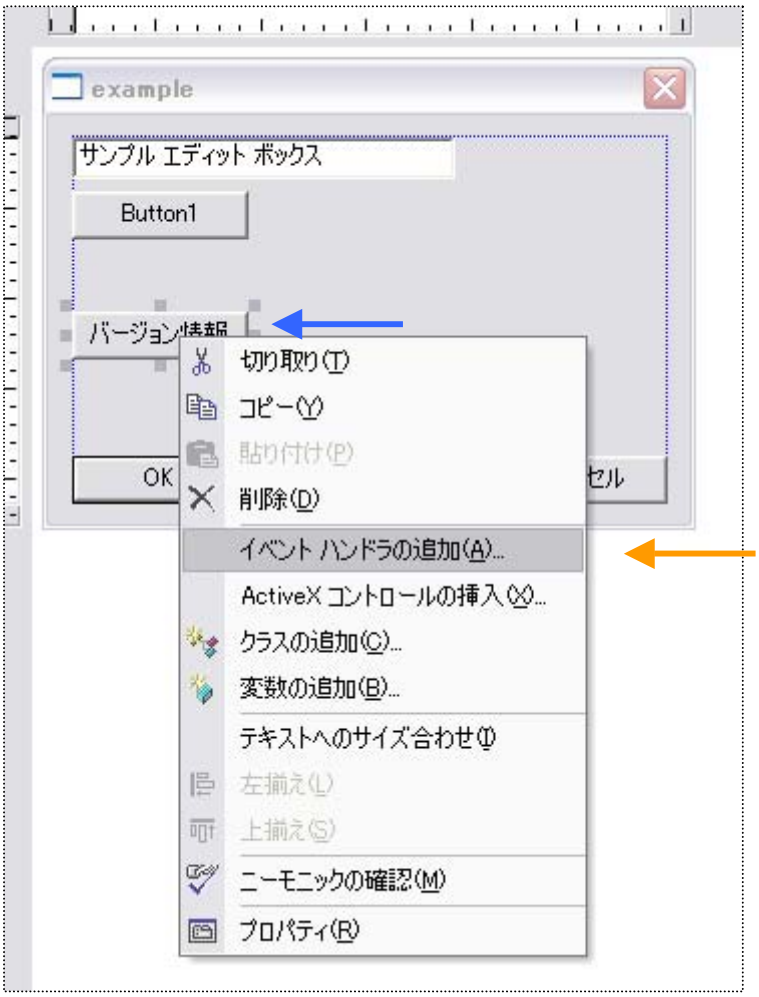
１．複数のダイアログ

バージョン情報を表示するダイアログというのがあります。
MFC で普通にプロジェクトを作成すると、このバージョン情報ダイアログクラスが自動で(勝手に)生成されます。
プロジェクトのダイアログクラスの cpp を開いてください。
一番上に CAboutDlg というクラスの宣言があると思います。
これが、AppWizard によって自動的に作られたダイアログクラスです。

では、早速表示してみましょう。

１）バージョン情報ボタンをダイアログリソースに追加し、クリックへのイベントハンドラを作成します。



2）イベントハンドラを次のように書き換えます。

```
void CexampleDlg::OnBnClickedButton2()
{
	CAboutDlg dlg;	// バージョン情報ダイアログのインスタンスを作成
	dlg.DoModal();	// モーダルで表示
}
```

