# 利用MFC 规则DLL实现软件的中英界面切换

## 一、效果图

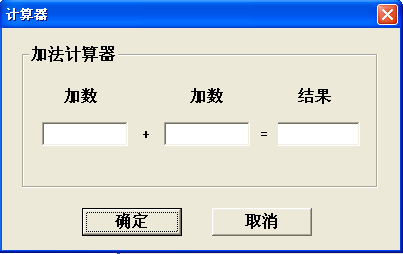
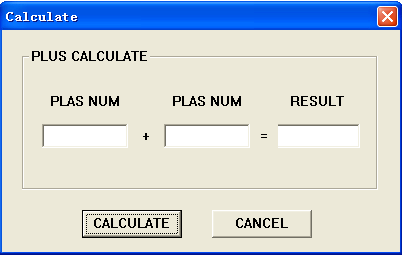
  

图1

## 二、实现原理

新建MFC规则DLL文件，为其添加两个样式相同，ID相同的两个对话框，界面上的文字分别为中文和英文。编译生成DLL文件。而在客户端程序中载入DLL中的资源，创建对话框类。这样，对话框类的资源在DLL中，而实现代码在客户端中，根据用户的选择来加载不同的资源，从而创建出不同语言的界面。

## 三、相关函数说明

### 3.1 LoadLibraryEx

将DLL作为数据文件映射到调用进程的地址空间。

### 3.2 FindResourceEx

### 3.3 LoadResource

将函数装载指定资源到全局存储器。

函数原型：HGLOBAL LoadResource(HMODULE hModule，HRSRC hResInfo)

参数：

hModule 处理包含资源的可执行文件的模块句柄，若为NULL，系统从当前过程中的模块中装载资源。

hReslnfo 将被装载的资源句柄，它必须由函数FindResource或FindResourceEx创建。

返回：如果运行成功，返回相应的资源数据句柄。若失败，返回NULL。

注意：为得到更多资源数据信息，请调用LockResource函数，当使用完资源后，必须通过调用函数释放资源。

### 3.4 LockResource

锁定内存中的指定资源，返回资源在内存中的地址，不须解锁操作。当过程创建终止时，系统将自动删除这些资源。

### 3.5 CreateDlgIndirect

使用内存中的对话框模板中创建一个非模态对话框，而Create是从资源中载入对话框模板创建一个非模态对话框

原型：

BOOL CreateIndirect(LPCDLGTEMPLATE lpDialogTemplate, CWnd\* pParentWnd = NULL);  
BOOL CreateIndirect(HGLOBAL hDialogTemplate, CWnd\* pParentWnd = NULL);

参数：  
lpDialogTemplate 包含用于创建对话框的对话框模板的指针。该模板位于DLGTEMPLATE结构和控件信息表单中，可以由强制转换LockResource()的返回值得到。

pParentWnd 指向含有对话框的父窗口对象的指针。如果为NULL，对话框对象的父窗口设置为应用的主窗口。

hDialogTemplate 包含对话框模板的全局内存的句柄。模板使用DLGTEMPLATE结构的形式。

返回值：

如果对话框创建和初始化成功，则返回非零值，否则为0。

## 四、实现步骤

4.1 新建一个MFC规则DLL工程 ，工程名为LanguageRes，插入一个对话框，语言设置为Chinese(中国)，ID为IDD\_CH\_DLG，添加相应的控件。

4.2 复制该对话框资源，将界面文字改成英文，语言设置为English(U.S.)，ID改为与前一个对话框一样，即IDD\_CH\_DLG，如图2所示。

4.3 编译程序，生成LanguageRes.dll。

4.4 新建基于对话框的工程，工程名为MultiLanguage，将4.3中生成的DLL文件放置MultiLanguage的工程目录下。

为其添加一个基于对话框的类Calculate，其模板ID设置为空，即如下代码：enum { IDD = NULL};

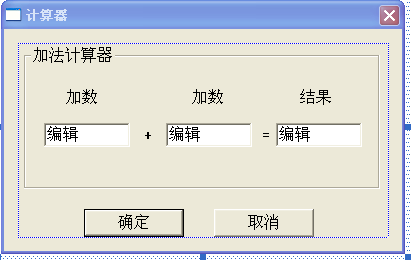
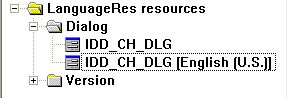
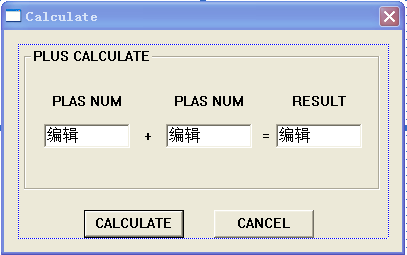
 

