2025-2026 学年第一学期

《.NET 体系结构与应用开发实验》实验指导任务书

实验序号	第 11 次	所在学院	软件工程学院
实验名称	Windows 高级控件（1）	任课教师	陈建国
实验地点	A310	年级班级	2024 级

一、实验目的

1. 学习掌握 ListView 列表视图控件

二、实验设备

主流 PC 机一台，要求安装 windows 操作系统。 Visual Studio 工具。

三、实验内容

任务 1、ListView 控件基本操作

任务 2、ListView 实现客户信息管理

四、实验步骤

任务 1、ListView 控件基本操作

学习 ListView 控件的基本操作，包括设置 ListView 控件外观、添加项、删除项和获取选中项改变事件。首先要创建一个 Windows 应用程序，启动 Visual Studio 开发环境，创建一个“Windows 窗体应用程序”项目，打开 Windows 应用程序的开发界面，默认显示 Form1 窗体。

1. 添加 ListView 等控件

打开工具栏，从工具栏中选中 ListVew 控件，按住该控件并将此拖放到 Form1 窗体指定位置，放开鼠标即可。添加 ListView 控件的过程如图 1 所示。

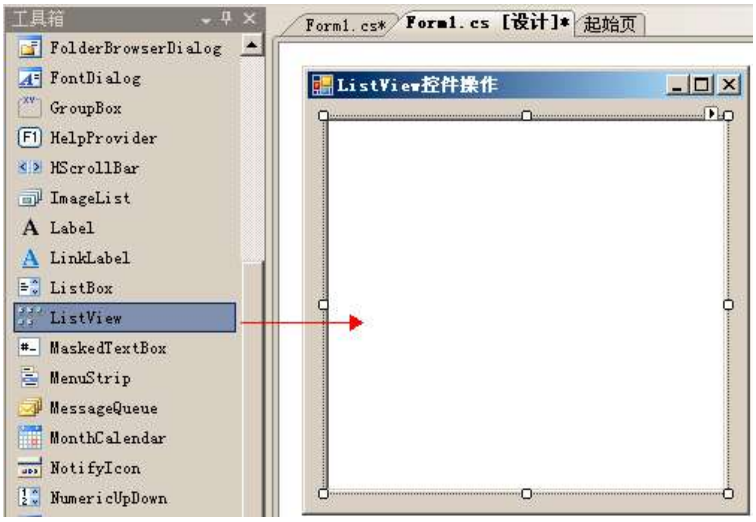

图 1 添加 ListView 控件

然后继续在 Form1 窗体中添加三个按钮控件，分别作为“添加项”、“移除项”和“清空项”。各个控件的属性设置说明如表 1 所示。

表 1 Form1 窗体各按钮控件属性设置说明

控件	属性	属性值	控件	属性	属性值
按钮 1	Name	btn_AddItem	按钮 3	Name	btn_ClearItem
	Text	添加项		Text	清空项
按钮 2	Name	btn_RemoveItem			
	Text	移除项			

2. 设置 ListView 外观

以 ListView 控件的详细信息（Detail）视图显示方式为例讲解 ListView 外观的具体设置。ListView 控件的 Detail 视图外观与普通表格相同，主要用于数据信息列表的显示。在 Detail 视图中，每个列表项以数据行的方式显示，列表项的子项信息以数据列的方式显示，并且可以设置各列的标题。ListView 外观设置的具体方法如下：

1）打开 ListView 控件的属性面板，首先设置 View 属性的属性值为 Details。

2)然后设置各列标题，找到 Columns 属性，单击其右侧的“…”图标，如图 7-3 所示。弹出 ColumnHeader（列标题）集合编辑器对话框，如图 3 所示。在该对话框中可以编辑各列标题、各列宽度以各列的显示顺序等。单击对话框左侧的【添加】按钮，即可添加一个列，默认名称为 columnHeader1，然后在对话框右侧可以设置该列的 Text 显示文本、Width 宽度等属性，还可以通过 DisplayIndex 属性设置各列的显示顺序。

