C#网络编程及应用开发实例与习题解答

21世纪高等院校计算机教材系列

C#网络编程及应用
开发实例与习题解答

刘瑞新 主编
马 骏 何 欣 等编著

机械工业出版社
CHINA MACHINE PRESS

21 世纪高等院校计算机教材系列

C#网络编程及应用开发实例与习题解答

刘瑞新　主编

马 骏　何 欣　等编著

机械工业出版社

本书是《C#网络编程及应用》的配套教材。本书结合教材的内容，介绍了 3 个应用编程的实际例子，可以帮助读者通过调试开发实例，提高实际动手的能力；也可以作为学生结合所学内容进行上机综合练习的题目。本书对《C#网络编程及应用》教材中的全部习题都给出了参考解答。

图书在版编目（CIP）数据

C#网络编程及应用开发实例与习题解答/刘瑞新主编. —北京：机械工业出版社，2004.6
（21 世纪高等院校计算机教材系列）
ISBN 7-111-14559-3

Ⅰ C … Ⅱ. 刘… …C 语言—程序设计—高等学校—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2004）第 049400 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策 划：胡…
责任编辑：… 琳
责任印制：洪汉军

三河市宏达印刷有限公司印刷 · 新华书店北京发行所发行

2004 年 6 月第 1 版 · 第 1 次印刷
787mm×1092mm 1/16 · 8.25 印张 · 198 千字
0001－5000 册
定价：13.00 元

出 版 说 明

计算机技术是一门迅速发展的现代科学技术，它在经济建设与社会发展中，发挥着非常重要的作用。近年来，我国高等院校十分注重人才的培养，大力提倡素质教育、优化知识结构，提倡大学生必须掌握计算机应用技术。为了满足教育的需求，机械工业出版社组织了这套“21 世纪高等院校计算机教材系列”。

在本套系列教材的组织编写过程中，我社聘请了各高等院校相关课程的主讲老师进行了充分的调研和细致的研讨，并针对非计算机专业的课程特点，根据自身的教学经验，总结出知识点、重点和难点，一并纳入到教材中。

本套系列教材定位准确，注重理论教学和实践教学相结合，逻辑性强，层次分明，叙述准确而精炼，图文并茂，习题丰富，非常适合各类高等院校、高等职业学校及相关院校的教学，也可作为各类培训班和自学用书。

参加编写本系列教材的院校包括：清华大学、西安交通大学、北方交通大学、北京邮电大学、北京化工大学、北京科技大学、山东大学、首都经贸大学等。

机械工业出版社

前　言

要想快速提高实际编程能力，除了多看书之外，最主要的方法就是多做练习，多上机实践。只有多试多练，才能逐渐积累宝贵的经验，才能真正使自己的编程水平得到实质性的提高。

本书是《C#网络编程及应用》的配套教材。为了给读者多提供一些素材，本书除了对《C#网络编程及应用》教材中的全部习题都给出了参考解答外，还介绍了 3 个应用编程的实际例子，可以帮助读者通过调试提高实际动手能力，也可以作为学生结合所学内容进行上机综合练习的题目。

全书所有的程序代码和习题参考答案均在 Visual Studio.NET 2003 下调试通过。读者学习时最好按部就班地独立做下去，一行行地把代码输入进去，即使已经有程序的源代码，也不要图省事，简单地利用复制、粘贴的办法输入代码，只有这样才能体会编程的细节，才能真正提高实际编程的水平。

本书由刘瑞新主编。全书共 4 章，马骏编写了第 1～3 章和第 4 章的第 1～5 节的习题及参考解答，何欣编写了第 4 章第 6 节和第 13 节的习题及参考解答，张磊编写了第 4 章第 7～8 节的习题及参考解答，周珂编写了第 4 章第 9～12 节的习题及参考解答，赵新强编写了第 4 章第 14～17 节的习题及参考解答。韩道军、张磊、毋琳、安广伟、贾培艳、贾继凯、符松、李军伟等也参与了本书部分代码的编写、调试、整理，以及录入校对等工作。

由于编者水平有限，书中肯定还有许多不妥之处，望广大读者不吝赐教。本书中的源文件均可到 www.cmpbook.com 网站免费下载。

编　者

目　录

第 1 章　Win Forms 数据库应用程序开发实例

本实例是一个简化了的公司员工管理程序，主要涉及到数据库方面的操作，包括对个人记录进行显示、添加、修改、删除、导入照片、打印及打印预览等。

1.1　数据库结构

在本例子中，主要用到两个数据表，一个是编码数据对照表，用于保存各部门对应的编码；另一个是个人情况表，表中包含了有代表性的几种数据类型。可以直接在 VS.NET 开发环境下创建一个数据库，然后创建对应的两个表；也可以用 SQL Server 查询分析器或者企业管理器创建数据库和数据表。

