Lab 7.1: Adding a Splitter Bar

Objectives

After completing this lab, you will be able to:

- Add a class to an application by using the ClassWizard.
- Add a static splitter bar to a window to create a split pane.

Prerequisites

Familiarity with the topics covered in this chapter.

Lab Setup

To run the solution to this lab, click this icon.

To see a demonstration of the solution for this lab, click this icon.

Estimated time to complete this lab: **20 minutes**.

Exercises

The following exercises provide practice working with the concepts and techniques covered in this chapter.

Exercise 1: Adding a Splitter Window

In this exercise, you will add a splitter bar to an SDI window.

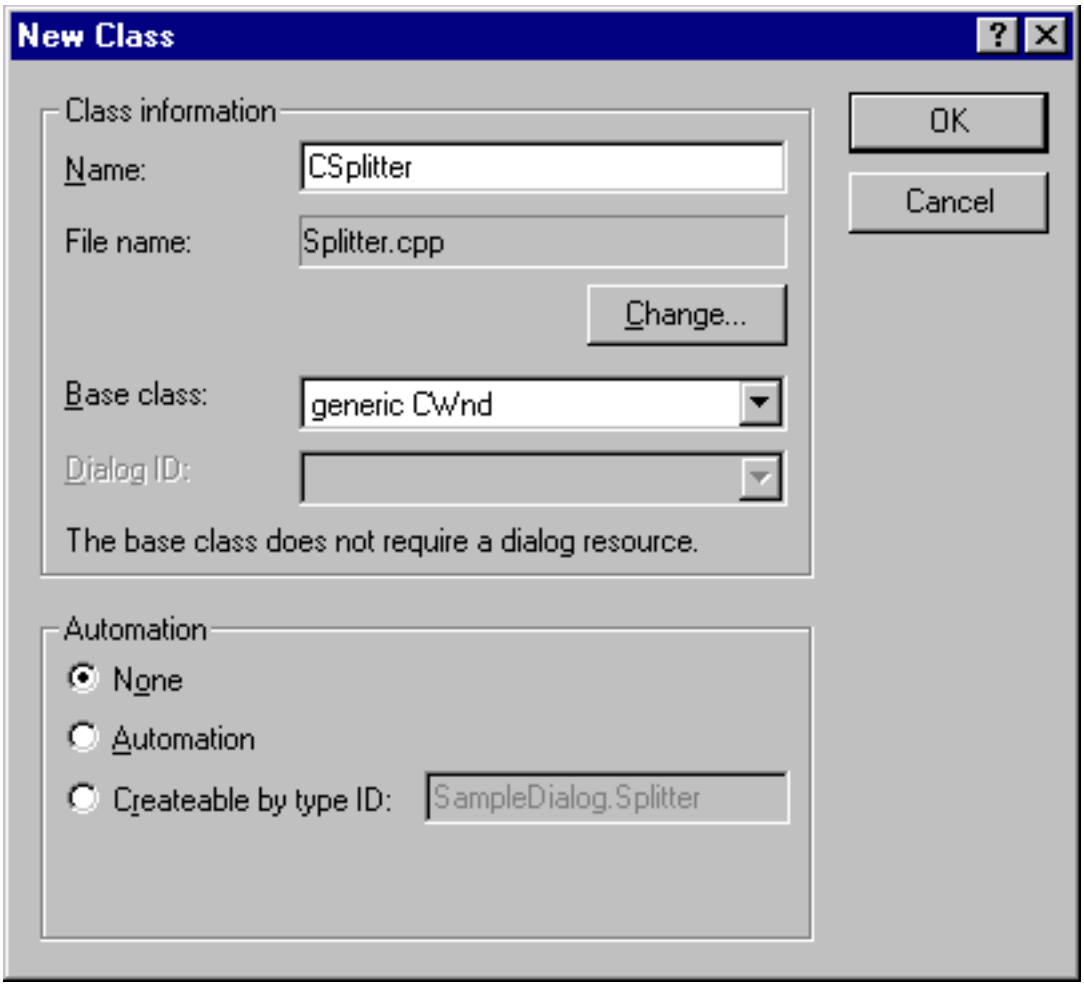
For this lab, you will use the project from Chapter 3, Lab 1. You can copy this from \Labs\C07\Lab01\Baseline. The completed code for these exercises is in \Labs\C07\Lab01\Xxx, where Xxx is the exercise number.