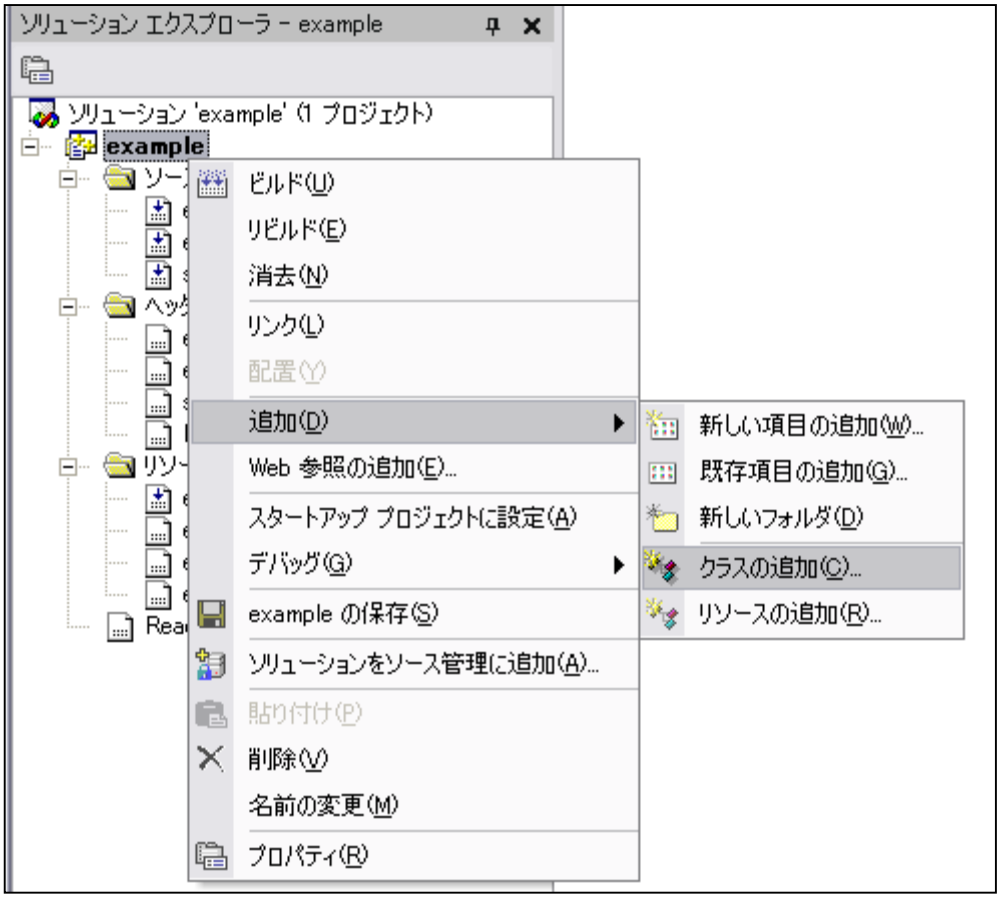
3）これで、バージョン情報ボタンが押下されたときに、バージョン情報ダイアログが表示されるようになりました。

※　コード中にある「モーダルで表示」とは、「そのダイアログが出ている間は、下のウィンドウは触れないようにする」という意味です。バージョン情報ダイアログが出ている間は、下のダイアログは触れなくなっています。

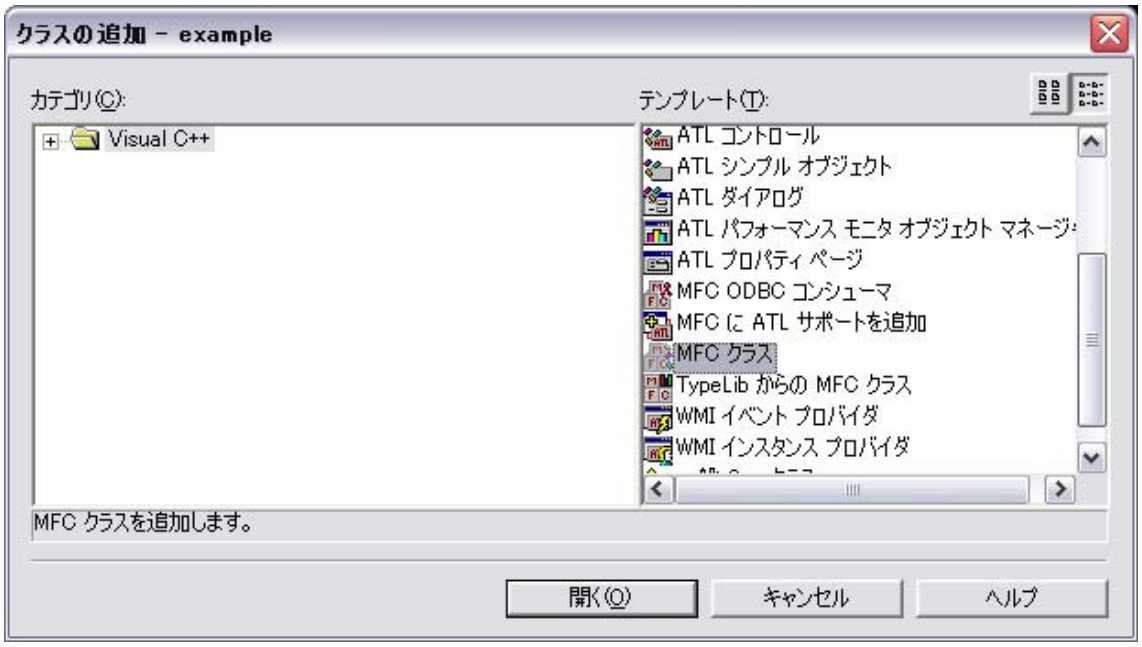
２．ダイアログクラスの作成

今までは全て AppWizard の作ってくれたダイアログのみを表示していましたが、今度は自分でダイアログクラスを作成・表示してみましょう。
ついでに、複数行表示が出来るエディットボックスも作成してみましょう。