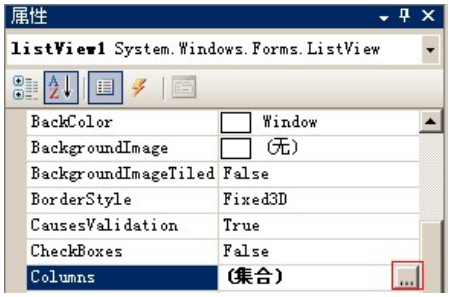

图 2 设置 Columns 属性

图 3 ColumnHeader 集合编辑器

3）设置各列标题后的 ListView 控件如图 4 所示，从图中可以看到 ListView 控件已经显示各列的标题，但是没有显示单元格分隔线条（网络线）。设置 GridLine 属性的属性值为 True，可以显示 ListView 网格线，如图 5 所示。

图 4 设置列标题后的 ListView 控件

图 5 设置 GridLine 属性后的 ListView 控件

4）设置 ListView 控件的其他属性，分别设置 FullRowSelect 属性的属性值为 True，设置 MultiSelect 属性的属性值为 False。

至此，ListView 控件外观设置完成，此外观与普通表格相同。

3. 添加列表项

在项目中继续实现添加列表项的功能，在图 5 中所示的 Form1 窗体中，双击【添加项】按钮，在【添加项】按钮单击事件中编写添加 ListView 列表项的函数。具体代码如下：

```
//添加项
private void btn_AddItem_Click(object sender, EventArgs e)
{
    ListViewItem myitem = new ListViewItem("1");  //创建一个 ListViewItem 项，并为第 1 列
赋值
    myitem.SubItems.Add("张宏");    //为第 2 列赋值
    myitem.SubItems.Add("男");      //为第 3 列赋值
    myitem.SubItems.Add("20");      //为第 4 列赋值
    myitem.SubItems.Add("保密...");  //为第 5 列赋值

    listView1.Items.Add(myitem);  //将新建项添加到 ListView 控件中
}
```

首先创建一个列表项 ListViewItem，并为第一列赋值，然后分别为该列表项的子项信息（即第 2 列以及其他列）赋值，最后将该列表项添加到 LsitView 控件中。

代码编写完成后，保存项目文件并运行程序，程序运行效果如图 6 所示。在程序运行界面中，单击【添加项】按钮，将在 ListView 控件中出现一个列表项。

图 6 添加 ListViewItem 项

4. 移除列表项

在项目中继续实现移除列表项的功能，在图 5 中所示的 Forml 窗体中，双击【移除项】按钮，在【移除项】按钮单击事件中编写移除 ListView 列表项的函数。具体代码如下：

```
//移除项
private void btn_RemoveItem_Click(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count > 0)  //如果有选中项
    {
        foreach (ListViewItem myitem in listView1.SelectedItems)  //遍历每一个选中项
        {
            listView1.Items.Remove(myitem);  //移除项
        }
    }
    else  //如果没选中项
    {
        MessageBox.Show("请先选择要移除的项");
    }
}
```

首先判断是否有选中要移除的项，LsitView 控件可以设置是否支持一次选择多个列表项（通过 MultiSelect 属性设置），然后遍历选中的每一个列表项，并将每个选中项从 ListView 控件中移除。

代码编写完成后，保存项目文件并运行程序，程序运行效果如图 7 所示。在程序运行界面中，先选中要移除的列表项，然后单击【移除项】按钮，则选中的列表项将从 ListView 控件中移除。

图 7 移除 ListViewItem 项

5. 清空列表项

继续实现清空列表项的功能，在图 5 中所示的 Form1 窗体中，双击【清空项】按钮，在【清空项】按钮单击事件中编写清空 ListView 列表项的函数。具体代码如下：

```
//清空项
private void btn_ClearItem_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();  //清空项
}
```

