2020/07/02_MFC_第4课_常用控件的使用

笔记本:	MFC
创建时间:	2020/7/2 星期四 15:44
作者:	ileemi
标签:	MFC常用控件的使用

常用控件的使用

文本编辑框控件的回顾

Static Test -- 静态文本框（标签），说明性作用

获取文本编辑框的内容，可以使用GetDlgItem获取CWnd* 的对象指针，也可以使用DDX进行绑定，绑定到某个变量类型（Edit绑定到CString上面），来获取文本编辑框内的文本信息。

MFC中类型不建议使用中文，缺点按钮控件双击不会快速生成代码。

使用 GetDlgItem 会返回一个CWnd* 类型的类指针如果想使用CEdit* 访问，只需要在GetDlgItem前强转对应的类型即可。
示例：
CEdit* pEdit = (CEdit*)GetDlgItem(EDIT_STU_NAME);

CEdit 提供的成员可以通过 MSDN 进行查看，但是其提供的成员函数并不多。
获取文本编辑框内的文本数据，可以使用下面的两种常见的方式：

```cpp
void CStudentInfoDlg::OnBnClickedGetStuinfo()
{
    // TODO: 在此添加控件通知处理程序代码
    /*
    使用 GetDlgItem 会返回一个CWnd* 类型的类指针如果想使用CEdit* 访问，
    是需要在GetDlgItem前强转对应的类型即可
    */
    // 方法一
    // 通过对应控件的类型访问控件 Edit --> CEdit，Button --> CButton
    CEdit* pEdit = (CEdit* )GetDlgItem(EDIT_STU_NAME);
    CString strStuName;
    pEdit->GetWindowText(strStuName);
    AfxMessageBox(strStuName);

    // 方法二，CEdit 一般使用都是对字符串的操作，直接将其绑定到CString上使用
    // 通过绑定变量也可以访问文本编辑框内的文本
    CString strStuName;
    m_stuName.GetWindowText(strStuName);
    AfxMessageBox(strStuName);
}
```

单选框

Radio -- 单选框 多个单选框之间选中一个，选中的生效
male -- 男
female -- 女
other -- 其它
unknown -- 未知的

Radio Button（单选框） 和 Cheak Box（多选框） 以及 Button 同属于一个类 --> **CButton**。

使用的时候，需要判断当前的按钮是否被选中，CButton 内部封装了几个功能函数，其中 GetStart(不好用)，可以使用 **GetCheck** -- 检索单选按钮或复选框的复选状态。

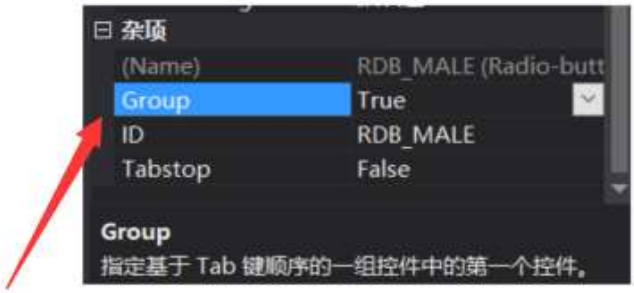
1. 方法一：使用GetCheck函数，这样的写法比较麻烦，代码冗余。

```cpp
// 获取学生的性别
CButton* pRdbMale = (CButton*)GetDlgItem(RDB_MALE);
CButton* pRdbFeMale = (CButton*)GetDlgItem(RDB_FEMALE);
CButton* pRdbOther = (CButton*)GetDlgItem(RDB_OTHER);
CButton* pRdbUnknown = (CButton* )GetDlgItem(RDB_UNKNOWN);

if (pRdbMale->GetCheck() == 1)
{
    AfxMessageBox("当前学生性别：男");
}
if (pRdbFeMale->GetCheck() == 1)
{
    AfxMessageBox("当前学生性别：女");
}
if (pRdbOther->GetCheck() == 1)
{
    AfxMessageBox("当前学生性别：其它");
}
```

MFC 同样支持使用 DDX 将一组单选框绑定到一个 int 类型的变量上。

2. 方法二：单选框需要进行分组，分组的方式就是给第一个单选框设置一个 Group 属性，后面没有设置 Group 属性的单选框和前面设置Group属性的单选框并为一组，知道后面遇到下一个具有 Group 属性的单选框开始，另一组开始。设置 Group 属性后，一组内的单选框只有一个可以被选中。
   Group的设置在单选框属性里：