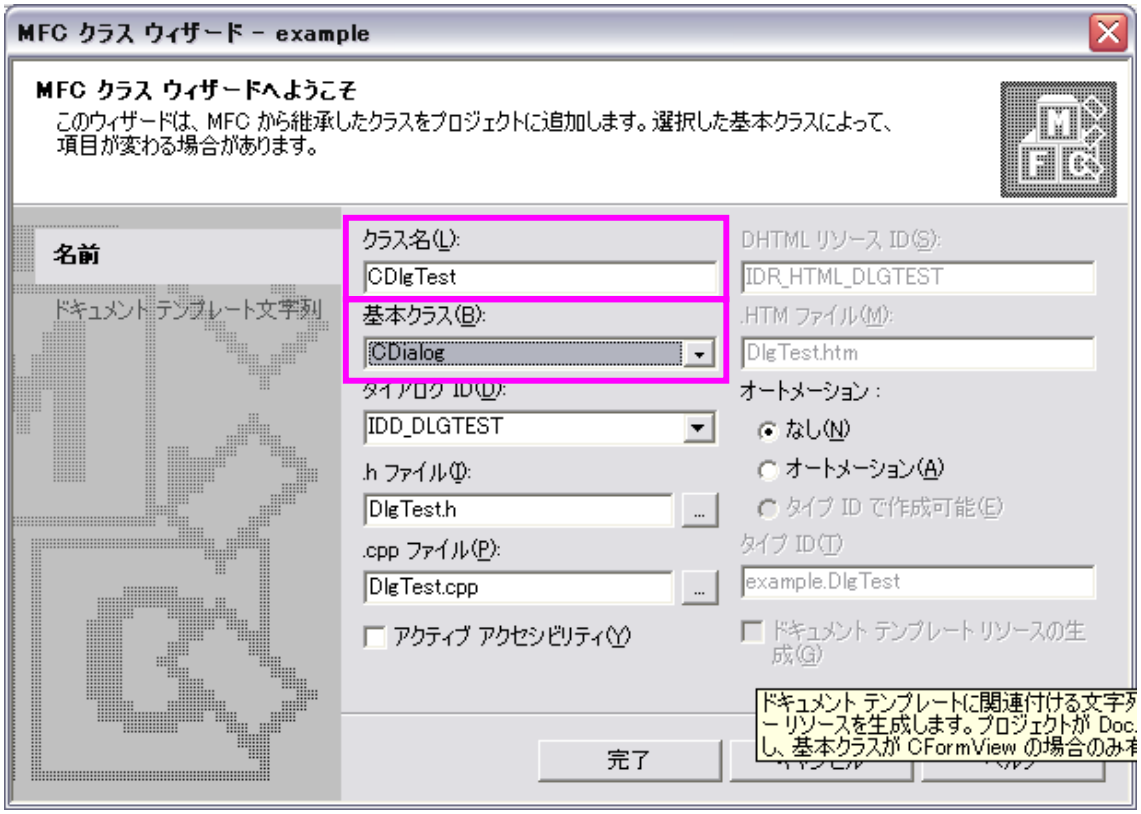
１）ソリューション・エクスプローラーのプロジェクト名の部分を右クリックし、[追加]-[クラスの追加]を選択します。