代码编写完成后，保存项目文件并运行程序，程序运行效果如图 8 所示。在程序运行界面中，单击【清空项】按钮，则 ListView 控件中的所有列表项都被清空了。

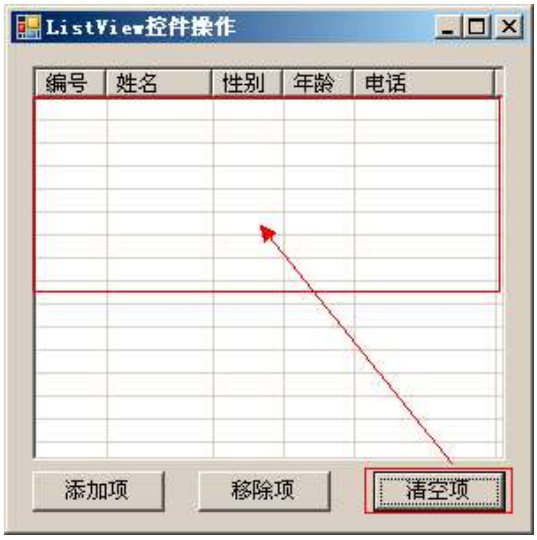

图 8 清空所有 ListViewItem 项

6. 获取当前选中项的信息

ListView 控件的 SelectedIndexChanged 事件是指当选中项改变时触发该事件，常用于当选中某行数据时实现相应功能。

继续实现获取当前选中项的功能，在 Form1 窗体中分别添加一个 Label 控件和一个 TextBox 控件，设置 Label 控件的 Text 属性值为“当前选中项：”。

然后选中 ListView 控件，为 ListView 控件编写 SelectedIndexChanged 事件，在属性面板中单击【事件】选项，如图 9 所示。进入事件列表面板，双击 SelectedIndexChanged 事件，进入代码编写窗口，编写相应代码如下。

```
//选中项改变事件
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count > 0)  //如果有选中项
    {
        ListViewItem myitem = listView1.SelectedItems[0]; //获取选中的第一行（需将
MultiSelect 属性设置为 False，即不能选中多行）
        textBox1.Text = myitem.SubItems[1].Text;  //将选中行的第二列的值赋值给文本框
    }
}
```

首先判断是否有选中列表项，接着获取选中的第一个列表项（需将 MultiSelect 属性设置为 False，即不能选中多行）。然后将选中行的第二列的值（即“姓名”列的值）赋值给文本框。

代码编写完成后，保存项目文件并运行程序，程序运行效果如图 10 所示。在程序运行界面中，首先单击【添加项】按钮，为 ListView 控件添加多个列表项。然后在 ListView 控件中单击各列表项信息，此时选中项的第二列信息，即“姓名”列的信息就会显示在文本框中。

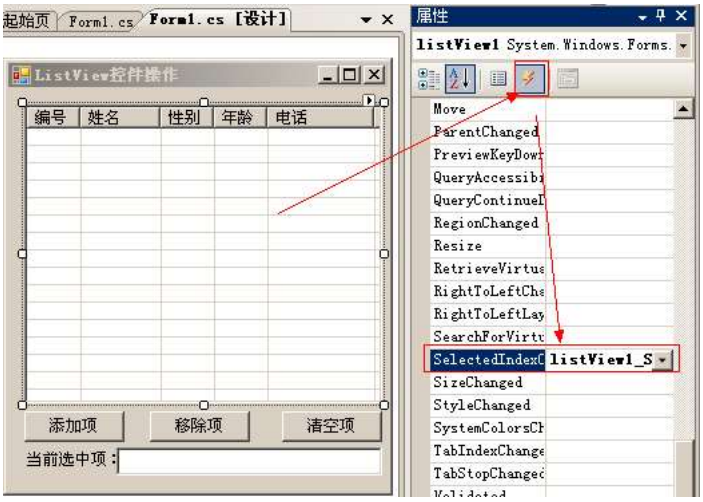

图 9 添加 SelectedIndexChanged 事件

图 10 ListView 选中项改变事件运行效果

任务 2、ListView 实现客户信息管理

使用 ADO.NET 知识和 List 控件相结合，实现客户信息管理模块的开发，包括添加客户信息、修改客户信息、删除客户信息以及客户信息的显示等功能。

【实现步骤】

步骤 1：启动 SQL Server，创建一个数据库，命名为 DB_CASE0705。然后在该数据库中创建一个客户信息表，命名为 Customer_Info。客户信息表的结构如下表 7-11 所示。

表 7-11 客户信息表的结构

字段名称	数据类型	说明
CustomerID	int	客户编号，主键、自增
CustomerName	varchar(20)	客户姓名
Company	varchar(50)	单位
Sex	varchar(2)	性别
Age	int	年龄
Telephone	varchar(20)	联系电话
Address	varchar(200)	联系地址