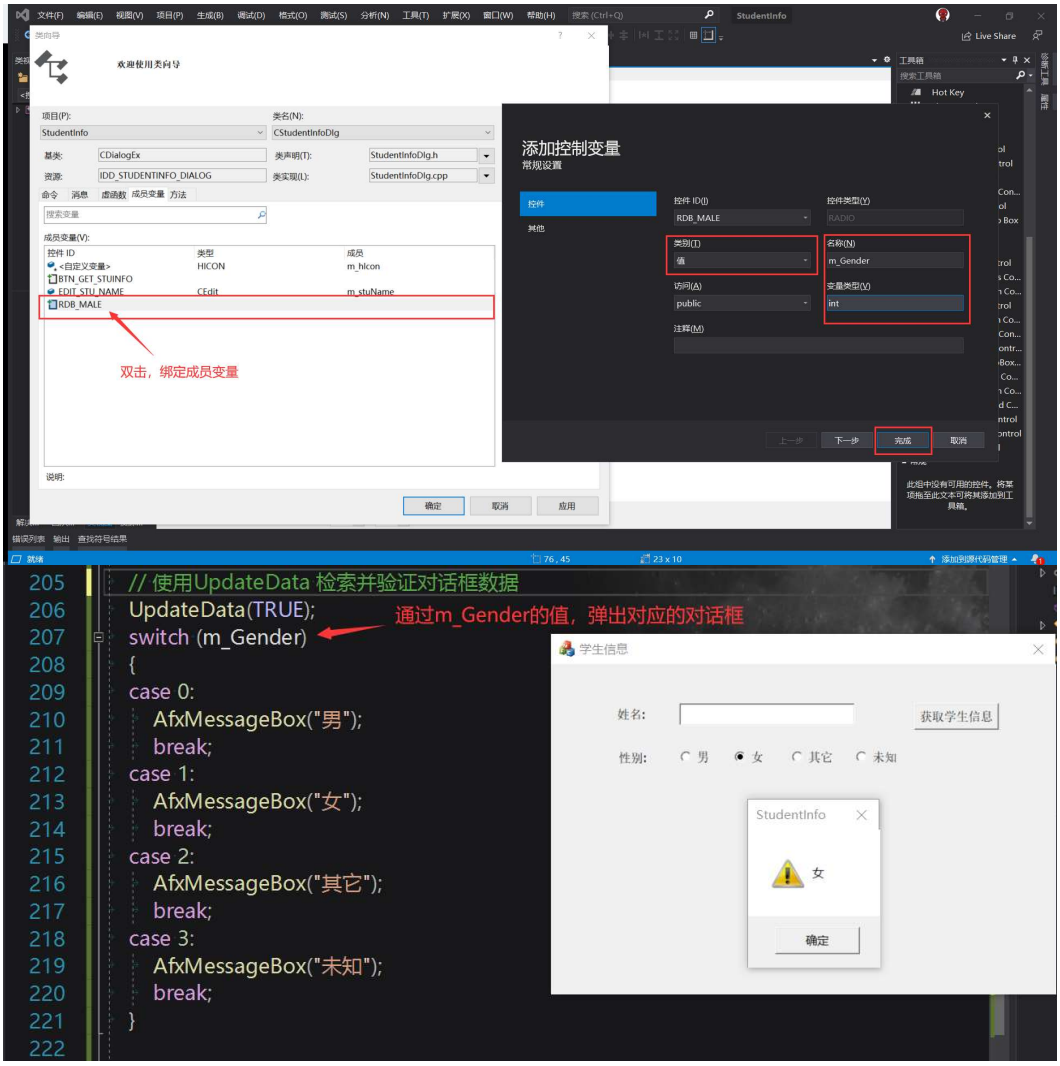


单选框的属性

注意：

为单选框绑定成员变量，只有设置 Group 属性的单选框才能绑定成员变量，一组单选框只需要绑定一次即可。

示例：

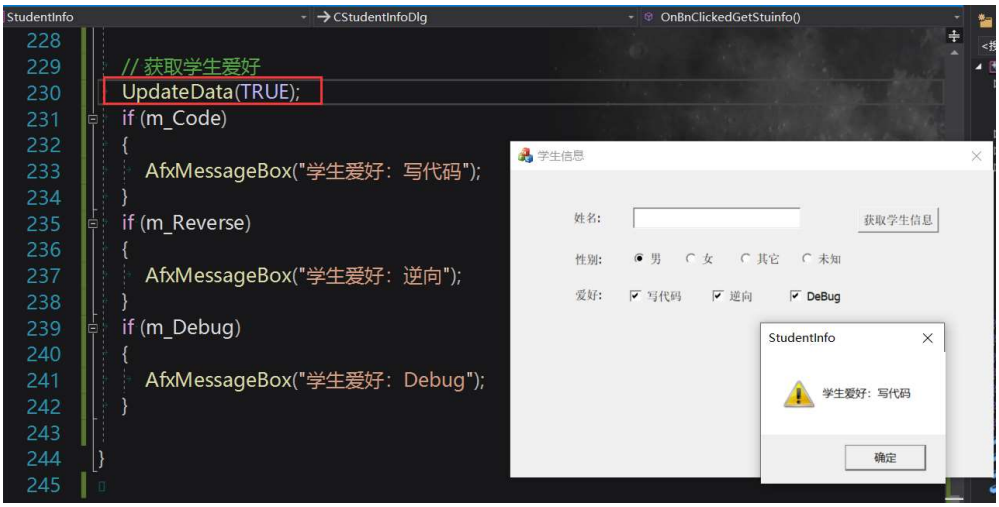


复选框（Check Box ）

支持多选。

使用方法：

- 方法一：可以使用 GetDlgItem 返回一个指向复选框的类对象指针（CButton* ），然后通过 GetCheck() 获取复选框的状态，这种方法比较麻烦。
- 方法二：使用 DDX，对于每个复选框都有选中和不选中两种状态，可以为每个复选框添加控制变量。



