息
}
```

(5) 添加发送按钮的响应代码:

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox4.Text.Trim()!="")
    {
        IPHostEntry ipinfo = Dns.Resolve(textBox3.Text);
        IPEndPoint endp = new IPEndPoint(ipinfo.AddressList[0], 11000);
        EndPoint remote = (EndPoint)endp;
        Socket s = new Socket(endp.Address.AddressFamily,
                               SocketType.Dgram,
                               ProtocolType.Udp);

        //创建无连接套接字，用它发送数据
    }
}
```

范例介绍

```
Byte[] date = Encoding.Unicode.GetBytes(textBox4.Text);  
s.SendTo(date, remote);    //向远程端点发送信息  
s.Close();                  //关闭发送套接字  
textBox4.Text="";          //清除发送输入框  
textBox4.Focus();           //重先定位输入焦点  
}
```

```
}
```

（6）添加窗体关闭事件处理方法：

```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)  
{  
    udp.Close();             //关闭接收套接字  
    th.Abort();               //强制终止接收线程  
}
```



相关知识

1、创建套接字

下面的示例创建一个无连接套接字，它可用于在基于 TCP/IP 的网络（如 Internet）上通信。

```
Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,  
ProtocolType.Udp);
```

AddressFamily 枚举指定 Socket 类用来解析网络地址的标准地址族（例如，AddressFamily.InterNetwork 成员指定 IP 版本 4 地址族）。

SocketType 枚举指定套接字的类型（例如，SocketType.Stream 成员表示一个用于发送和接收数据并且支持流控制的标准套接字）。

ProtocolType 枚举指定在 Socket 上通信时使用的网络协议（例如，ProtocolType.Tcp 指示套接字使用 TCP；ProtocolType.Udp 指示套接字使用 UDP）。

2、无连接套接字所用的主要方法：

SendTo 将信息发往指定的端点。

ReceiveFrom 从指定的端点接收信息。

Bind 使 Socket 与一个本地终结点相关联。

Close 关闭 Socket 连接并释放所有关联的资源。



相关知识

3、相关类介绍

1) **IPHostEntry**类 为 Internet 主机地址信息提供容器类。

主要属性成员：

| 名称 | 说明 |
|----|----|
|----|----|

| | |
|--------------------|----------------------|
| AddressList | 获取或设置与主机关联的 IP 地址列表。 |
|--------------------|----------------------|

| | |
|----------------|------------------|
| Aliases | 获取或设置与主机关联的别名列表。 |
|----------------|------------------|

| | |
|-----------------|------------------|
| HostName | 获取或设置主机的 DNS 名称。 |
|-----------------|------------------|

2) **Dns**类 提供简单的域名解析功能。

主要方法成员：

GetHostName 获取本地计算机的主机名。

Resolve 将 DNS 主机名或 IP 地址解析为 **IPHostEntry** 实例。

3) **EndPoint** 类 标识网络地址。这是一个 abstract（抽象）类。

4) **IPEndPoint** 类 从**EndPoint**类派生，将网络端点表示为 IP 地址和端口号。**IPEndPoint** 类包含应用程序连接到主机上的服务所需的主机和本地或远程端口信息。通过组合服务的主机 IP 地址和端口号，**IPEndPoint** 类形成到服务的连接点。



相关知识

主要属性

名称

说明

Address 获取或设置终结点的 IP 地址。

AddressFamily 获取网际协议 (IP) 地址族。

Port 获取或设置终结点的端口号。

其中，

属性 Address 属于 IPAddress 类型。

属性 AddressFamily 属于 AddressFamily 枚举类型。

EndPoint 构造函数：

EndPoint (IPAddress, Int32) 用指定的地址和端口号初始化 **EndPoint** 类的新实例。如本项中有，

```
EndPoint endp = new EndPoint(addr,11000);
```