创建数据库及数据表的 SQL 语句如下：

```
--创建数据库
create database DB_CASE0705;
--使用数据库
use DB_CASE0705;

--创建表
create table Customer_Info
(
CustomerID  int primary key identity,
CustomerName    varchar(20),
Company varchar(50),
Sex varchar(2),
Age int,
Telephone   varchar(20),
Address varchar(200)
)
```

步骤 2：创建一个 Windows 窗体应用程序，将 Form1 窗体设置为客户信息管理界面，打开 Form1.cs 的窗体设计器，为窗体添加 2 个分组面板（GroupBox），分别用于编辑客户信息和显示客户信息列表。

接着在 groupBox1 分组面板中添加 6 个标签控件（Label）、4 个文本框控件（TextBox）、2 个单选按钮控件（RadioButton）、1 个数字选择控件（NumericUpDown）、2 个按钮控件（Button）。

在 groupBox2 分组面板中添加 1 个列表视图控件（ListView）、1 个按钮控件（Button）。最后添加两个标签控件，其中一个标签设置其名称为 lbl_status，用于显示编辑状态。另一个标签设置其名称为 lbl_note，用于显示操作结果。

添加完各控件后，需要为各控件设置相应的属性值。客户信息管理界面各控件的属性设置说明如表 2 所示。

表 2 客户信息管理界面控件属性设置说明

控件	属性	属性值	控件	属性	属性值
groupBox1	Text	编辑客户信息　状态：	状态标签	name	lbl_Status
groupBox2	Text	客户信息列表	结果提示标签	name	lbl_Note
姓名文本框	name	txt_Name	保存按钮	Name	btn_Save
单位文本框	name	txt_Company		Text	保存
单选按钮 1	name	rbtn_Sex1	取消按钮	Name	btn_Cancel
	Text	男		Text	取消
单选按钮 2	name	rbtn_Sex2	删除按钮	Name	btn_Del
	Text	女		Text	删除
NumericUpDown 年龄框	name	nudown_Age	listView1	name	lv_Customer
	Minimun	0		View	Details
	Maximum	100		MultiSelect	False
电话文本框	name	txt_Telephone		FullRowSelect	True
地址文本框	nme	txt_Address		GridLines	True
				Columns	编号、姓名等

客户信息管理界面设计效果如图 11 所示。

图 11 客户信息管理界面设计效果

步骤 3：添加客户信息

本步骤实现添加客户信息的功能，双击【保存】按钮，为该按钮编写 Click 事件，系统自动打开 Form1.cs 代码界面，并且自动生成 btn_Save_Click 事件函数。在 Form1.cs 代码界面中，首先引入 System.Data.SqlClient 命名空间；然后在 btn_Save_Click 函数中编写客户信息添加程序。具体代码如下：

```
//保存按钮
private void btn_Save_Click(object sender, EventArgs e)
{
    string name = txt_Name.Text.Trim();                    //姓名
    string company = txt_Company.Text.Trim();              //公司
    string sex = rbtn_Sex1.Checked == true ? "男" : "女"; //性别
    string age = nudown_Age.Value.ToString();              //年龄
    string telephone = txt_Telephone.Text.Trim();          //电话
    string address = txt_Address.Text.Trim();              //地址

    if (name == "")  //姓名为空
    {
        lbl_Note.Text = "姓名不能为空！";
        lbl_Note.ForeColor = Color.Red;
        txt_Name.Focus();
    }
    else if (lbl_Status.Text == "添加")  //如果是“添加”状态
    {
        //1、创建数据库连接
        SqlConnection conn = new SqlConnection("Data Source=captainstudio;
database=db_case0705; Integrated Security=SSPI");

        //2、打开连接
        conn.Open();

        //3、数据插入语句
        string str = string.Format("insert into Customer_Info values('{0}','{1}',
'{2}',{3},'{4}','{5}')",name,company,sex,age,telephone,address);
```

```
        //4、创建执行对象
        SqlCommand cmd = new SqlCommand(str, conn);

        //5、执行操作，返回受影响的行数
        int i = cmd.ExecuteNonQuery();

        //6、关闭连接
        conn.Close();

        //7、处理结果
        if (i > 0)
        {
            lbl_Note.Text = "恭喜您，客户信息添加成功！";
            lbl_Note.ForeColor = Color.Blue;
            ClearTexBox();   //调用函数，清空各控件
        }
        else
        {
            lbl_Note.Text = "对不起，客户信息添加失败！";
            lbl_Note.ForeColor = Color.Red;
        }
    }
}
```

```
//清空各控件
protected void ClearTexBox()
{
    txt_Name.Text = "";
    txt_Company.Text = "";
    txt_Telephone.Text = "";
    txt_Address.Text = "";
    rbtn_Sex1.Checked = true;
    nudown_Age.Value = 0;
    lbl_Status.Text = "添加";
}
```