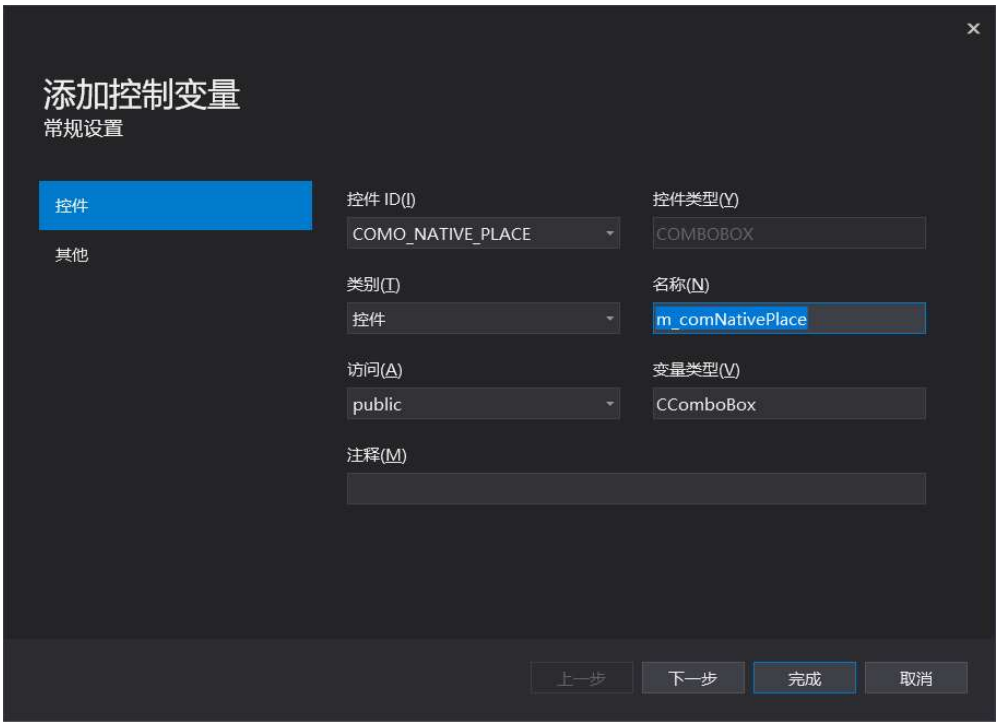
Combo Box -- 下拉列表

常用于籍贯的地址选择。本质上和 Edit + List Box 差不多。

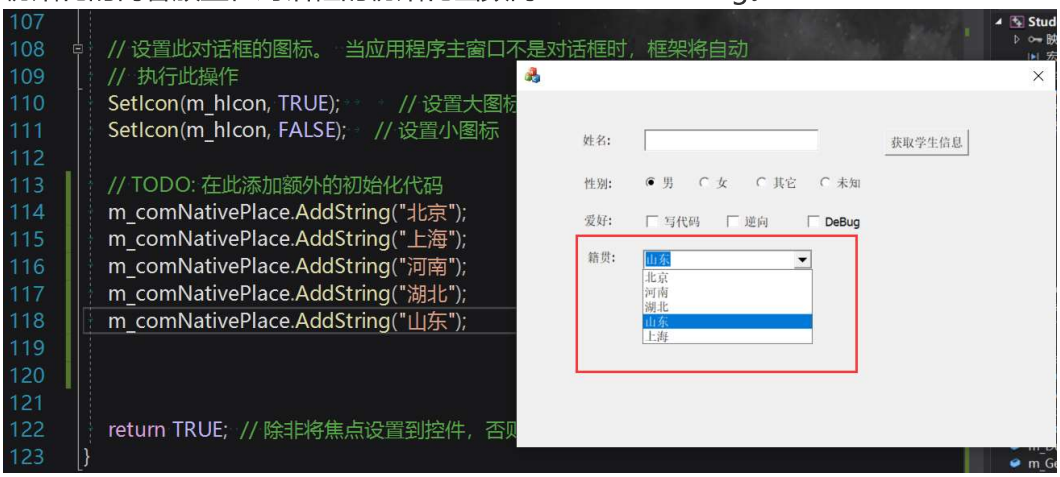
其具有三个属性：

- Simple -- 可以输入，内容不会折叠。
- DropDown -- 可以修改下拉列表中的数据。
- 下拉列表 -- 对于籍贯，这个属性常用，不可以修改下拉列表中的数据。
  对这个控件有很多操作的话，不适合绑定到基础类型上，适合绑定到封装的对应

类型上，CComboBox类上。



初始化的内容放置在对话框的初始化函数内 -- OnInitDialog。



初始化下拉列表的内容
类.AddString("xxx")
为下拉列表添加默认值
类.SetCurSel(索引)

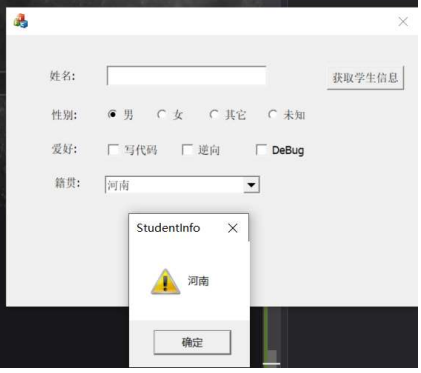
获取下拉列表中选中的数据，需要使用API函数 -- GetLBText，使用 GetLBText 还需要使用 GetCurSel 获取鼠标点击选中的数值索引值。

GetLBText 函数定义：
void GetLBText(int nIndex, CString& rString) const;

使用示例：

```
// 获取学生的籍贯
int nCurSel = m_comNativePlace.GetCurSel();
CString strNativePlace;
m_comNativePlace.GetLBText(nCurSel, strNativePlace);