Exercise 1: Adding the Splitter Window

If you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C07\Lab01\Baseline.

➢ Create a new CSplitter class

1. Open the Diff project.
2. From the View menu, choose the ClassWizard, or press CTRL+W.
3. Click Add Class, and then click New.
4. Set the name of the class to **CSplitter**, and base the class on the generic **CWnd** class. Accept the defaults for the other fields. Click OK.



Note The ClassWizard does not present **CSplitterWnd** as a base class. You can change the base class directly.

5. Edit Splitter.H. Click **CSplitter** in FileView, then choose Go to Definition.

   Change the declaration line of **CSplitter** from:

   ```
   class CSplitter : public CWnd
   ```

   to:

   ```
   class CSplitter : public CSplitterWnd
   ```

6. Add a method to get the protected width of the splitter window to the public attributes section of the **CSplitter** definition:

   ```
   int GetSplitterWidth() const { return m_cxSplitter; }
   ```

7. Open Splitter.Cpp and update the message map declaration. It reads:

   ```
   BEGIN_MESSAGE_MAP(CSplitter, CWnd)
       //{{AFX_MSG_MAP(CSplitter)
   ```

8. Change the first line to reference **CSplitterWnd** instead of **CWnd**.

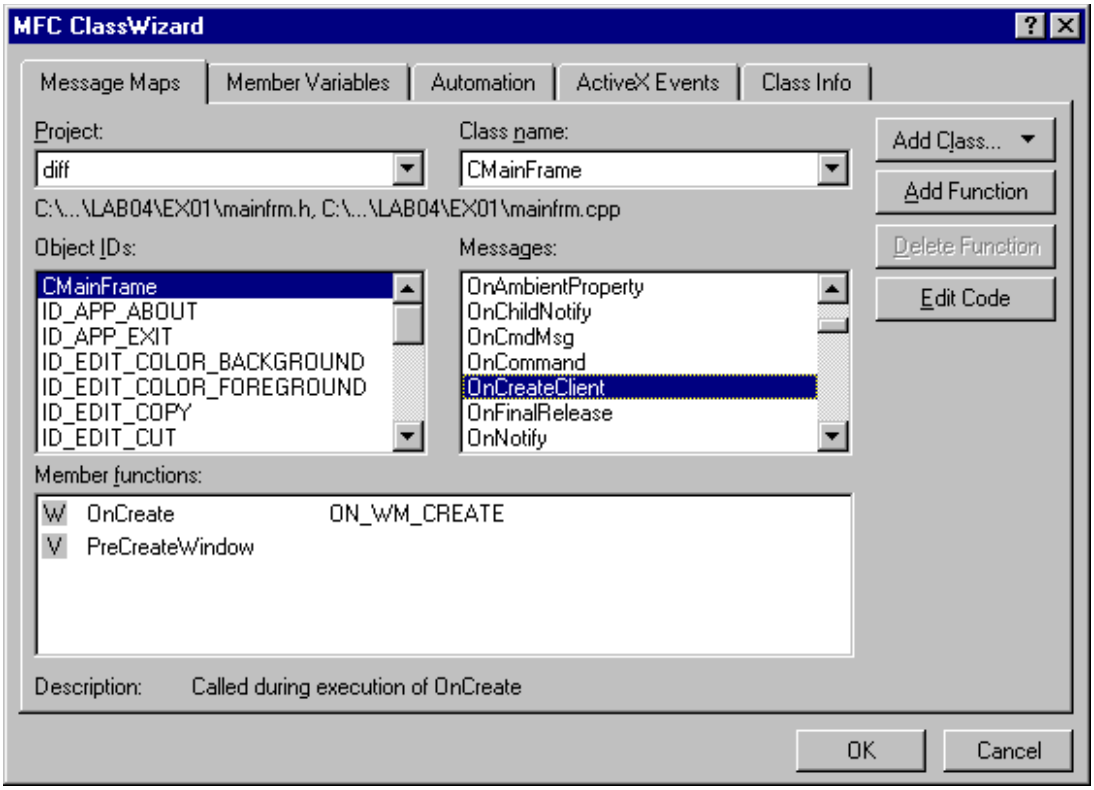
   ```
   BEGIN_MESSAGE_MAP(CSplitter, CSplitterWnd)
   ```