1) 建立名为 MyDatabase 的 SQL Server 数据库。

2) 建立编码数据对照表。

表名：department

结构：编码，char，2　主键

名称，char，10

记录：01 人事部

02 生产部

03 计划部

04 财务部

05 销售部

3) 建立个人情况表。

表名：personal

结构：编号，char，5　主键

姓名，char，8

性别，char，2

工资，int，4

出生日期，datetime，8

照片，image，16

1.2　界面设计

1) 创建一个 Windows 应用程序项目，设计窗体界面如图 1-1 所示。

主要控件属性如表 1-1 所示。

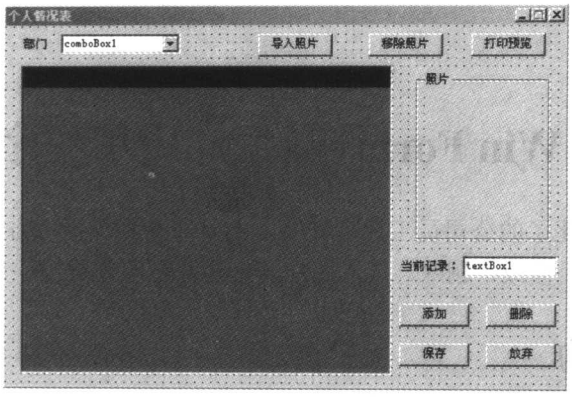

图 1-1　设计窗体界面

表 1-1　主要控件的属性

控　件	属　性	属 性 值
Form	FormBorderStyle	FixedDialog
	MaximizeBox	False
	StartPosition	CenterScreen
Button	Name	buttonAppend
	Text	添加
Button	Name	buttonDelete
	Text	删除
Button	Name	buttonSave
	Text	保存
Button	Name	buttonCancel
	Text	放弃
Button	Name	buttonFromPhoto
	Text	导入照片
Button	Name	buttonClearPhoto
	Text	移除照片
Button	Name	buttonPrint
	Text	打印预览
ComboBox	Name	comboBox1
GroupBox	Name	groupBox1
	Text	照片
PictureBox（在 groupBox1 内）	Name	pictureBox1
TextBox	Name	textBox1
DataGrid	Name	dataGrid1

2）单击 DataGrid1 的“TableStyles”属性，添加 DataGridTableStyle1，将其“MappingName”属性设置为“personal”，如图 1-2 所示。

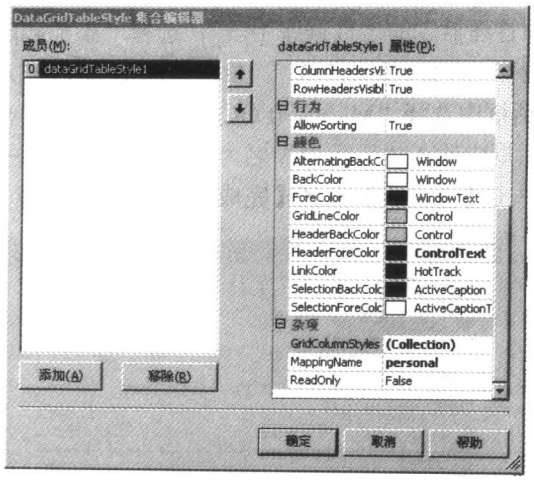

图 1-2　添加 DataGridTableStyle1

单击“GridColumnStyles”属性，添加 1 列，设置“HeaderText”属性为“编号”，清除“nullText”属性值，设置“mappingName”属性为“编号”。如果需要，也可以修改“width”属性为该列设置合适宽度，如图 1-3 所示。按照上述方法再添加 4 列，分别对应“姓名”、“性别”、“工资”、“出生日期”，单击【确定】。