AfxMessageBox(strNativePlace);

}
```



SetItemData GetItemData 的使用

列表框 或者 稍微复杂一点的控件都提供这个功能：

Combo Box、List Box、List Control、Tree Control等
SetItemData -- 允许为下拉列表项添加数据时同时添加一个附加数据（一个数值等）
GetItemData -- 通过索引获取对应索引上存储的数据

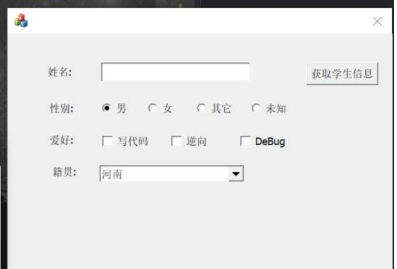
使用示例：

```
int nIdx = 0;
nIdx = m_comNativePlace.AddString("北京");
m_comNativePlace.SetItemData(nIdx, 0x00001);
nIdx = m_comNativePlace.AddString("上海");
m_comNativePlace.SetItemData(nIdx, 0x00002);
nIdx = m_comNativePlace.AddString("河南");
m_comNativePlace.SetItemData(nIdx, 0x00003);
nIdx = m_comNativePlace.AddString("湖北");
m_comNativePlace.SetItemData(nIdx, 0x00004);
nIdx = m_comNativePlace.AddString("山东");
m_comNativePlace.SetItemData(nIdx, 0x00005);
nIdx = m_comNativePlace.SetCurSel(0);
```