5) IPAddress 类 提供网际协议 (IP) 地址。



范例介绍

[项目八]TCP套接字应用。下面是一面向连接套接字的程序范例---也是一个聊天程序，但功能要强大些，也相对要复杂些，该聊天系统分两部分：聊天服务端和客户端。

服务端负责监听接受客户端的登录，接收来自客户端发来的信息，并根据信息类别向其它客户端转发信息。

客户端为聊天平台，它可根据服务端传送的所有在线的客户名信息有针对性地选择一个客户，作为聊天对象，然后将聊天内容发到服务端，由服务端转发到目标客户。

所有客户端在聊天时，都一直同服务端处于连接状态。这也是同无连接套接字工作时的主要区别。

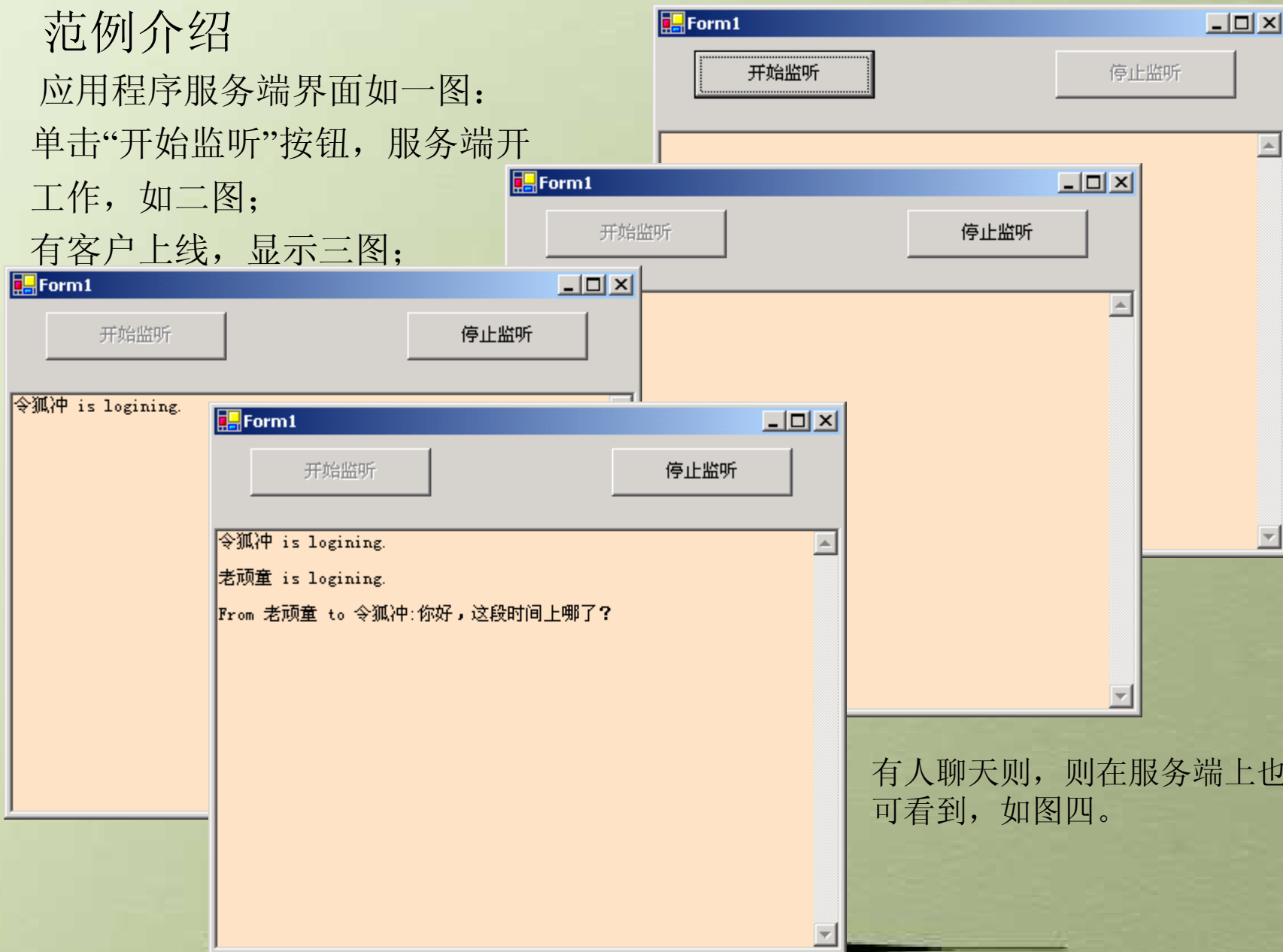


范例介绍

应用程序服务端界面如一图：

单击“开始监听”按钮，服务端开
工作，如二图；

有客户上线，显示三图；

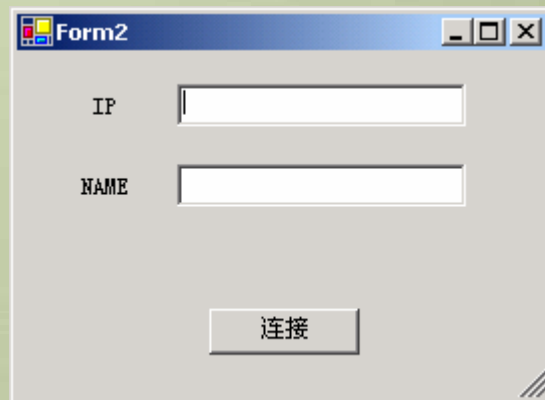


有人聊天则，则在服务端上也可看到，如图四。

范例介绍

客户端找开后如图一，要求连接到服务端，输入服务端IP地址及自己的网名。

正确登录到服务端后，出现下图二，

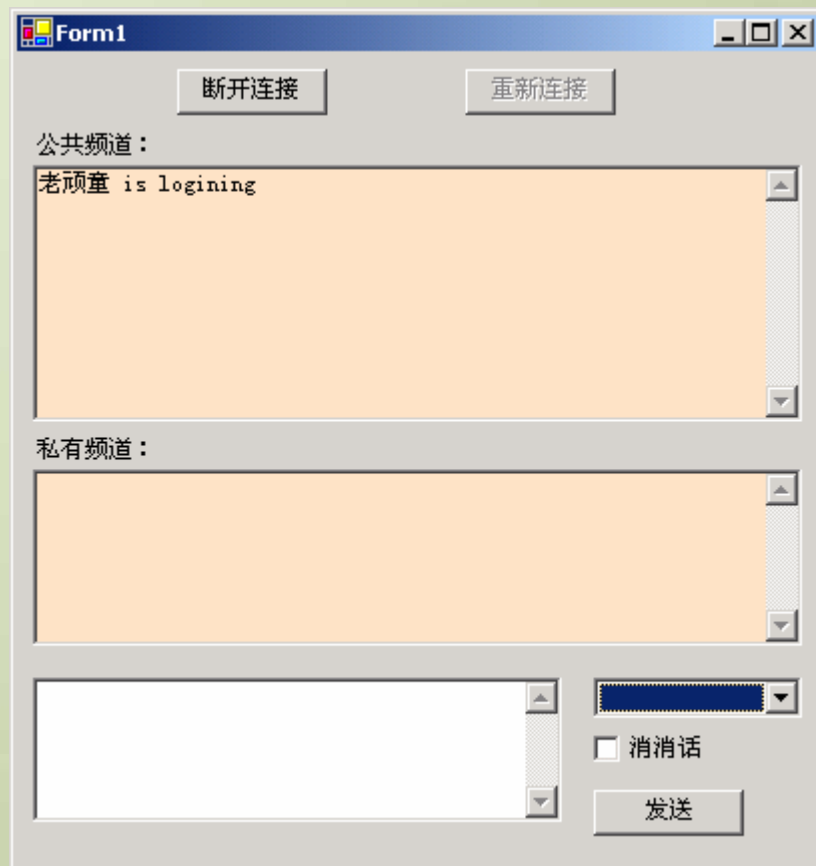


Form2

IP

NAME

连接



Form1

断开连接 重新连接

公共频道：

老顽童 is logging

私有频道：

☐ 悄悄话

发送

该界面两个按钮分别为断开和重新连接到服务端；公共频道公开聊天内容，私有频道为其它客户发送的私密聊天内容，不会出现在其它客户界面上；

最下面的编辑框为聊天信息发送框；右边的组合框中显示所有在线用户，可在其中选择聊天对象，其下是悄悄话选项，如果选中，信息只发往目标用户的私有频道，如果未选，则发向目标客户的信息显示在所有客户的公共频道。

下面的按钮为发送信按钮。

范例介绍

图一为“老顽童”未选悄悄话时向“令狐冲”发的信息。

图二为“老顽童”选择悄悄话时向“令狐冲”发的信息。

The screenshot shows a chat application window titled "Form1". At the top, there are two buttons: "断开连接" (Disconnect) and "重新连接" (Reconnect). Below these, there are two sections: "公共频道:" (Public Channel) and "私有频道:" (Private Channel). The "公共频道:" section contains a text area with the message "老顽童 is logging" and "From 令狐冲 to 老顽童:我在忙于教务。". The "私有频道:" section is empty. At the bottom, there is a text input field, a dropdown menu showing "令狐冲", a checkbox labeled "悄悄话" (Whisper) which is unchecked, and a "发送" (Send) button.