图2

4.5 复制Language工程下中Resource.h文件中ID的定义到MultiLanguage工程中的Resource.h文件中，如：



4.6 添加三个CEdit控件m\_PlusOne，m\_PlusTwo，m\_Result变量，对应资源中三个编辑框控件，在DoDataExchange函数中手动将其与相应的ID相关联：



4.7 添加OnCalculte()函数，并为其添加实现代码：

void CCalculate::OnCalculte()

{

CString plusone,plusetwo,ret;

m\_PlusOne.GetWindowText(plusone);

m\_PlusTwo.GetWindowText(plusetwo);

if (plusone.IsEmpty()||plusetwo.IsEmpty())

{

return;

}

int ipluseone=atoi(plusone.GetBuffer(0));

int iplusetwo=atoi(plusetwo.GetBuffer(0));

int iret=ipluseone+iplusetwo;

ret.Format("%d",iret);

m\_Result.SetWindowText(ret);

}

4.8 添加一个CreateDlgFromDll函数，该函数接收一个DLL文件的路径和一个创建类型标识，

HWND CCalculate::CreateDlgFromDll(LPCSTR csFileName,UINT Flag)

{

HMODULE hMod=LoadLibraryEx(csFileName,NULL,LOAD\_LIBRARY\_AS\_DATAFILE);

if (hMod!=NULL)

{

HRSRC hDialog=NULL;//指向资源的句柄

if (Flag==0)

{ hDialog=FindResourceEx(hMod,RT\_DIALOG,MAKEINTRESOURCE(IDD\_CH\_DLG), MAKELANGID(LANG\_CHINESE,SUBLANG\_CHINESE\_SIMPLIFIED));

HGLOBAL hData=LoadResource(hMod, hDialog);

if (hData!=NULL)

{

LPCDLGTEMPLATE lpDialogTemplate=(LPCDLGTEMPLATE)LockResource(hData);

CWnd::CreateDlgIndirect(lpDialogTemplate,NULL);

}

UnlockResource(hData);

}

else

{

hDialog=FindResourceEx(hMod,RT\_DIALOG,MAKEINTRESOURCE(IDD\_CH\_DLG),

MAKELANGID(LANG\_ENGLISH,SUBLANG\_ENGLISH\_US));

HGLOBAL hData=LoadResource(hMod,hDialog);

if (hData!=NULL)

{

LPCDLGTEMPLATE lpDialogTemplate=(LPCDLGTEMPLATE)LockResource(hData);

CWnd::CreateDlgIndirect(lpDialogTemplate,NULL);

}

UnlockResource(hData);

}

}

FreeLibrary(hMod);

return NULL;

}

4.9 为主对话框添加两个对话框类成员变量：

public:

CCalculate m\_CHDialog;

CCalculate m\_ENDialog;

4.10 设计主对话框界面，添加两个按钮，并为其添加响应函数：

void CMultiLanguageDlg::OnCh()

{

::DestroyWindow(m\_CHDialog.m\_hWnd);

m\_CHDialog.CreateDlgFromDll("LanguageRes.dll",0);

::ShowWindow(m\_CHDialog,SW\_SHOW);

}

void CMultiLanguageDlg::OnUs()

{

::DestroyWindow(m\_ENDialog.m\_hWnd);

m\_ENDialog.CreateDlgFromDll("LanguageRes.dll",1);

::ShowWindow(m\_ENDialog,SW\_SHOW);

}

## 五、总结

本实例演示了利用MFC规则DLL封闭资源，并在客户端中以这些资源为模板创建对话框的方法。主要的关键技术有MFC规则DLL的创建、DLL中资源的载入、对话框的手动创建、手动关联控件变量等。

DDL中资源的载入用到的几个关键函数有：

LoadLibraryEx载入DLL、FindResourceEx查找资源、LoadResource载入资源。

对话框的手动创建用到的几个关键函数有：

LockResource()得到资源指针、:CreateDlgIndirect()创建对话框