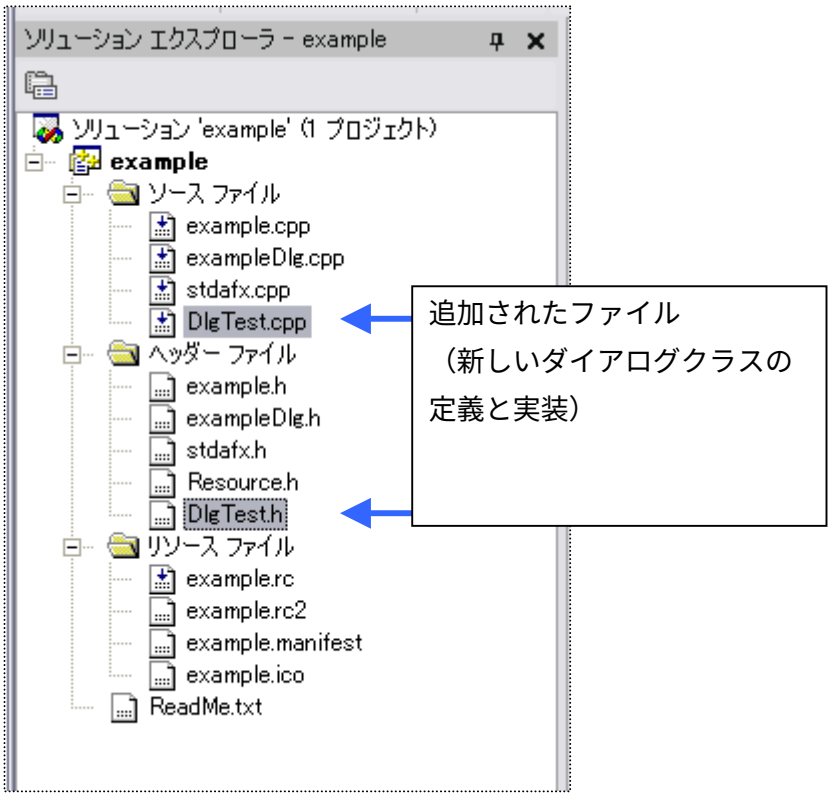
2）クラスの追加ダイアログボックスが表示されるので、「テンプレート」から『MFC クラス』を選択します。



3）MFC クラスウィザードダイアログが表示されるので、新しいダイアログクラスの名前を入力し、基本クラスに「CDialog」を選択します。
ダイアログ ID は、変更したい場合は変更しましょう。例では、CDlgTest というダイアログクラスを作成しています。



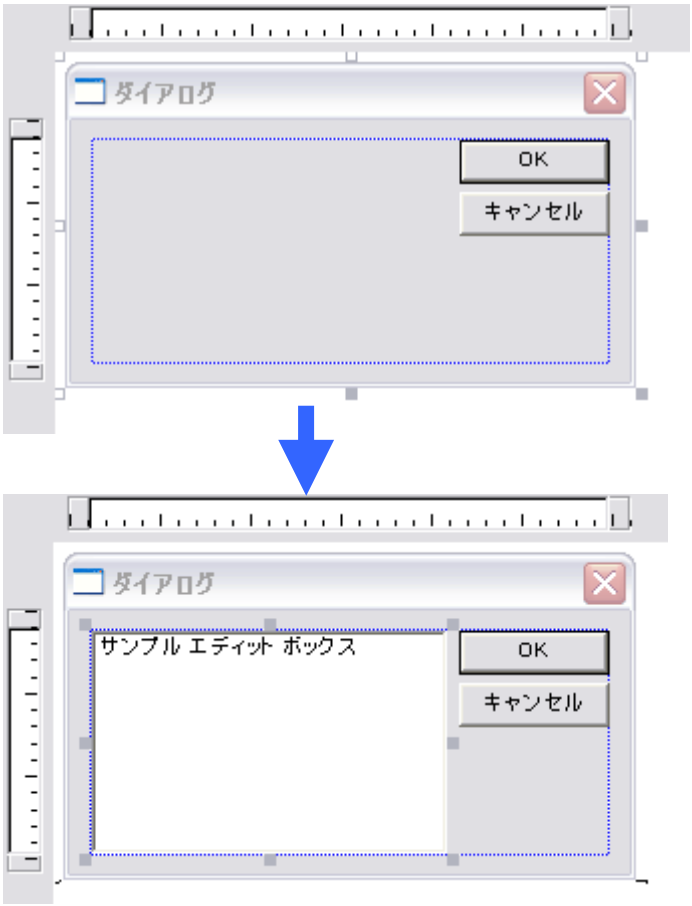
4）完了を押下すると、新しいダイアログクラスの定義ファイル(.h)と実装ファイル(cpp)、加えて指定したダイアログ ID をもつダイアログがリソースに追加されています。