The screenshot shows a chat application window titled "Form1". At the top, there are two buttons: "断开连接" (Disconnect) and "重新连接" (Reconnect). Below these, there are two sections: "公共频道:" (Public Channel) and "私有频道:" (Private Channel). The "公共频道:" section contains a text area with the message "老顽童 is logging" and "From 令狐冲 to 老顽童:我在忙于教务。". The "私有频道:" section contains a text area with the message "From 令狐冲 to 老顽童:你现在哪里?". At the bottom, there is a text input field, a dropdown menu showing "令狐冲", a checkbox labeled "悄悄话" (Whisper) which is checked, and a "发送" (Send) button.

范例介绍

1、项目分析

本应用程序实际上包含服务端程序和客户端程序，所以要制作生成两个不同项目，由于服务端和客户端要进行通信，并且信息类型有多种，如有：

连接上线、聊天、私聊、下线等。

所有通信双方要互为识别，并共同约定。

本聊天系统中采用在所发信息的前面加上不同的标识，以标记不同的信息类型，接收方收到以后，根据类型作不同的处理。

本系统规定，在服务端，收到的信息类型标识为：

| 标识 | 含义 | 格式 | 所作反应 |
|------|------|-------------|-------------------------|
| conn | 用户上线 | conn 用户名 | 向所有在线用户发出conn类型信息，通知谁上线 |
| chat | 聊天信息 | chat 信息 | 向所有在线用户转发chat类型信息 |
| priv | 私聊信息 | priv 目标方 信息 | 向指定客户转发priv类型信息 |
| gone | 用户下线 | gone 用户 | 向所有在线用户发出gone类型信息，通知谁下线 |

这些各种信息由客户发出，然后由服务端处理并转发向各客户端，同时，服务端在停止监听关闭所有连接时，向客户端发出“end”信息标识。



范例介绍

在客户端，收到的信息类型标识为：

| 标识 | 含义 | 信息格式 | 所作反应 |
|------|-------|------------|--------------------|
| conn | 用户上线 | conn 上线用户 | 显示某某上线消息，更新用户列表组合框 |
| chat | 聊天信息 | chat 信息 | 在公共频道显示聊天信息 |
| priv | 私聊信息 | chat 来源 信息 | 在私有频道显示聊天信息 |
| gone | 用户下线 | gone 离开用户名 | 更新用户列表组合框 |
| end | 服务端关闭 | end | 关闭套接字，更新界面 |

2、服务端项目制作过程

1) 生成服务端应用程序主界面。

在主窗体中添加两个按钮button1和button2，用来开始监听和停止监听。

添加一个textBox1文本控件，显示来自客户端的信息。

2) 主要属性设置。

| 控件 | 属性 | 值 |
|-----------------|---------------|--------|
| 停止监听按钮(button1) | Enable | false |
| textBox1 | ReadOnly(只读) | true |
| | BackColor(背景) | 自行设置即可 |



范例介绍

3) 代码实现

(1) 在主窗体代码前添加相关的名字空间引用

```
using System.Net;
```

```
using System.Net.Sockets;
```

```
using System.Threading;
```

(2) 在当前项目中添加新类ClientItem

说明：通过该类封装一个套接字成员及相它相关成员，当服务器接收到一个客户端连接并生成通信套接字后，实例化该类（生成一个该类对象），使该实例的套接字成员引用到生通信套接字。

简单地说，每一个ClientItem类对象代到一个客户端的连接。

所以接着要做下面几步：

(3) 在本类代码前添加名字间的引用

```
using System.Net;
```

```
using System.Net.Sockets;
```

```
using System.Threading;
```

```
using System.Text;
```



范例介绍

(4) 在类ClientItem中添加字段成员：

```
public string name;    //连接客户的用户名
public Socket client;  //同客户方建立连接的套接字
public Thread thread;  //接收信息的线程
public Form1 fk;       //引用主窗体
delegate void f(ClientItem ci,string msg); //定义委托类型，其实例将用以
//封装主窗体的void Receive (ClientItem ci,string msg)方法。
```