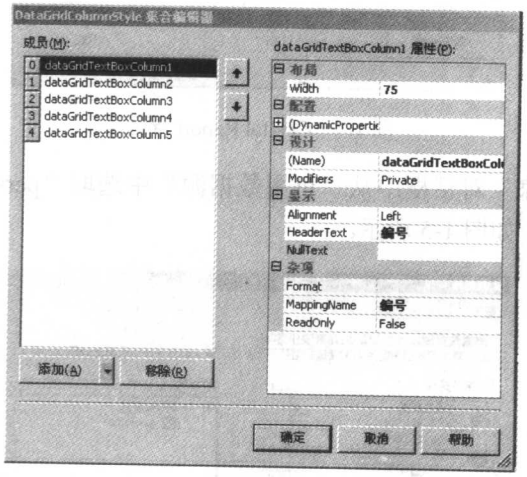

图 1-3　“GridColumnStyles”属性

1.3　设计报表

由于水晶报表具有非常强大的功能，所以这个例子中，仍然使用水晶报表作为打印预览的界面。

1) 右击“解决方案资源管理器”中的项目名，在弹出的快捷菜单中选择“添加新项”，然后选择“Crystal Report”，输入报表名 personal.rpt，单击【打开】。

2) 在弹出的界面中选择“作为空白报表”，单击【确定】，如图 1-4 所示。

3) 右击“字段资源管理器”的“数据库字段”，选择“添加删除数据库”。在弹出的“数据库专家”对话框中选择“OLE DB (ADO)”。单击“OLE DB (ADO)”左侧加号，在弹出的“OLE DB (ADO)”对话框中选择“Microsoft OLE DB Provider for SQL Server”。单击【下一步】，在弹出的连接对话框中输入“服务器”名为“localhost”，选中“集成安全”，选择数据库名“MyDatabase”。单击【下一步】直到【完成】，最后返回到“数据库专家”对话框。

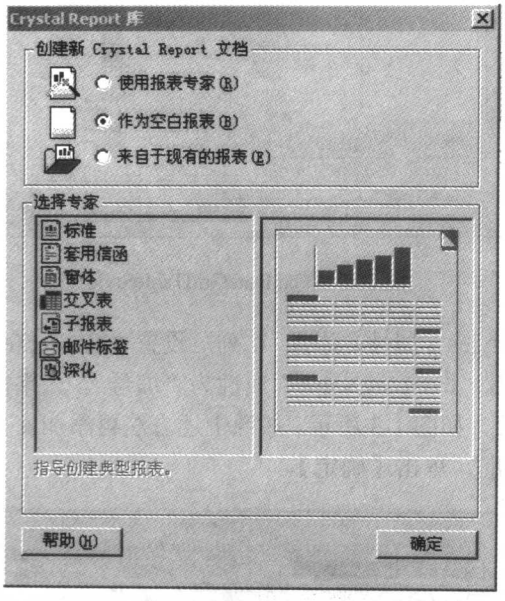

图 1-4　Crystal Report 界面

4) 在“数据库专家”对话框中从“可用数据源”中选取“personal”，添加到“选定的表”中，单击【确定】，如图 1-5 所示。

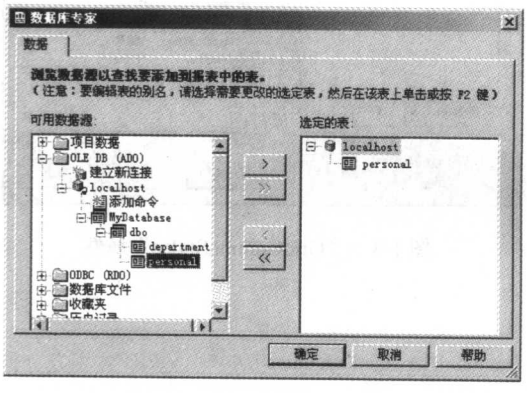

图 1-5　选定数据表

5) 设计报表的形式，如图 1-6 所示。

6) 添加新的 Windows 窗体，名字为“print.cs”，设置该窗体的“WindowState”属性为“Maximized”。然后向窗体添加一个 CrystalReportViewer 控件，并设置其“Dock”属性为“Fill”，“ReportSource”属性为带路径的“personal.rpt”。