代码编写完成后，保存项目文件并运行程序。在程序界面中分别输入姓名、单位等客户信息，单击【保存】按钮，系统将客户信息保存到数据库中，并显示“客户信息添加成功”的提示信息，程序运行效果如图 12 所示。

图 12 添加客户信息

说明：ClearTexBox()函数的功能是清空各控件的值，该函数将用于当客户信息添加成功、客户信息修改成功和客户信息删除成功后调用执行。如果没有该函数，客户信息添加完成后，各文本框的内容没有自动清空，当用户再次单击【保存】按钮时，依然可以将同一个客户信息又添加一次，而且当用户想添加下一位客户信息时，需要手动将每个文本框的内容清空。

至此，添加客户信息的功能已基本完成，但还存在一处问题，客户信息添加完成后，并没有在界面下方的 ListView 控件中自动显示出来。因此，接下来实现客户信息的加载显示。

步骤 4：加载客户信息

接着编写加载客户信息的功能代码，定义一个加载客户信息的函数，然后分别在以上四个事件中调用该函数。

首先实现打开客户信息管理界面时加载客户信息并显示在 ListView 控件中的功能,选中 Form1 窗体，为 Form1 控件编写 Load 事件，在属性面板中单击【事件】选项，进入事件列表面板。双击 Load 事件，进入代码编写窗口。首先定义一个 DataBind_Customer 函数，从数据库中查询客户信息并绑定到 ListView 控件，然后在 Form1_Load 函数中调用该函数，实现加载客户信息功能。具体代码如下：

```
//加载客户信息
protected void DataBind_Customer()
{
    //1、创建并打开数据库连接
    SqlConnection conn = new SqlConnection("Data Source=captainstudio;
database=db_case0705; Integrated Security=SSPI");
    conn.Open();

    //2、客户端发出请求：数据查询语句
    string str = "select * from Customer_Info";

    //3、创建执行对象
    SqlDataAdapter da = new SqlDataAdapter(str, conn);

    //4、创建临时数据表
    DataTable dt = new DataTable();

    //5、执行查询，返回结果，填充到临时表中
    da.Fill(dt);

    //6、关闭连接
    conn.Close();

    //7、显示结果
    lv_Customer.Items.Clear();  //先清空列表视图控件中现有行
    foreach (DataRow dr in dt.Rows)
    {
        ListViewItem myitem = new ListViewItem(dr["CustomerID"].ToString());  //创建一
个 ListViewItem 项，并为第 1 列赋值，客户编号
        myitem.SubItems.Add(dr["CustomerName"].ToString()); //第 2 列，姓名
        myitem.SubItems.Add(dr["Company"].ToString());      //第 3 列，单位
        myitem.SubItems.Add(dr["Sex"].ToString());          //第 4 列，性别
        myitem.SubItems.Add(dr["Age"].ToString());          //第 5 列，年龄
        myitem.SubItems.Add(dr["Telephone"].ToString());    //第 6 列，电话
        myitem.SubItems.Add(dr["Address"].ToString());      //第 7 列，地址

        lv_Customer.Items.Add(myitem);  //将新建项添加到 ListView 控件中
    }
}

//窗体加载事件
private void Form1_Load(object sender, EventArgs e)
{
    DataBind_Customer(); //加载客户信息
}
```

代码编写完成后，保存项目文件并运行程序。读者可以看到，当程序界面打开时，系统将自动从数据库中查询客户信息并显示到 ListView 控件中，程序运行效果如图 13 所示。