(5) 定义ClientItem 类的一个构造函数：

```
public ClientItem(Form1 fk,string name,Socket client)
{
    this.fk = fk;        //创建时保存主窗体引用
    this.name=name;      //对应客户名
    this.client=client;  //保存同客户端连接的套接字引用
    thread = new Thread(new ThreadStart(Run)); //创建接收信息的线程
    thread.Start();      //线程开始工作
}
```



范例介绍

(6) 在类ClientItem中定义方法void Run(), 作为线程thread的工作代码:

```
void Run()
{   while (true)
    {   Byte[] buff= new Byte[256];
        string str="";
        int i;
        i=client.Receive(buff);
        str+=System.Text.Encoding.Unicode.GetString(buff,0,i);
        f fhandle = new f(fk.Receive);//利用委托f的实例封装主窗体方法Receive
        fk.Invoke(fhandle,new object[]{this,str});//调用主窗体的Receive
                                           //以处理接收到的数据。
    }
}
```

(7) 在类ClientItem中定义方法void Send(string smg), 用发送指定信息:

```
public void Send(string smg)
{
    Byte[] bytes=Encoding.Unicode.GetBytes(smg);
    client.Send(bytes);
}
```



范例介绍

(8) 在类ClientItem中定义方法void close(), 用以关闭同客户的连接:

```
public void close()
{
    client.Close();
    thread.Abort();
}
```

(9) 在主窗体类中加入下列字段:

```
TcpListener Server; //该类封了基础Socket, 开始后可自动绑定并监听客户连接
ArrayList Clients= new ArrayList(10); //用数组链表保存每一到客户的连接
Thread thread; //该线程用以监并接受客户的连接请求
bool stop=true; //作为thread线程代码中的一个循环条件, 是否继续接受连接请求
```

(10) 在主窗体中添加自定义方法Listen, 将作为线程thread的工作代码。

```
private void Listen ( )
{
    Server = new TcpListener ( 1234 );
    Server.Start ( ); //开始侦听
    while(!stop) {
        Socket temp = Server.AcceptSocket( ); //返回可以用以处理连接的Socket
        //实例
```



范例介绍

```
ClientItem ci=new ClientItem(this,"",temp);//this代表本窗体对象，网名  
//暂时为空，temp为同客户端连接的套接字，构造生成一个ClientItem类的实例  
Clients.Add(ci);//将一个代表同客户端连接的实例加入到列表  
}
```

```
}
```

(11) 添加“开始监听”按钮的鼠标单击事件处理代码

```
private void button1_Click(object sender, System.EventArgs e)  
{
```

```
    stop=false;
```

```
    thread = new Thread(new ThreadStart(Listen));//创建线程
```

```
    thread.Start();           //启动线程
```

```
    button1.Enabled=false;    //当前按钮禁用
```

```
    button2.Enabled=true;     //停止监听按钮可用
```

```
}
```

(12) 添加自定义方法Receive，用来处理ClientItem实例收到并传来的信息。

```
public void Receive(ClientItem ci,string msg)
```

```
{
```

```
    string[] temp = msg.Split(new char[]{ '|' });调用字符串的Split方法，分离出  
//用“|”分隔的多个字符串，保存到字符串数组temp中。
```



范例介绍

```
switch(temp[0])
{
    //根据不同的预定义的消息头，调用不同的自定义方法
    case "conn":login(ci,msg);break;//用login方法处理conn(上线)类型消息
    case "chat":chat(ci,msg);break; //用chat方法处理chat（聊天）类型消息
    case "priv":priv(ci,msg);break; //用priv方法处理priv（私聊）类型消息
    case "gone":gone(ci);break;      //用gone方法处理gone（离开）类型消息
}
```

（13）自定义上面提到的四个分类处理方法

```
private void login(ClientItem ci,string msg)
{
```

```
    string[] temp = msg.Split(new char[]{"|"});
    string ssend="conn|"+temp[1];
    for(int i=0;i<Clients.Count;i++)
    {
        //遍历所有客户连接
        ClientItem e=(ClientItem)(Clients[i]);
        if(e!=ci) ssend+="|"+e.name;//如果e不是当前接收到conn连接信
                                   //号的实例
        else      e.name=temp[1]; //如果e是当前接收到信号的实例，
                                   //则将用户名域填上
    }
```