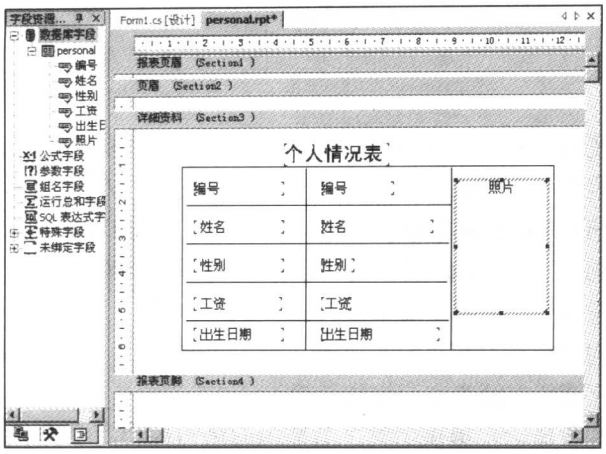

图 1-6　设计报表

注意：使用绝对路径并不是一个好方法，实际设计应用程序时可以将.rpt 文件与.exe 文件放在同一目录下，然后在代码中利用 Application.StartupPath 得到.rpt 文件所在路径。

1.4　创建惟一性约束

为了获取或设置一个惟一值，要对属性列进行惟一性约束。本示例将这些惟一性约束代码添加在构造函数中。

```
//创建惟一性约束，防止有重复的人员编号
UniqueConstraint constraint = new UniqueConstraint(
    new DataColumn[]{dataset.Tables[0].Columns[0]});
dataset.Tables[0].Constraints.Add(constraint);
```

UniqueConstraint 对象（可分配给 DataTable 中的单独一列或一组列）确保指定的某列或多个列中的所有数据对于每行都是惟一的。由于可以同时为多列创建惟一性约束，所以构造函数中使用 DataColumn 类型的数组作为传递的参数。

在构造函数代码中，还有一个对数据源的绑定。DataGrid.SetDataBinding 方法用于在运行时设置 DataSource 和 DataMember 属性，一般用于动态地改变数据源的情况。在运行时，如果使用 this.dataGrid1.DataSource 改变数据源，可能会产生不固定的莫名其妙的异常。该方法的调用形式为：

```
public void SetDataBinding(object dataSource, string dataMember);
```

其中，参数 dataSource 指示 System.Windows.Forms.DataGrid 控件的数据源；dataMember 指示要绑定的 DataSource 中的属性。

用 DataView 作为数据源的目的是为了不让 DataGrid 中自动添加新行。如果用 DataTable 作为绑定的数据，系统会自动添加一新行（前面有*的行）。当不希望显示该行时可以用 DataView 作为数据源，并将其 AddNew 属性设置为 false 即可。注意，当 DataSource 为 DataView 或 DataTable 时，需要将 DataMember 设置成空字符串（""）。

1.5 导入照片

在数据库中，是使用 Image 类型保存照片的。在 C#中直接使用 Sql 语句进行图像的存取操作虽然也允许，但效率不是太高，因此本实例不采用这种方法。在这个开发实例中，先用 SqlDataAdapter 的 Fill 方法将从数据库中选择的内容填充到 DataSet 的 DataTable 中，然后对 DataTable 中的内容进行添加、修改、显示等操作。由于 C#对数据库中的图像处理提供了强大的支持，因此操作方便，使用也简单。

保存到数据库中的图像信息可以看作是一系列二进制字节，由于不存在字节数组和图像格式的直接转换，因此要依靠流来完成这种转换。

流是字节序列的抽象概念，提供了对字节数组的读写操作。可以利用流的方法，将图像信息转换成字节数组存入数据库中，显示时再将字节数组转换成相应的图像。

流有以下几种基本操作：

1) 读取流。读取是从流到数据结构（如字节数组）的数据传输。

2) 写入流。写入是从数据结构到流的数据传输。

3) 查找。查找是对流内的当前位置进行查询和修改。查找功能取决于流具有的后备存储区类型。

Stream 是所有流的抽象基类。其他流都是继承于 Stream 类。对文件、输入/输出设备进行操作一般使用 FileStream 流；对内部进程进行操作一般使用 MemoryStream 流；而对 TCP/IP 套接字进行操作，一般使用 NetworkStream 流。

