

# EXPERIMENT 实验 5

## 框架窗口、文档和视图

在 MFC 中,框架窗口(CMainFrame)、文档(CDocument)和视图(CView)是构成文档应用程序的核心。框架窗口是文档和视图的容器。文档代表一个数据单元,用户可使用“文件”菜单中的“打开”和“保存”命令进行文档数据操作。视图是框架窗口的子窗口,它与文档紧密相联,是用户与文档之间的交互接口。同时,它们也是 MFC 极为重要的一种结构体系,这种结构体系使得程序中的数据与它的显示形式和用户交互分离开来,能较容易地满足单文档和多文档的框架结构的需要。

本实验(实训)中,Ex\_Form 是用表单视图来进行学生的课程成绩管理的一个简单例程,但这次要用到 Visual Studio 2010 中 MFC 强大的 PropertyGrid(属性网格)控件。同时,为了能用列表视图来显示学生的课程成绩,还添加了“视图切换”功能。Ex\_Look 是一个自制的简单资源浏览器的例程,其中还用到了切分窗口、文档和文件夹的本地查找以及 CImageList 类的使用等。

### 实验目的

- 了解使类可序列化的方法。
- 学会使用文档类机制存取数据。
- 熟悉不同视图类的创建和使用方法。
- 了解一档多视切换的实现方法。
- 熟悉文档视图结构,学会切分窗口的使用方法。

### 实验内容

- 表单 Ex\_Form。
- 视图切换。
- 切分窗口。

### 实验准备和说明

- 具备知识:框架窗口、文档和视图(教程第 5 章)。
- 构思并准备上机所需要的程序 Ex\_Form、Ex\_Look。
- 创建本实验(实训)的工作文件夹“D:\Visual C++ 程序\LiMing\5”。

## 5.1 表单 Ex\_Form

表单视图实质上是将对话框(资源)模板机制应用在视图中,这样就可通过表单视图应用程序 Ex\_Form 对课程成绩进行管理,如图 5.1 所示。单击“添加”按钮,则从后台存储空间中添加数据,同时显示在表单的列表框中。单击“删除”按钮,则从后台存储空间中删除数据,同时在表单中的列表框选项也被删除。单击“刷新”按钮,则将后台存储空间中的数据重新显示在表单的列表框中。当关闭程序后,若后台存储空间中的数据被更改,则还提示一个消息对话框用来是否将数据存储到文档中。若选择“文件”→“打开”菜单项,则还能从外部文件中调用课程成绩数据。






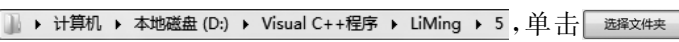

图 5.1 Ex\_Form 运行结果

具体的实验(实训)过程如下。

- (1) 设计表单。
- (2) 可序列化类。
- (3) MFC 属性网格。
- (4) 实现数据操作。

### 5.1.1 设计表单

具体步骤如下。

- (1) 启动 Microsoft Visual Studio 2010。
- (2) 选择“文件”→“新建”→“项目”菜单命令或按快捷键 Ctrl+Shift+N 或单击顶层菜单下的标准工具栏中的  按钮,弹出“新建项目”对话框。在“已安装的模板”栏下选中“Visual C++”下的“MFC”结点,在中间的模板栏中选中  MFC 应用程序。
- (3) 单击“位置”编辑框右侧的“浏览”按钮 ,从弹出的“项目位置”对话框中指定项目所在的文件夹 ,单击  按钮,回到“新建项目”对话框中。

(4) 在“新建项目”对话框的“名称”编辑框中输入名称“Ex\_Form”。同时,要取消勾选“为解决方案创建目录”复选框。



(5) 单击 **确定** 按钮,出现“MFC 应用程序向导”欢迎页面,单击 **下一步 >** 按钮,出现“应用程序类型”页面。选中“单个文档”应用程序类型,取消勾选“使用 Unicode 库”复选框,选中右侧的“项目类型”的“MFC 标准”,取消勾选“启用视觉样式切换”复选框,如图 5.2 所示。单击左侧“用户界面功能”,取消勾选“用户定义的工具栏和图像”及“个性化菜单行为”复选框,如图 5.3 所示。