图 13 加载客户信息

接着，完善步骤 3 中的客户信息添加功能。当客户信息添加成功后，需要重新加载客户信息到 ListView 控件。在“7、处理结果”的代码段中增加一行“**DataBind_Customer();**”的语句，调用该函数，实现重新加载客户信息的功能。具体代码如下：

```
...
//7、处理结果
 if (i > 0)
 {
     lbl_Note.Text = "恭喜您，客户信息添加成功！";
     lbl_Note.ForeColor = Color.Blue;
     ClearTexBox();   //调用函数，清空各控件
     DataBind_Customer();  //重新加载客户信息
 }
 ...
```

代码编写完成后，保存项目文件并运行程序。在程序界面中分别输入姓名、单位等客户信息，单击【保存】按钮，系统将客户信息保存到数据库中，同时将刚添加的客户信息即时信息到 ListView 控件中。

注意：读者看到的效果是每添加一位客户信息，界面下方的 ListView 控件就多一行，很多读者就会误解为 ListView 控件只是改变一行数据。其实不然，每添加一位客户信息，ListView 控件就会先全部清空现有列表视图中的数据，从数据库中重新查询全部的客户信息并显示。

步骤 5：修改客户信息

本步骤实现修改客户信息的功能，具体分为两个部分实现。

首先，在客户信息列表中选择要修改的客户，系统将选中的客户信息显示到各文本框中，并且将当前状态由“添加”改为“修改”。具体实现过程为：选中 ListView 控件，为 ListView 控件编写选中项改变事件，在属性面板中单击【事件】选项进入事件列表面板，双击 SelectedIndexChanged 事件，进入代码编写窗口，编写相应代码如下。

```
string customerid = "";  //定义全局变量，用于存储客户编号

//客户信息选中项改变事件
private void lv_Customer_SelectedIndexChanged(object sender, EventArgs e)
{
    if (lv_Customer.SelectedItems.Count > 0)  //如果有选中项
    {
        ListViewItem myitem = lv_Customer.SelectedItems[0]; //获取选中的第一行（一次只
能选一行）

        customerid = myitem.SubItems[0].Text; //将选中行第 1 列的值赋值全局变量，客户编号
        txt_Name.Text = myitem.SubItems[1].Text;    //选中行第 2 列，姓名
        txt_Company.Text = myitem.SubItems[2].Text; //选中行第 3 列，单位
        rbtn_Sex1.Checked = myitem.SubItems[3].Text == "男" ? true : false; //性别
        rbtn_Sex2.Checked = myitem.SubItems[3].Text == "女" ? true : false; //性别
        nudown_Age.Value = decimal.Parse(myitem.SubItems[4].Text);  //年龄
        txt_Telephone.Text = myitem.SubItems[5].Text; //电话
        txt_Address.Text = myitem.SubItems[6].Text;   //地址
        lbl_Status.Text = "修改";  //当前状态
    }
}
```

代码编写完成后，保存项目文件并运行程序。在程序界面的客户信息列表中选择各客户信息，则选中客户的各个信息分别显示在相应的文本框中。同时，当前状态变为“修改”，为修改客户信息的功能实现做出准备。

接着，继续实现客户信息修改功能。当用户根据具体情况修改客户相关信息后，将单击【保存】按钮实现修改功能。因此我们继续完善“保存”的单击事件函数，在“btn_Save_Click”函数代码段中相应位置编写如下代码。

```
//保存按钮
private void btn_Save_Click(object sender, EventArgs e)
{
    …
    else if (lbl_Status.Text == "添加")  //如果是“添加”状态
    {
        …
    }
    else   //修改操作
    {
        //1、创建数据库连接
        SqlConnection conn = new SqlConnection("Data Source=captainstudio;
database=db_case0705; Integrated Security=SSPI");

        //2、打开连接
        conn.Open();

        //3、数据修改语句
        string str = string.Format("update Customer_Info set
CustomerName='{0}',Company='{1}',Sex='{2}',Age={3},Telephone='{4}',Address='{5}' where
CustomerID={6}", name, company, sex, age, telephone, address, customerid);

        //4、创建执行对象
        SqlCommand cmd = new SqlCommand(str, conn);

        //5、执行操作，返回受影响的行数
        int i = cmd.ExecuteNonQuery();

        //6、关闭连接
        conn.Close();

        //7、处理结果
        if (i > 0)
        {
            lbl_Note.Text = "恭喜您，客户信息修改成功！";
            lbl_Note.ForeColor = Color.Blue;
            ClearTextBox();   //调用函数，清空各控件
            DataBind_Customer();  //重新加载客户信息
        }
        else
        {
            lbl_Note.Text = "对不起，客户信息修改失败！";
            lbl_Note.ForeColor = Color.Red;
        }
    }
}
```

