Integration and configuration

How to integrate CTreePropSheetEx

Using `CTreePropSheetEx` is very similar to using `CPropertySheet` or `CTreePropSheet`. At this point, if you haven't already done so, I would recommend that you read Sven's article regarding CTreePropSheet [^]. In the following explanation, I will assume that you already have a property sheet and its pages created. To use the class, you need to:

1. Add the following files to your project:

File names (37 files)	Description
TreePropSheetEx.h *TreePropSheetEx.cpp*	`CTreePropSheetEx` class.
TreePropSheetBordered.h *TreePropSheetBordered.cpp*	`CTreePropSheetBordered` class, derived from `CTreePropSheetEx`, draws a border around the frame when XP theme is not available.
TreePropSheetOffice2003.h *TreePropSheetOffice2003.cpp*	`CTreePropSheetOffice2003` class, derived from `CTreePropSheetEx`, provides a look and feel similar to Office 2003 option dialogs.
TreePropSheetBase.h *TreePropSheetBase.cpp*	`CTreePropSheetBase` class, base class for `CTreePropSheetEx`.
TreePropSheetTreeCtrl.h *TreePropSheetTreeCtrl.cpp*	`CTreePropSheetTreeCtrl` class, `CTreeCtrl` derived allowing custom color for each tree item. Based on CTreeCtrlEx[^].
PropPageFrame.h *PropPageFrame.cpp*	`CPropPageFrame` class. Abstract base class for the frame drawn around pages when in tree mode.
PropPageFrameEx.h *PropPageFrameEx.cpp*	`CPropPageFrameEx` class. `CPropPageFrame` implementation that provides flicker-free frame. It is the class used by default by `CTreePropSheetBase` and `CTreePropSheetEx`.
PropPageFrameBordered.h *PropPageFrameBordered.cpp*	`CPropPageFrameBordered` class. Derived from `CPropPageFrameEx`, extends it by drawing a border around the frame when XP theme is not available.
PropPageFrameOffice2003.h *PropPageFrameOffice2003.cpp*	`CPropPageFrameOffice2003` class. Derived from `CPropPageFrameEx`, extends it by providing a look and feel similar to Office 2003 option dialogs.
TreePropSheetSplitter.h *TreePropSheetSplitter.cpp*	`CTreePropSheetSplitter` class. Based on CSimpleSplitterWnd[^], implements the splitter class.
TreePropSheetUtil.hpp	Misc. utility classes.
ThemeLibEx.h *ThemeLibEx.cpp*	Helper class for accessing XP theme functions.
TreePropSheetResizableLibHook.h *TreePropSheetResizableLibHook.cpp*	Implements the resizable library using message hooking rather than inheritance.

ResisableGrip.h *ResizableGrip.cpp* *ResizableLayout.h* *ResizableLayout.cpp* *ResizableMinMax.h* *ResizableMinMax.cpp* *ResizableMsgSupport.h* *ResizableMsgSupport.inl* *ResizableState.h* *ResizableState.cpp*	Resizable library[^] classes.
Hookwnd.h *Hookwnd.cpp*	CHookWnd[^] class. Defines the interface for an MFC class to implement message hooking.
memDC.h	CMemDC[^] class. Implements a memory Device Context which allows flicker free drawing.
HighColorTab.hpp	Dynamically updates the tab's image list to add 16M color icons support.
ResizablePage.h *ResizablePage.cpp*	Resizable library class for property pages[^]. Not directly used by `CTreePropSheetEx` but useful to add resizing capability to property pages.

2. Use `CTreePropSheetEx`, `CTreePropSheetBordered` or `CTreePropSheetOffice2003` instead of `CPropertySheet`. If you already have a class deriving from `CPropertySheet`, you need to derive from `CTreePropSheetEx` (or `CTreePropSheetBordered` or `CTreePropSheetOffice2003`) instead. You also need to replace all references to `CPropertySheet` by `CTreePropSheetEx` (or `CTreePropSheetBordered` or `CTreePropSheetOffice2003`). If you are using a `CPropertySheet` object directly, just change its type to `CTreePropSheetEx` (or `CTreePropSheetBordered` or `CTreePropSheetOffice2003`).

   Note: `CTreePropSheetEx`, `CTreePropSheetBordered` and `CTreePropSheetOffice2003` are in the namespace `TreePropSheet`.

   Note: `CHookWnd` uses `CCriticalSection`. This class is defined in *<afxmt.h>*, therefore you should add the following line to your precompiled header file:

   ```
   #include <afxmt.h>
   ```

   Also, `CPropPageFrame` uses `dynamic_cast` which means that RTTI support should be enabled.

3. If you choose to have the sheet resizable (which is the option enabled by default), you will also have to add resizing support to your property pages. The easiest way to do this is to use Paolo Messina's `CResizablePage` class. The following code snippet demonstrates how this is done for one of the property pages in the sample application:
   a. Change the base class for your property page from `CPropertyPage` to `CResizablePage`.
   b. Edit the `OnInitDialog` method of each page in order to add the sizing constraints for the controls inside the page. For instance, in one of the pages in the demo, this looks like this:

```
BOOL CPageEmail::OnInitDialog()
{
  CResizablePage::OnInitDialog();

  // Preset layout
  AddAnchor(IDC_EMAIL1, TOP_LEFT, TOP_RIGHT);
  AddAnchor(IDC_EMAIL2, TOP_LEFT, TOP_RIGHT);
  AddAnchor(IDC_EMAIL3, TOP_LEFT, TOP_RIGHT);
  AddAnchor(IDC_COMBO_DEFAULT_EMAIL, TOP_LEFT, TOP_RIGHT);

  return TRUE;
}
```

For more information, you should read Paolo's article[^].

4. If you are using `CTreePropSheetOffice2003`, you need to customize each property page in order to draw a white background (or more accurately, a background with the system color `COLOR_WINDOW`). To do so, you need to handle `WM_CTLCOLOR` for each property page. This message is sent by each child control to the parent page before it is drawn and let the parent prepare the DC before the control is rendered. The new message handler should be as follows:

```
HBRUSH CPageDates::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
  pDC->SetBkMode(TRANSPARENT);
  return ::GetSysColorBrush( COLOR_WINDOW );
}
```

This code is a minimal default implementation. You can look at the demo project to see how it is possible to update property pages so that they can be rendered in the 'standard mode' and the Office 2003 mode.