图 5.2 应用程序类型选择



图 5.3 用户界面功能选择

(6) 单击左侧“生成的类”,将 CEx\_FormView 的基类选为 CFormView。保留其他默认选项,单击  按钮,提示“没有可用于 CFormView 的打印支持。是否继续?”,单击“是”按钮,系统开始创建,并又回到了 Visual C++ 主界面。将项目工作区切窗口换到“解决方案管理器”页面,双击头文件结点  stdafx.h,打开 stdafx.h 文档,滚动到最后代码行,将“#ifdef \_UNICODE”和最后一行的“#endif”删除(注释掉)。



(7) 将项目工作区窗口切换到“资源视图”页面,展开结点,双击 Dialog 下的 IDD\_EX\_FORM\_FORM,打开表单资源模板。单击对话框编辑器上的“网格切换”按钮 ,显示模板网格。删除原来的静态文本控件,调整表单模板的大小(调为 417×177px),参看图 5.1 控件布局添加如表 5.1 所示的一些控件(其中添加的 MFC 属性网格控件的 Description Rows Count 属性设为 2,其他默认)。


表 5.1 表单添加的控件

添加的控件	ID	标题	其他属性
列表框	IDC_LIST1	—	默认
MFC 属性网格	IDC_MFCPROPERTYGRID1	—	见前
按钮	IDC_BUTTON_REFRESH	刷新	默认
按钮	IDC_BUTTON_ADD	添加	默认
按钮	IDC_BUTTON_DEL	删除	默认

(8) 右击列表框 IDC\_LIST1 控件,从弹出的快捷菜单中选择“添加变量”命令,弹出“添加成员变量向导”对话框,在“变量名”中输入“m\_ListBox”,单击  按钮。类似地,为 MFC 属性网格 IDC\_MFCPROPERTYGRID1 添加控件变量 m\_PropGrid。

## 5.1.2 可序列化类

这里为学生课程成绩建立一个可序列化的 CScore 类,具体步骤如下。

(1) 选择“项目”→“类向导”菜单或按快捷键 Ctrl+Shift+X,弹出“MFC 类向导”对话框。单击右侧“添加类”按钮的下拉按钮,从弹出的下拉选项中选择“MFC 类”,弹出“MFC 添加类向导”对话框,将“基类”选为 CObject,输入“类名”为 CScore,如图 5.4 所示,单击  按钮。关闭“MFC 类向导”对话框。

(2) 在打开的 Score.h 文件中添加下列 CScore 类声明代码。

```
class CScore : public CObject
{
public:
    CScore();
    virtual ~CScore();
public:
    CScore(CString strcid, CString strcname, CString strclass,
        CString strstuid, CString strstuname, float fscore);
```



图 5.4 添加 CScore 类

```

public:
    void SetData(CString strcid, CString strcname, CString strclass,
        CString strstuid, CString strstuname, float fscore);
    void GetDataString(CString &strData);
    CString    GetCourseID(void)
    {
        return strCID;
    }
    CString    GetCourseName(void)
    {
        return strCName;
    }
    CString    GetClassName(void)
    {
        return strClass;
    }
    CString    GetStudentID(void)
    {
        return strStuID;
    }
    CString    GetStudentName(void)
    {

```

```

        return strStuName;
    }
    float    GetScore(void)
    {
        return fScore;
    }
    void Serialize(CArchive &ar);
private:
    CString    strCID;           //课程号
    CString    strCName;        //课程名
    CString    strClass;        //班级
    CString    strStuID;        //学号
    CString    strStuName;      //学生姓名
    float      fScore;          //成绩
    DECLARE_SERIAL(CScore)      //序列化声明
};