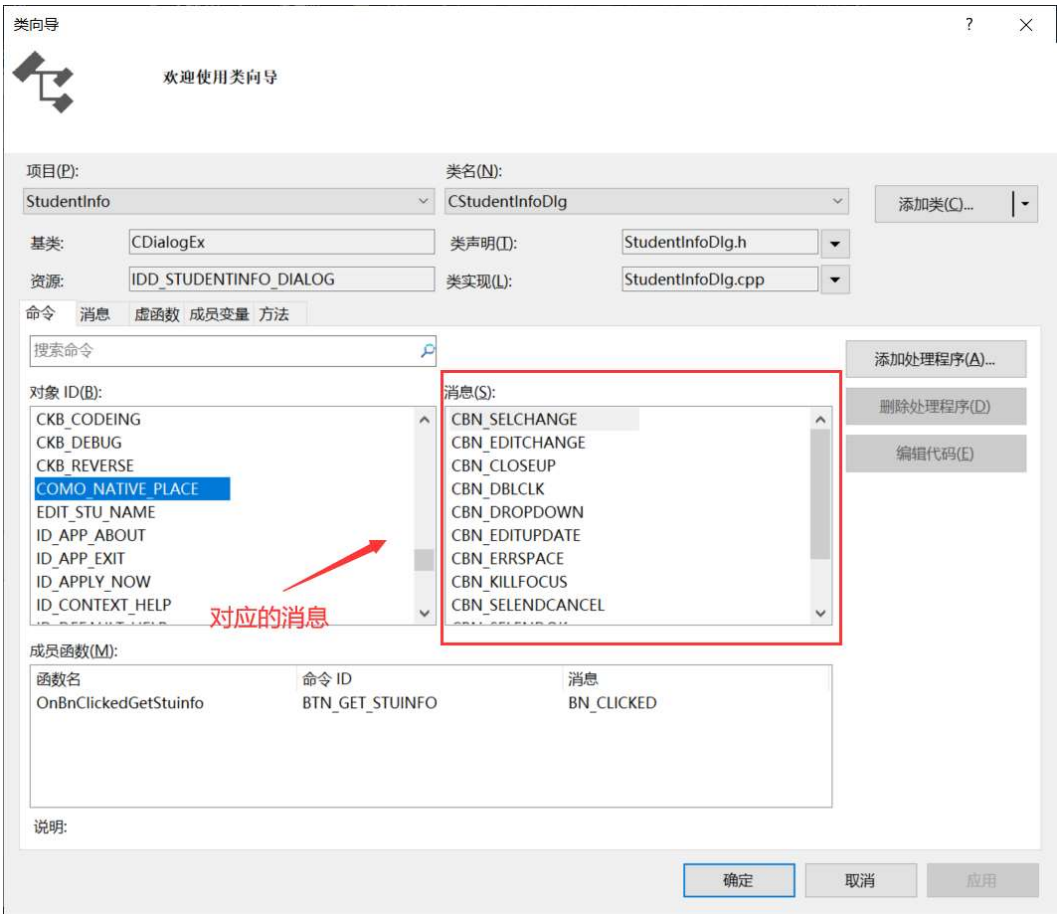
```
// 获取学生的籍贯
int nCurSel = m_comNativePlace.GetCurSel();     // 获取当前鼠标选中的索引
CString strNativePlace;
m_comNativePlace.GetLBText(nCurSel, strNativePlace);
DWORD dwItemData = m_comNativePlace.GetItemData(nCurSel);
CString strFmt;
strFmt.Format("%s %04x", strNativePlace, dwItemData);
AfxMessageBox(strFmt);

}
```

需要格式化字符串



Combo Box 相关的消息：



List Control

类似于文件夹的文件信息栏，如下图所示：

对于 List Control 的基本操作就是列表内容的增删改查等。

注意事项：

- 增加的时候需要分两部分，先初始化表头，在增加显示表中的内容
- 需要为 List Control 添加单独类变量，变量的类型为 **CListCtrl**

CListCtrl 插入新列 API -- InsertColumn

CListCtrl::InsertColumn

int InsertColumn(int *nCol*, **const LVCOLUMN*** *pColumn* **);**

int InsertColumn(int *nCol*, **LPCTSTR** *lpszColumnHeading*, **int** *nFormat* **= LVCFMT_LEFT, int** *nWidth* **= -1, int** *nSubItem* **= -1);**

提供了重载

Return Value

The index of the new column if successful or -1 otherwise.