ソリューション・エクスプローラー

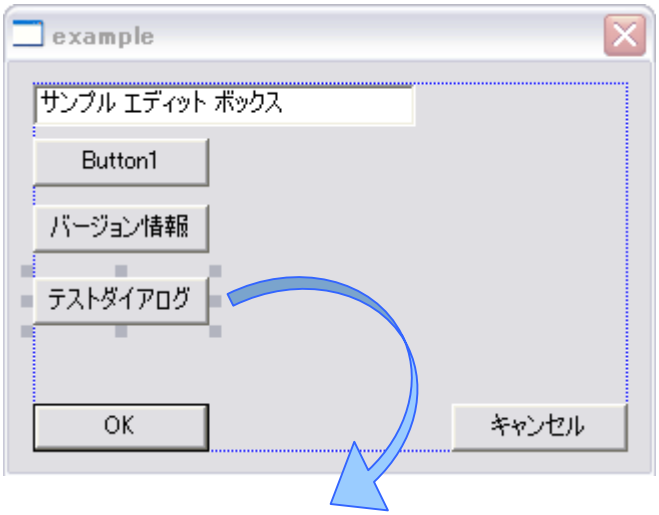
リソースビュー

5）リソースビューから、追加されたダイアログリソースをダブルクリックして、ダイアログエディタを開いてください。OK ボタンとキャンセルボタンしかないダイアログが表示されます。これが、今作成したダイアログクラスの外見です。ここに、エディットボックスを「なるべく縦の幅を広く」配置してください。



6）エディットボックスのプロパティを見てみると、「MultiLine」(複数行表示の許可)と「Want Return」(Enter キー押下のメッセージを受け取る)というプロパティが存在します。それぞれ、初期値は False ですが、これを True にすると、複数行の入力が可能なエディットボックスになります。

7）では、メインのダイアログにまたボタンを一つ配置し、クリックのイベントハンドラを作成し、新しく作成したダイアログをモーダルで呼ぶよう記述してください。




```
void CexampleDlg::OnBnClickedButton3()
{
	CDlgTest dlg;
	int ret = dlg.DoModal();
}
```