```

(3) 打开 Score.cpp 文件,在最后添加下列 CScore 类实现代码。

```

//CScore 成员函数
IMPLEMENT_SERIAL(CScore, CObject, 1)    //序列化实现
void CScore::Serialize(CArchive &ar)
{
    if(ar.IsStoring())
        ar<<strCID<<strCName<<strClass<<strStuID<<strStuName<<fScore;
    else
        ar>>strCID>>strCName>>strClass>>strStuID>>strStuName>>fScore;
}
CScore::CScore(CString strcid, CString strcname, CString strclass,
               CString strstuid, CString strstuname, float fscore)
{
    strCID      = strcid;
    strCName    = strcname;
    strClass    = strclass;
    strStuID    = strstuid;
    strStuName  = strstuname;
    fScore      = fscore;
}
void CScore::SetData(CString strcid, CString strcname, CString strclass,
                    CString strstuid, CString strstuname, float fscore)
{
    strCID      = strcid;
    strCName    = strcname;
    strClass    = strclass;
}

```

```

        strStuID      = strtstuid;
        strStuName    = strtstuname;
        fScore       = fscore;
    }
void CScore::GetDataString(CString &strData)
{
    strData.Format("%10s%12s%10s%10s%10s%7.1f",
        strCID, strCName, strClass, strStuID, strStuName, fScore);
}

```

(4) 编译。

### 5.1.3 MFC 属性网格

MFC 属性网格(PropertyGrid)是 Visual Studio 2010 中新增的 MFC 控件,顾名思义,其主要是用来显示、设置和获取某一(或多个)对象的属性值。在 MFC 中,属性网格由 CMFCPropertyGridCtrl 类封装,而每个属性则由 CMFCPropertyGridProperty 类来控制,属性也可分组(类别)。这里将学生课程成绩信息分为两组:学生信息和课程信息。学生信息包含姓名、班级和学号,课程信息包含课程名、课程号和课程成绩。

具体步骤如下。

(1) 为了简单编程,在 CEx\_FormView 类中添加 6 个属性成员指针变量,分别表示姓名、班级、学号、课程名、课程号和课程成绩。

```

//操作
public:
    CMFCPropertyGridProperty *   m_propStuName;        //姓名
    CMFCPropertyGridProperty *   m_propStuClass;       //班级
    CMFCPropertyGridProperty *   m_propStuID;         //学号
    CMFCPropertyGridProperty *   m_propCName;         //课程名
    CMFCPropertyGridProperty *   m_propCID;          //课程号
    CMFCPropertyGridProperty *   m_propScore;        //课程成绩

```

(2) 在 CEx\_FormView::OnInitialUpdate() 函数中添加下列初始化代码。

```

void CEx_FormView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    GetParentFrame()->RecalcLayout();
    ResizeParentToFit();
    if(m_PropGrid.GetPropertyCount()>0)
        return;
    //属性网格
    m_propStuName    = new CMFCPropertyGridProperty("姓名",
        "LiMing", "学生姓名全称!");
}

```

```
m_propStuClass      = new CMFCPropertyGridProperty("班级",
                                                    "202105", "6位!");
m_propStuID         = new CMFCPropertyGridProperty("学号",
                                                    "20210501", "8位,前6位是班号!");
m_propCName         = new CMFCPropertyGridProperty("课程名",
                                                    "Visual C++", "按课程计划而定!");
m_propCID           = new CMFCPropertyGridProperty("课程号",
                                                    "2110911", "7位!");
m_propScore         = new CMFCPropertyGridProperty("成绩",
                                                    "90", "60分及格,最高100分!");

//分组
CMFCPropertyGridProperty * pGroup1, * pGroup2;
pGroup1 = new CMFCPropertyGridProperty("学生信息");
pGroup1->AddSubItem(m_propStuName);
pGroup1->AddSubItem(m_propStuClass);
pGroup1->AddSubItem(m_propStuID);
pGroup2 = new CMFCPropertyGridProperty("课程信息");
pGroup2->AddSubItem(m_propCName);
pGroup2->AddSubItem(m_propCID);
pGroup2->AddSubItem(m_propScore);
//添加到属性网格
m_PropGrid.AddProperty(pGroup1);
m_PropGrid.AddProperty(pGroup2);
//展开所有属性
m_PropGrid.ExpandAll();
//设置表头及首列宽度
HDITEM item;
item.cxy          = 120;
item.mask         = HDI_WIDTH;
m_PropGrid.GetHeaderCtrl().SetItem(0, new HDITEM(item));
m_PropGrid.EnableHeaderCtrl(TRUE, "信息属性", "值");
//设置颜色
COLORREF c        = (COLORREF)-1; //默认颜色
m_PropGrid.SetCustomColors(afxGlobalData.clrBtnLight, c,
                           afxGlobalData.clrBtnShadow, afxGlobalData.clrBtnHilite,
                           RGB(188, 199, 216), c, afxGlobalData.clrBtnDkShadow);
m_PropGrid.RedrawWindow();
}
```