范例介绍

```
ci.Send(ssend);//向当前实例对应的客户连接端发送所有客户名信息
textBox1.AppendText(temp[1]+" is logining.");//向文本框显示登录信息
textBox1.AppendText("\r\n\r\n");
foreach(ClientItem e in Clients)
    if(e!=ci)
        e.Send("conn|"+temp[1]); //向非当前发信号客户发送当前上线的用户名
}

private void chat(ClientItem ci,string msg)
{
    string[] temp = msg.Split(new char[]{'|'});
    textBox1.AppendText(temp[1]);
    textBox1.AppendText("\r\n\r\n");
    foreach(ClientItem e in Clients)
    {
        e.Send("chat|"+temp[1]);    //向所有客户发送聊天信息
    }
}
```



范例介绍

```
private void priv(ClientItem ci,string smg)
{
    string[] temp = smg.Split(new char[]{'|'});
    foreach(ClientItem e in Clients)
        if(e.name==temp[1])
            e.Send("priv|"+ci.name+"|"+temp[2]);
    textBox1.AppendText(temp[2]);
    textBox1.AppendText("\r\n\r\n");
}
private void gone(ClientItem ci)
{
    foreach(ClientItem e in Clients)
        if(e!=ci) e.Send("gone|"+ci.name); //向非当前用户发当前用
                                                //户已下线
    textBox1.AppendText("BYE "+ci.name);
    textBox1.AppendText("\r\n\r\n");
    ci.close(); //关闭同该客户的连接
    Clients.Remove(ci); //从链表中移除客户连接
}
```



范例介绍

(14) 添加停止监听按钮的事件代码

```
private void button2_Click(object sender, System.EventArgs e)
{
    Server.Stop();      //关闭监听套接字
    stop=true;          //停止线程循环
    thread.Abort();      //终止线程
    foreach(ClientItem elem in Clients)
    {
        elem.Send("end"); //向所有客户端发止服务信号
        elem.close();      //关闭同所有客户端的套接字连接
    }
    Clients.Clear();      //清空客户连接链表
    button1.Enabled=true;; //使“开始监听”按钮可用
    button2.Enabled=false; //使“停止监听”按钮禁用
}
```

(15) 添加窗体将要关闭时所听响应的Closing事件代码

```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    if(!stop) {Server.Stop();
               stop=true;
               thread.Abort();
               }
}
```



范例介绍

```
foreach(ClientItem elem in Clients)
{
    elem.Send("end");
    elem.close();
}
Clients.Clear();
}
```

3、客户端项目制作过程

1) 生成服务端应用程序主界面。

(1) 在主窗体上添加三个框辑框:

textBox1: 用来发送信息

textBox2: 用来显示不保密的信息

textBox3: 用来显示保密信息

(2) 在主窗体上添加三个按钮:

button1: 发送信息按钮

button2: 断开连接按钮

button3: 重新连接按钮



范例介绍

(3) 在主窗体上添加一个组合框和一个复选框:

comboBox1: 显示在线用户列表; checkBox1: 悄悄话选项。

2) 主要属性设置。

| 控件 | 属性 | 值 |
|-------------|------------|-----------------|
| 三个框编辑框 | ScorllBars | Vertical(垂直滚动条) |
| 其中两个信息显示编辑框 | ReadOnly | true |

其它属性设置参考主界面。

3) 代码实现

(1) 在主窗体代码前添加相关的名字空间引用

```
using System.IO; //程序中用到的网络输入/输出流类所在名字空间
```

```
using System.Net; //网络相关类名字空间
```

```
using System.Net.Sockets; //网络套接字名字空间
```

```
using System.Threading; //线程类的名字空间
```

(2) 在主窗体类中加入所需的字段:

```
TcpClient client; //定义客户套接字, TcpClient类中封装了Socket类, 简化使用
```

```
bool loop=true; //loop定义为进程代码中的循环控制变量
```

```
Thread thread; //定义接收数据进程
```

```
string name; //保存登录用户名
```



范例介绍

(3) 在当前项目中添加一个窗体Form2，用连接登录到服务端：

- 在窗体Form2中加入两个编辑框textBox1和textBox2，分加用来输入IP地址和用户名；
- 在两个编辑框的前面各加一个标签label1和label2，分加将其Text属性改为IP和Name；
- 在窗体Form2中加入一个连接按钮。

(4) 在窗体Form2类代码中加入下面字段：

```
public string ip; //运行时获取IP地址
```

```
public string name; //运行时获取用户名
```