代码编写完成后，保存项目文件并运行程序。在客户信息列表中选择要修改的客户，此时系统将选中的客户信息显示到各文本框中。接着根据修改相关客户信息，单击【保存】按钮，系统将到数据库中更新指定客户信息。最后，重新加载客户信息到 ListView 控件中。程序运行效果如图 14 所示。

(a) 选择要修改的客户信息 (b)客户信息修改成功

图 14 修改客户信息

步骤 6：删除客户信息

删除客户信息的功能的具体过程是：首先在客户信息列表中选择要删除的客户，然后单击【删除】按钮，实现删除操作。

双击【删除】按钮，为该按钮编写 Click 事件，系统自动打开 Form1.cs 代码界面，并且自动生成 btn_Delete_Click 事件函数。删除事件具体代码如下。

```
//【删除】按钮
private void btn_Del_Click(object sender, EventArgs e)
{
    if (customerid == "")   //如果没有选中要删除的客户信息
    {
        MessageBox.Show("请先选择要删除的客户信息");
    }
    else
    {
        //弹出删除确认提示框
        DialogResult result = MessageBox.Show("确定要删除选中的客户信息？", "删除提示",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.Yes) //如果确定删除
        {
            //1、创建数据库连接
            SqlConnection conn = new SqlConnection("Data Source=captainstudio;
database=db_case0705; Integrated Security=SSPI");

            //2、打开连接
            conn.Open();

            //3、数据修改语句
            string str = string.Format("delete from Customer_Info where
CustomerID={0}",customerid);

            //4、创建执行对象
            SqlCommand cmd = new SqlCommand(str, conn);
```

```
            //5、执行操作，返回受影响的行数
            int i = cmd.ExecuteNonQuery();

            //6、关闭连接
            conn.Close();

            //7、处理结果
            if (i > 0)
            {
                lbl_Note.Text = "恭喜您，客户信息删除成功！";
                lbl_Note.ForeColor = Color.Blue;
                ClearTextBox();   //调用函数，清空各控件
                DataBind_Customer();  //重新加载客户信息
            }
            else
            {
                lbl_Note.Text = "对不起，客户信息删除失败！";
                lbl_Note.ForeColor = Color.Red;
            }
        }
    }
}
```

接着继续完善 ClearTextBox()函数，添加代码“customerid = "";”，当删除操作完成后，在自动清空各控件信息的同时，清空全局变量 customerid 的值。具体代码如下：

```
//清空各控件
protected void ClearTextBox()
{
    ...
    lbl_Status.Text = "添加";
    customerid = "";
}
```

代码编写完成后，保存项目文件并运行程序。在程序界面客户信息列表中选择要删除的客户，然后单击【删除】按钮，系统弹出“删除确认”提示框，当用户选择“是”时，系统将选中客户信息从数据库中删除，并重新加载客户信息到 ListView 控件中。当用户选择“否”时，系统将不执行删除操作，程序运行效果如图 15 所示。

(a) 删除确认提示　　(b)删除成功提示

图 15 删除客户信息

步骤 7：【取消】按钮事件

【取消】按钮的作用就是清空各控件的值，还原各变量的状态。【取消】按钮主要用在以下情况。

1）添加操作：当用户填写客户的部分信息后，又不想继续添加，可以单击【取消】按钮。

2）修改操作：当用户选中某客户后，又不想修改该客户信息，想要重新添加其他客户，可以单击【取消】按钮。

3）删除操作：当用户选中某客户后，又不想删除该客户信息，想要重新添加其他客户，可以单击【取消】按钮。

双击【取消】按钮，为该按钮编写 Click 事件，系统自动打开 Form1.cs 代码界面，并且自动生成 btn_Cancel_Click 事件函数。具体代码如下：

```
//【取消】按钮
private void btn_Cancel_Click(object sender, EventArgs e)
{
    ClearTextBox();  //调用函数，清空各控件
    lbl_Note.Text = "";
}
```

至此，使用 ListView 实现客户信息管理的功能开发完成。