➢ **Add a reference to the splitter in the MainFrame object**

1. Open MainFrm.H.
2. Add a protected **CSplitter** member to the class definition.

```
// splitter bar embedded members
CSplitter   m_wndSplitter;
```

3. Use the ClassWizard to add an **OnCreateClient** handler; using the Message Map tab in the ClassWizard, choose the **CMainFrame** class and object ID, and the OnCreateClient message. Click Add Function, and then click OK in the ClassWizard. The ClassWizard creates a reference to the **OnCreateClient** handler in MainFrm.H. and a blank implementation in MainFrm.Cpp.



The major implementation task in adding a Splitter Bar to an application is to create the split window itself. Because you allow only two files to be active, you use static splitter windows. Put the bar in the middle of the frame. For more information, see "Working with Frame Windows, Documents and Views" in the Visual C++ online documentation.

➢ Add code to the OnCreateClient handler to create the splitter window

1. Create the static splitter by adding a call to **CSplitterWnd::CreateStatic**.

```
m_wndSplitter.CreateStatic (this, 1, 2, WS_CHILD);
```

2. **CSplitterWnd::CreateStatic** sets up a constant number and arrangement of splitter panes.
3. Size the splitter windows to two equal panes.

```
SIZEsize;
CRect   rect;
GetClientRect(&rect);

size.cx = (rect.right - m_wndSplitter.GetSplitterWidth())/2;
size.cy = rect.bottom;
```

4. Attach the views to the windows.

```
m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CDiffView), size,     pContext);
m_wndSplitter.CreateView(0,1,RUNTIME_CLASS(CDiffView), size,     pContext);
SetActiveView((CView *)m_wndSplitter.GetPane(0,1));
```

5. Show the splitter window.

```
m_wndSplitter.ShowWindow(SW_SHOWNORMAL);
m_wndSplitter.UpdateWindow();
```

6. The complete function follows.

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
    CCreateContext* pContext)
{
    // Create a static splitter window with two side-by-side panes

    if(!m_wndSplitter.CreateStatic (this,1,2,WS_CHILD))
    {
        return FALSE;
    }

    // Calculate the size of the splitter panes
    SIZEsize;
    CRect   rect;
    GetClientRect(&rect);

    size.cx = (rect.right - m_wndSplitter.GetSplitterWidth())/2;
    size.cy = rect.bottom;

    //set the views
    m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CDiffView), size,
        pContext);
    m_wndSplitter.CreateView(0,1,RUNTIME_CLASS(CDiffView), size,
        pContext);
    SetActiveView((CView *)m_wndSplitter.GetPane(0,1));

    //show the splitter
    m_wndSplitter.ShowWindow(SW_SHOWNORMAL);
    m_wndSplitter.UpdateWindow();

    return TRUE;
}
```

➢ Clean up, build, and run the Diff application

1. In MainFrm.Cpp, include Splitter.H, DiffDoc.H. and DiffView.H. Splitter.H. must be included before MainFrm.H, because there is a **CSplitter** member of **CMainFrame**.

```
#include "splitter.h"
#include "MainFrm.h"
#include "diffdoc.h"
#include "diffview.h"
```

2. In Diff.Cpp, include Splitter.H.

```
#include "splitter.h"
#include "MainFrm.h"
```

3. From the Build menu, choose Build Diff.Exe.

4. From the Build menu, choose Run Diff.Exe. Notice the splitter bar; you can reposition the bar, but you cannot delete or add splitters.

The completed code for this exercise is in \Labs\C07\Lab01\Ex01.