(5) 在窗体Form2中添加连接按钮的Click事件处理方法：

```
private void button1_Click(object sender, System.EventArgs e)
```


```
{  
    if(this.textBox1.Text=="") return;  
    if(this.textBox3.Text=="") return;  
    ip=textBox1.Text; //保存IP  
    name=textBox3.Text; //保存用户名  
    this.DialogResult=DialogResult.OK;//关闭本对话框，返回OK  
}
```



范例介绍

(6) 添加主窗体的Load事件处理方法:

```
private void Form1_Load(object sender, System.EventArgs e)
{
    Form2 dlg = new Form2();
    if(dlg.ShowDialog() == DialogResult.OK) { //显示对话框dlg
        try { client = new TcpClient(dlg.ip, 1234); //用对话框的IP和端口生成套接字
            }
        catch(SocketException ee) { //try.....catch.....结构是捕获程序异常
            MessageBox.Show("连接失败!"); }
        name = dlg.name; //连接成功后保存用户名
        send("conn|" + name); //向服务端发送conn(上线)信号
        thread = new Thread(new ThreadStart(Run)); //创建接收信息的线程
        thread.Start(); //线程工作, 开始接收信号
        this.button1.Enabled = true; //断开连接按钮可用
        this.button2.Enabled = true; //发送按钮可用
        this.button3.Enabled = false; //重新连接按钮禁用
    } else { //如果不是单击连接按钮关闭对话框
        this.button1.Enabled = false; //断开连接按钮禁用
        this.button2.Enabled = false; //发送按钮禁用
        this.button3.Enabled = true; //重新连接按钮可用
    }
}
```



范例介绍

(7) 在主窗体中添加自定义方法Run,用作thread线程的工作代码:

```
void Run()
{
    string msg=string.Empty;    //定义接收信息字符串（可不赋初始值）
    NetworkStream stream = client.GetStream();    //取得套接字网络数据流
    Byte[] data = new Byte[256]; //定义接收信息的字节数组
    while(loop)
    {
        //当loop为真时循环
        Int32 bytes = stream.Read(data, 0, data.Length); //等待读取数据
        msg = System.Text.Encoding.Unicode.GetString(data, 0, bytes); //转换成字符串
        process(msg);    //调用自定义方法以处理接收到的信息
    }
    stream.Close();    //循环退出时关闭数据流
    client.Close();    //关闭套接字
}
```

(8) 自定义方法process以处理分析收到的信息:

```
void process(string msg)
{
    string[] temp = msg.Split(new char[] { '|' }); //分开用“|”分隔的信息
    switch(temp[0])
    {
        //temp[0]中为信息类型，根据具体类型调度不同的处理方法
    }
```



范例介绍

```
        case "conn":conn(msg);break;
        case "chat":chat(temp[1]);break;
        case "priv":priv(msg);break;
        case "gone":gone(temp[1]);break;
        case "end":svr_end();break;
    }
}
```

(9) 下面分别添加上面所调用的五个自定义方法

//处理上线消息

```
void conn(string msg)
{
    string[] temp = msg.Split(new char[]{'|'});
    for(int i=1;i<temp.Length;i++)
    {
        comboBox1.Items.Add(temp[i]);//向组合框中添加用户名
    }
    textBox2.AppendText(temp[1]+" is logining"); //编辑框添加新用户上线
    textBox2.AppendText("\r\n\r\n");
}
```

//在数组temp中，temp[0]为conn标识，其它元素为用户名，如果当前用户不是刚
//刚上线用户，则数组中将只有两个元素，其中temp[1]为刚上线用户名



范例介绍

//处理聊天消息

```
void chat(string msg)
```

```
{  
    textBox2.AppendText(msg);           //添加信息到公共频道显示框  
    textBox2.AppendText("\r\n\r\n");    //空一行  
}
```

//处理私聊消息

```
void priv(string msg)
```

```
{  
    string[] temp = msg.Split(new char[]{'|'});  
    int ind=comboBox1.FindString(temp[1]);//temp[1]中为信息来源用户  
    if(ind!=-1)  
    {  
        comboBox1.SelectedIndex=ind;//在组合框上选中信息来源用户  
        textBox3.AppendText(temp[2]);//在私有频道显示框中添加信息  
        textBox3.AppendText("\r\n\r\n");//空一行  
    }  
}
```