这里使用 MenoryStream 流实现字节数组和图像格式的转换。

```
private void buttonFromPhoto_Click(object sender, System.EventArgs e)
{
    string bianhao=this.textBox1.Text;
    if(bianhao.Length<5)
    {
        MessageBox.Show("请先选择记录（点击每行左边的灰色部分）。");
        return;
    }
    bianhao=bianhao.Substring(0,5);
    OpenFileDialog myfile=new OpenFileDialog();
    if(myfile.ShowDialog() == DialogResult.OK)
    {
        try
        {
            //根据打开的图像文件创建原始图像大小的 Bitmap 对象
            Bitmap bitmap=new Bitmap(myfile.OpenFile());
            //缩放到 112 像素宽，144 像素高
            Bitmap image=new Bitmap(bitmap,112,144);
            //创建内存流
            MemoryStream memStream=new MemoryStream();
            //将图像以 jpeg 格式保存到内存流中
```

```
                image.Save(memStream,System.Drawing.Imaging.ImageFormat.Jpeg);
                //将内存流数据写入字节数组
                byte[] bytes=memStream.ToArray();
                //关闭内存流
                memStream.Close();
                //求当前行号
                int num=this.dataGrid1.CurrentRowIndex;
                //将字节数组保存到当前行的“照片”字段中
                dataset.Tables[0].Rows[num]["照片"]=bytes;
            }
            catch(Exception err)
            {
                MessageBox.Show(err.Message);
            }
            GetImage(this.textBox1.Text.Substring(0,5));
        }
    }
```

代码中首先根据用户选择的图像文件创建一个 Bitmap 对象，然后将该图像缩放到指定的大小。这样可以适用于任何分辨率的数码相机拍摄的相片，惟一的要求是照片必须是纵向拍摄的（高＞宽），否则缩放后会变得很瘦。得到缩放的图像后，再通过内存流写入到字节数组中，并将该字节数组赋给 DataTable 中当前记录的“照片”字段即可。

1.6 显示照片

显示照片与导入照片的过程相反，首先通过显示转换将“照片”字段转换为字节数组，然后通过内存流得到 Bitmap 对象，并将此 Bitmap 对象作为 PictureBox 的 Image 属性即可。

```
private void GetImage(string str)
{
    //释放占用的资源
    if (this.pictureBox1.Image != null)
    {
        this.pictureBox1.Image.Dispose();
        }
    int num=this.dataGrid1.CurrentRowIndex;
    if(num==-1)
    {
        return;
    }
    if(dataset.Tables[0].Rows[num]["照片"]!=Convert.DBNull)
    {
        byte[] bytes=(byte[])dataset.Tables[0].Rows[num]["照片"];
        MemoryStream    memStream=new MemoryStream(bytes);
        Bitmap bitmap = new Bitmap(memStream);
```

```
            memStream.Close();
            this.pictureBox1.Image = bitmap;
        }
        else
        {
            this.pictureBox1.Image = null;
        }
    }
```

dataGrid1.CurrentRowIndex 指用户单击后得到的当前行号，如果用户单击的位置不在DataGrid 网格范围内，该值返回-1，否则返回鼠标单击所在的行号。得到当前行后，判断“照片”字段是否为空。注意，Convert.DBNull 表示 SQL Server 数据表中的 null，不能在 C#代码中直接写为 null。

1.7 移除照片

移除照片就比较简单了，只需要把选中记录的“照片”字段设置为空即可。但要注意，这里的“空”是指 SQL Server 中的 null，该值在 C#中表示为 Convert.DBNull。

```
private void buttonClearPhoto_Click(object sender, System.EventArgs e)
{
    int num=this.dataGrid1.CurrentRowIndex;
    if(num>-1)
    {
        dataset.Tables[0].Rows[num]["照片"]=Convert.DBNull;
        GetImage(this.textBox1.Text.Substring(0,5));
    }
}
```

1.8 自动生成编号

在向数据库中添加员工信息时，每一个员工都有相应的编号。编号的前两位表示所属部门编码，后三位表示员工在该部门的编号。为了提高用户录入速度，可以在系统中自动生成一些字段信息。实现代码如下：

```
private void buttonAppend_Click(object sender, System.EventArgs e)
{
    string bh=this.comboBox1.SelectedItem.ToString().Substring(1,2)+"000";
    int num=Int32.Parse(bh)+1;
    DataRow newRow=dataset.Tables[0].NewRow();
    while(true)
    {
        try
        {
            newRow["编号"]=num.ToString("d5");
```

```
                newRow["性别"]="男";
                dataset.Tables[0].Rows.Add(newRow);
                break;
            }
            catch
            {
                num++;
                continue;
            }
        }
    }
```

自动添加时，首先从所在部门的编号 000 开始，依次检查数据表中是否已经有该员工编号。如果已经有此员工，再向数据表中添加该编号就会产生异常。根据这个原理，就可以在异常中将编号加 1，然后再添加，直至成功为止。