Parameters

nCol
The index of the new column.

pColumn
Address of an **LVCOLUMN** structure that contains the attributes of the new column.

lpszColumnHeading
Address of a string containing the column's heading.

nFormat
Integer specifying the alignment of the column. It can be one of these values: **LVCFMT_LEFT**, **LVCFMT_RIGHT**, or **LVCFMT_CENTER**.

nWidth
Width of the column, in pixels. If this parameter is -1, the column width is not set.

nSubItem
Index of the subitem associated with the column. If this parameter is -1, no subitem is associatied with the column.

添加表头数据使用 **SetColumnWidth** 设置表头数据的间隔宽度。

CListCtrl::SetColumnWidth

BOOL SetColumnWidth(int *nCol*, **int** *cx* **);**

Return Value

Nonzero if successful; otherwise zero.

根据内容自动调整列宽

Parameters

nCol
Index of the column whose width is to be set. In list view, this parameter must be -1.

cx
The new width of the column. Can be either **LVSCW_AUTOSIZE** or **LVSCW_AUTOSIZE_USEHEADER**, as described in LVM_SETCOLUMNWIDTH in the *Platform SDK*.

根据表头自动调整列宽

Remarks

Changes the width of a column in report view or list view.

增加学生数据

步骤一：想要获取头部列表项数的个数，由于表头被封装成了一个类，需要拿到类对象的指针，然后通过类对象指针使用API函数 **GetItemCount** 来获取对应的表头个数。

InsertColumn -- 在列表视图控件中插入新列

GetHeaderCtrl -- 检索列表视图控件的标题控件

定义：CHeaderCtrl* GetHeaderCtrl( );

GetItemCount -- 检索标题控件中项的个数。

CHeaderCtrl Class Members

Construction

CHeaderCtrl	Constructs a **CHeaderCtrl** object.
Create	Creates a header control and attaches it to a **CHeaderCtrl** object.

Attributes

GetItemCount	Retrieves a count of the items in a header control.

```
void CStudentInfoDlg::OnBnClickedBtnAddinfo()
{
    // TODO: 在此添加控件通知处理程序代码
    // 插入表头数据
    int nListIdx = 0;  // 列的索引
    m_listcStuInfo.InsertColumn(nListIdx++, "学号");
    m_listcStuInfo.InsertColumn(nListIdx++, "姓名");
    m_listcStuInfo.InsertColumn(nListIdx++, "性别");
    m_listcStuInfo.InsertColumn(nListIdx++, "爱好");
    m_listcStuInfo.InsertColumn(nListIdx++, "籍贯");

    // 自动调整列表头部列宽
    CHeaderCtrl* pHdrCtrl = m_listcStuInfo.GetHeaderCtrl();
    for (int i = 0; i < pHdrCtrl->GetItemCount(); i++)
    {
        m_listcStuInfo.SetColumnWidth(i, LVSCW_AUTOSIZE_USEHEADER);
    }
}
```