范例介绍

//处理有用户下线的消息

```
void gone(string name)
```

```
{
```

```
    int ind=comboBox1.FindString(name); //在组合框中找到下线用户
```

```
    if(ind!=-1)
```

```
    { //如果找到
```

```
        comboBox1.Items.RemoveAt(ind);    //删除找到的用户名
```

```
        textBox2.AppendText(name+" is left");//显示用户已离开的信息
```

```
        textBox2.AppendText("\r\n\r\n");    //空一行
```

```
    }
```

```
}
```

//处理服务器已关闭的消息

```
void svr_end()
```

```
{
```

```
    MessageBox.Show("聊天服务器关闭。");
```

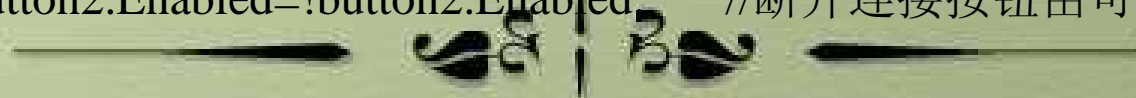
```
    comboBox1.Items.Clear();    //清除组合框中所有用户
```

```
    textBox2.Text="";
```

```
    textBox3.Text="";;
```

```
    button1.Enabled=!button1.Enabled;    //发送按钮由可用变为禁用
```

```
    button2.Enabled=!button2.Enabled;    //断开连接按钮由可用变为禁用
```



范例介绍

```
button3.Enabled=!button3.Enabled;    //重新连接按钮由禁用变为可用
client.Close();                      //关闭套接字
loop=false;                          //终止线程中的循环语句
thread.Abort();                      //终止接收信息的thread线程
```

```
}
```

(10) 在主窗体中添加自定义方法Send, 用以发送消息

```
void send(string msg)
```

```
{
    Byte[] buff=System.Text.Encoding.Unicode.GetBytes(msg);
    NetworkStream stream = client.GetStream();//取得套接字网络数据流
    stream.Write(buff,0,buff.Length);        //发送消息
    textBox1.Text="";                        //清空发送框
    textBox1.Focus();                        //焦点重新定位到发送框
}
```

(11) 在主窗体中添加断开连接的Click事件处理方法

```
private void button2_Click(object sender, EventArgs e)
```

```
{
    send("gone|" + name);                  //向服务端发送下线的信息
    client.Close();                        //关闭套接字
    loop=false;                            //将线程中的循环条件置为假
```



范例介绍

```
thread.Abort();           //终止线程
comboBox1.Items.Clear();  //清空组合框
textBox2.Text="";
textBox3.Text="";
button1.Enabled=false;    //发送按钮设为禁用
button2.Enabled=false;    //断开按钮设为禁用
button3.Enabled=true;     //重新连接按钮设为可用
}
```

（12）添加主窗体重重新连接按钮的Click事件处理方法：

```
private void button3_Click(object sender, System.EventArgs e)
```

```
{//代码同窗体的Load事件处理方法一样，所以注释略。
```

```
Form2 dlg =new Form2();
```

```
if(dlg.ShowDialog()==DialogResult.OK)
```

```
{      try
```

```
{
```

```
client= new TcpClient(dlg.ip,1234);
```

```
}
```

```
catch(SocketException ee )
```

```
{
```

```
MessageBox.Show("连接失败！");
```

```
}
```

范例介绍

```
        name=dlg.name;
        loop=true;
        thread= new Thread(new ThreadStart(Run));
        thread.Start();
        send("conn|"+name);
        this.button1.Enabled=true;
        this.button2.Enabled=true;
        this.button3.Enabled=false;
    }
    else
    {
        this.button1.Enabled=false;
        this.button2.Enabled=false;
        this.button3.Enabled=true;
    }
}
```

（13）添主窗体发送按钮的Click事件处理方法

```
private void button1_Click(object sender, System.EventArgs e)
{
    string msg;
```



范例介绍

```
if(comboBox1.SelectedIndex!=-1)
{
    string name=(string)comboBox1.SelectedItem;
    if(checkBox1.Checked)
        msg="priv|" + name + "|From " + this.name + " to " + name + ":" + textBox1.Text;
    else
        msg= "chat|From " + this.name + " to " + name + ":" + textBox1.Text;
    send(msg);
}
}
```