(3) 编译并运行。

#### 5.1.4 实现数据操作

具体步骤如下。

(1) 为 CEx\_FormDoc 类添加下列成员变量,用来保存添加的 CScore 类对象数据。



```
//操作
public:
    CObArray          m_cobArray;          //对象集合类对象
```

(2) 在 CEx\_FormDoc 类析构函数 ~CEx\_FormDoc 中添加下列删除并释放对象的代码。

```
CEx_FormDoc::~CEx_FormDoc()
{
    int nIndex = m_cobArray.GetSize();
    while(nIndex-->0)
        delete m_cobArray.GetAt(nIndex);    //删除并释放对象的内存空间
    m_cobArray.RemoveAll();
}
```

(3) 在 CEx\_FormDoc::Serialize() 函数中添加下列代码。

```
void CEx_FormDoc::Serialize(CArchive& ar)
{
    if(ar.IsStoring())
    {}else
    {}
    m_cobArray.Serialize(ar);
}
```

(4) 为 CEx\_FormView 表单中的按钮 IDC\_BUTTON\_ADD、IDC\_BUTTON\_DEL 和 IDC\_BUTTON\_REFRESH 添加 BN\_CLICKED“事件”的消息映射,保留默认的映射处理函数名,并添加下列代码。

```
void CEx_FormView::OnBnClickedButtonAdd()
{
    CString strCID = m_propCID->GetValue();
    strCID.Trim();
    if(strCID.IsEmpty())
    {
        MessageBox("[课程号]不能为空!");    return;
    }

    CString strCName = m_propCName->GetValue();
    strCName.Trim();
    if(strCName.IsEmpty())
    {
        MessageBox("[课程名]不能为空!");    return;
    }
}
```

```
CString strClass = m_propStuClass->GetValue();
strClass.Trim();
if(strClass.IsEmpty())
{
    MessageBox("[班级]不能为空!"); return;
}

CString strStuName = m_propStuName->GetValue();
strStuName.Trim();
if(strStuName.IsEmpty())
{
    MessageBox("[姓名]不能为空!"); return;
}

CString strStuID = m_propStuID->GetValue();
strStuID.Trim();
if(strStuID.IsEmpty())
{
    MessageBox("[学号]不能为空!"); return;
}

CEx_FormDoc * pDoc = GetDocument();
//查找是否有相同的课程号与学号相同的记录数据
BOOL bFind = FALSE;
CScore * pTemp;
CString strTemp;
int nIndex = 0;
for(nIndex=0; nIndex<pDoc->m_cobArray.GetSize(); nIndex++)
{
    pTemp = (CScore *) (pDoc->m_cobArray.GetAt(nIndex));
    if((pTemp->GetCourseID().Find(strCID)>=0) &&
        (pTemp->GetStudentID().Find(strStuID)>=0))
    {
        bFind = TRUE; break;
    }
}
if(bFind)
{
    strTemp.Format("当前第[%d]项有重复!", nIndex);
    MessageBox(strTemp); return;
} else
{
    CString strScore = m_propScore->GetValue();
```