步骤二：添加学生数据，一行一行的插入，设置文本
使用到的API函数：

- **InsertItem** -- 在列表视图控件中插入新项，在一行的开头插入数据
- **SetItemText** -- 更改列表视图项或子项的文本 -- 在当前行后面的列中插入数据
- **CListCtrl::SetExtendedStyle** -- 设置风格，选中一行（**LVS_EX_FULLROWSELECT**），需要注意的就是，为控件添加风格的同时还需要保持原有的风格。
- **SetColumnWidth** -- 设置列宽

```
// 设置风格，鼠标滑动选中一行
m_listcStuInfo.SetExtendedStyle(m_listcStuInfo.GetExtendedStyle() |
    LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);

// 插入表头数据
int nListIdx = 0;  // 列的索引
m_listcStuInfo.InsertColumn(nListIdx++, "学号");
m_listcStuInfo.InsertColumn(nListIdx++, "姓名");
m_listcStuInfo.InsertColumn(nListIdx++, "性别");
m_listcStuInfo.InsertColumn(nListIdx++, "爱好");
m_listcStuInfo.InsertColumn(nListIdx++, "籍贯");

// 添加学生数据，一行一行数据进行添加
int nItemIdx = 0;
int nColIdx = 0;
m_listcStuInfo.InsertItem(nItemIdx, "001");
m_listcStuInfo.SetItemText(nItemIdx, ++nColIdx, "李四");
m_listcStuInfo.SetItemText(nItemIdx, ++nColIdx, "男");
m_listcStuInfo.SetItemText(nItemIdx, ++nColIdx, "Coding");
m_listcStuInfo.SetItemText(nItemIdx++, ++nColIdx, "北京");
```

```
//设置风格，鼠标滑动选中一行
m_listcStuInfo.SetExtendedStyle(m_listcStuInfo.GetExtendedStyle() |
    LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES);
```

获取选中的学生数据

API -- **GetItemText**
函数原型：

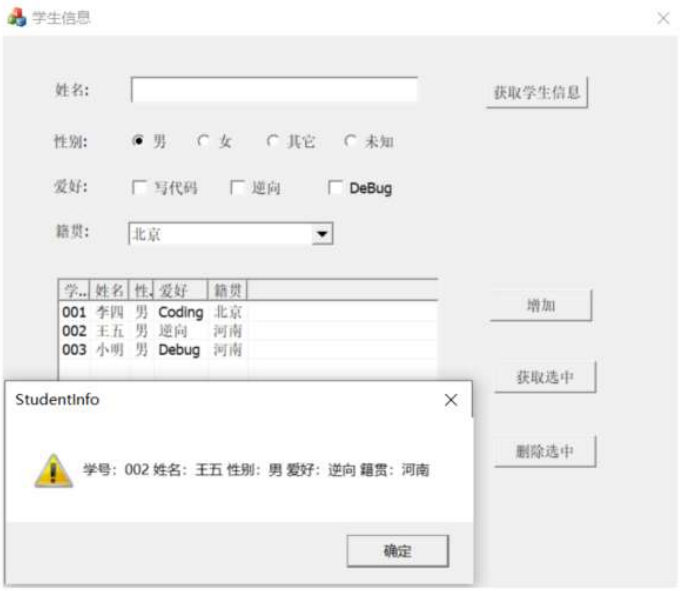
CString GetItemText(int nItem, int nSubItem) const;

CListCtrl::**GetSelectionMark** -- 检索列表视图控件的选择标记，说白就是获取鼠标选中的的项在 List Box 中的索引。

代码示例：

```
//获取鼠标选中的学生信息
void CStudentInfoDlg::OnBnClickedBtnGetinfo()
{
//TODO: 在此添加控件通知处理程序代码
//获取鼠标选中的索引
int nIdx = m_listcStuInfo.GetSelectionMark();

//获取索引对应的学生数据
int nSubItemText = 0;
CString strStuID = m_listcStuInfo.GetItemText(nIdx, nSubItemText++);
CString strStuName = m_listcStuInfo.GetItemText(nIdx, nSubItemText++);
CString strStuGender = m_listcStuInfo.GetItemText(nIdx, nSubItemText++);
CString strStuHobby= m_listcStuInfo.GetItemText(nIdx, nSubItemText++);
CString strStuNativePlace = m_listcStuInfo.GetItemText(nIdx, nSubItemText++);
CString strFmt;
strFmt.Format("学号：%s 姓名：%s 性别：%s 爱好：%s 籍贯：%s",
 strStuID, strStuName, strStuGender, strStuHobby, strStuNativePlace);
AfxMessageBox(strFmt);
}
```



删除选中的学生信息

DeleteItem -- 从控件中删除项
函数原型：BOOL DeleteItem(int nItem);

将鼠标选中的行数据进行删除。

```
359    //·删除鼠标选中的学生信息
360    void·CStudentInfoDlg::OnBnClickedBtnDelinfo()
361    {
362    //·TODO:·在此添加控件通知处理程序代码
363    //·获取鼠标编辑的项索引值
364    int·nIdx·=·m_listcStuInfo.GetSelectionMark();
365    m_listcStuInfo.DeleteItem(nIdx);
366    }
367
```

修改选中的学生信息

选中要修改的学生信息，然后可以通过 SetItemText 对学生信息进行相关的修改。