また、同ファイルにてダイアログクラスのヘッダーをインクルードしてください。

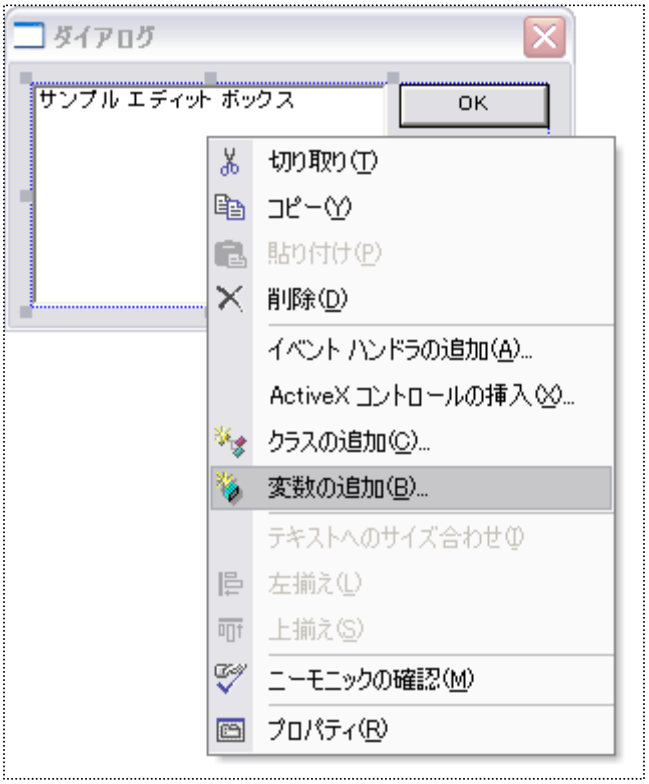
```
#include "DlgTest.h"
```

8）ビルドして、実行すると、ボタン押下とともに作成したダイアログが表示されます。また、エディットボックスは複数行入力が可能となっています。

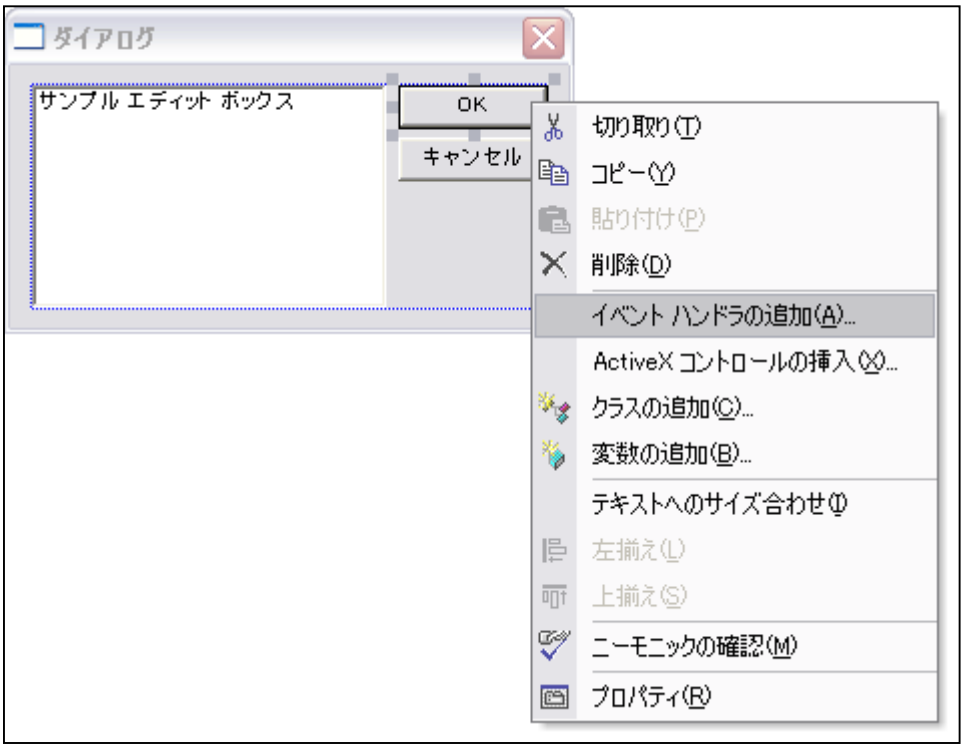
３．一時ダイアログからの値の取得

今度は、今作ったばっかりのダイアログから、値を取得してみましょう。せっかく複数行入力のできるエディットボックスを作ったので、そこから値を一時ダイアログに保持し、呼び出し元で取得してみます。
また、ダイアログで OK ボタンが押されたかキャンセルボタンが押されたかの判断もしてみましょう。

１）前項で作成したダイアログリソースをエディタで開き、エディットボックスのメンバ変数を作成します。



2）同じく OK ボタンについて、クリック時のイベントハンドラを作成します。



3）作成したダイアログの定義ファイル(.h)を開き、CString のメンバを追加します。

```
#pragma once
// CDlgTest ダイアログ
class CDlgTest : public CDialog
{
	DECLARE_DYNAMIC(CDlgTest)

public:
	CDlgTest(CWnd* pParent = NULL);   // 標準コンストラクタ
	virtual ~CDlgTest();

// ダイアログ データ
	enum { IDD = IDD_DLGTEST };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート

	DECLARE_MESSAGE_MAP()
public:
	CEdit   m_EditMulti;
	CString m_sValue;
};
```

4）OK ボタンのイベントハンドラの中身で、CString のメンバにエディットボックスの入力値を取得します。これは、OK ボタンが押されてダイアログが消えてしまうと、エディットコントロールである m_EditMulti も一緒に使用不可能になってしまうため、CString の変数に保持しています。ちなみに、OK ボタンのイベントハンドラ内にある OnOK()関数は、「DoModal()関数に対し、IDOK を戻り値として返し、ダイアログを閉じる」という動作が定義されている、CDialog クラスのメソッドです。

```
void CDlgTest::OnBnClickedOk()
{
	m_EditMulti.GetWindowText(m_sValue);
	OnOK();
}
```

5）メインのダイアログのハンドラを、次のように修正します。

```
void CExampleDlg::OnBnClickedButton3()
{
	CDlgTest dlg;
	int ret = dlg.DoModal();
	if(ret == IDOK){
		AfxMessageBox(dlg.m_sValue);
	}
}
```

DoModal()関数は、CDlgTest ダイアログが表示され、OK ボタンかキャンセルボタンが押されるまで返ってきません。そして、上記の ret に何が入るかというと、OK ボタンが押されたときには IDOK、キャンセルボタンが押されたときには IDCANSEL が入ります。ちなみに IDOK と IDCANCEL とは、Win32API のヘッダーファイルの中で次のように定義されています。

```
#define IDOK		1
#define IDCANCEL	2
```

OK ボタンが押されると、ダイアログの OK ボタンクリックハンドラによって m_sValue にはエディットボックスの値が入っているはずです。それをメッセージボックスで表示しています。