当然，从最大编号加 1 开始检查速度会快一些。实际应用时，程序员可以根据公司要求决定是从 0 开始检查还是从最大编号开始检查。

1.9 删除一条或多条记录

在简单的系统中，即使不实现这部分代码，用户直接按〈Delete〉键也能完成删除一条或多条记录的操作。但是如果用户不知道按哪个键怎么办？按错键了怎么办？所以最好还是有一些明显的提示，如让用户单击【删除】按钮完成删除功能，并提示用户是否确实要删除，以防止出现意外的数据丢失。

为了让程序简单些，本例中并没有考虑用户按错键的情况，而是采用较直观的方法，让用户单击【删除】按钮完成删除操作。

如何判断用户选择的是哪些行呢？在 DataGrid 控件中，有一个 IsSelected 方法，该方法返回用户是否选择了指定的行。利用这个方法，就可以通过一个循环检查每一行是否被选中。

```
private void buttonDelete_Click(object sender, System.EventArgs e)
{
    int num=dataview.Count-1;
    if(num>-1)
    {
        ArrayList ar=new ArrayList();
        for(int i=num;i>=0;i--)
        {
            if(this.dataGrid1.IsSelected(i))
            {
                ar.Add(i);
            }
        }
        if(ar.Count>0)
        {
```

```
                if(MessageBox.Show("确实要删除选中的"+ar.Count.ToString()
                    +"行吗？ 注意：删除后就无法恢复了！","警告"
                    ,MessageBoxButtons.YesNo,MessageBoxIcon.Question)
                    ==DialogResult.Yes)
                {
                    foreach(int i in ar)
                    {
                        dataview.Delete(i);
                    }
                    adapter.Update(dataset,tableName);
                }
            }
            else
            {
                MessageBox.Show("请先用鼠标点击左边灰色部分选择要删除的"+
                    "行(可以在按住〈Ctrl〉键或者〈Shift〉键的同时用鼠标选中多行)，"+
                    "然后再删除。","提示");
            }
        }
    }
```

例子中采用删除后直接保存的方法，这是为了保证让 DataGrid 中的行与 DataSet 中数据表的行一一对应。否则删除后再单击选择行时，可能会出现一些异常。

1.10 保存数据

由于操作的数据全部在内存中，并未对数据库进行修改。因此需要根据修改的内容更新数据源，即将更改保存到数据库中。可以用 SqlDataAdapter 的 Update 方法来完成此操作。

```
private void buttonSave_Click(object sender, System.EventArgs e)
{
    try
    {
        adapter.Update(dataset,tableName);
        MessageBox.Show("保存成功。","恭喜");
    }
    catch(Exception err)
    {
        MessageBox.Show(err.Message,"保存信息出错！",
        MessageBoxButtons.OK,MessageBoxIcon.Exclamation);
    }
}
```

1.11 通过下拉列表框选择不同部门

一旦部门编码确定以后，就不能随意更改了，除非尚无该部门编码的人员。在这个例子

中，用户一旦选择了部门编码，屏幕上就能及时显示当前部门的员工信息。ComboBox 控件有一个 SelectionChangeCommitted 事件，当用户改变并提交了 ComboBox 中的当前选项时会触发该事件。可以利用这个事件即时设置部门过滤条件。

```
private void comboBox1_SelectionChangeCommitted(object sender, System.EventArgs e)
{
    if(dataset.HasChanges())
    {
        try
        {
            adapter.Update(dataset,tableName);
        }
        catch(Exception err)
        {
            MessageBox.Show(err.Message+"\n\n 请先修改错误！","保存信息出错！",
                MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation);
            return;
        }
    }
    string str="-1";
    if(this.comboBox1.SelectedIndex>=0)
    {
        str=this.comboBox1.SelectedItem.ToString().Substring(1,2);
        dataview.RowFilter="substring(编号,1,2)='"+str+"'";
    }
}
```

代码中首先保存原来修改的信息，如果保存成功，则直接修改过滤条件。否则提示用户异常信息，并继续让用户修改当前部门的信息，正确保存后才能处理其他部门的人员信息。

注意，在 C#中，字符串索引是从 0 开始编号的，而在 SQL Server 中则是从 1 开始编号的。因此，提供的过滤条件求子串的 T-SQL 函数是 substring(编号,1,2)，意思是取字符串的前两个字符。

1.12 防止直接关闭窗体引起数据丢失

添加关闭窗体前激发的事件代码：

```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    if(dataset.HasChanges())
    {
        if(MessageBox.Show("数据尚未保存，退出会引起修改无效！确实要退出吗？"
            ,"小心",MessageBoxButtons.YesNo,MessageBoxIcon.Question)==DialogResult.Yes)
        {
```