# CMFCPropertyGridCtrl的简单教程

 weixin_34216107 于 2012-08-07 12:29:08 发布 

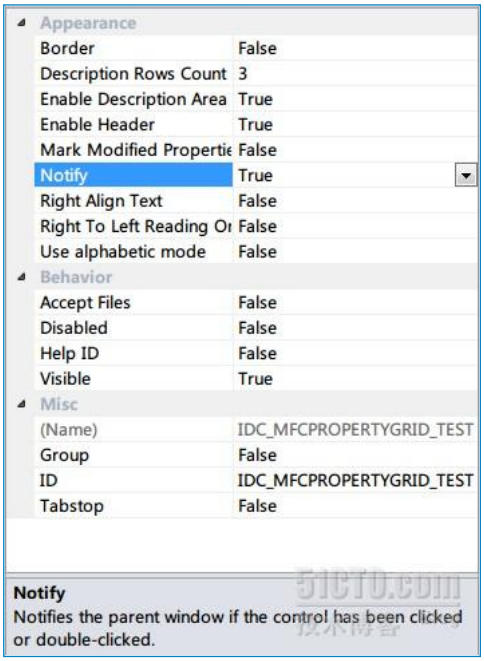
我写博客从一开始还是日经一文，到后来是周经一文，现在都直接变成月经一文了。。

闲话少说，最近的一个需求是把一些文字跟下拉框 集合 、对应在一起，如果就那样堆几个static、comboBox平铺在对话框上，不免显得有些单调，而且文字对齐啥的也麻烦，还要排版。找来找去发现了CMFCPropertyGridCtrl这个控件。找了点教程，发现这个控件使用起来还是蛮简单的，关键一点就是要搞清楚层次关系，才不至于混乱，谁是谁的子项，谁是谁的组员等等。

老惯例，上例子。

用vs2010建立一个基于对话框的 MFC 工程，拖一个CMFCPropertyGridCtrl进去，大小调整好。（注：首先采用的是静态创建的办法，大部分需要的属性在对话框编辑界面就可以编辑。）然后为该控件更改ID为IDC_MFCPROPERTYGRID_TEST，并且使用ClassWizard为控件添加一个变量m_propertyGrid。

接下来更改控件的一些属性。




其实这不就是个CMFCPropertyGridCtrl控件么。Border神马的基础东西就不说了。

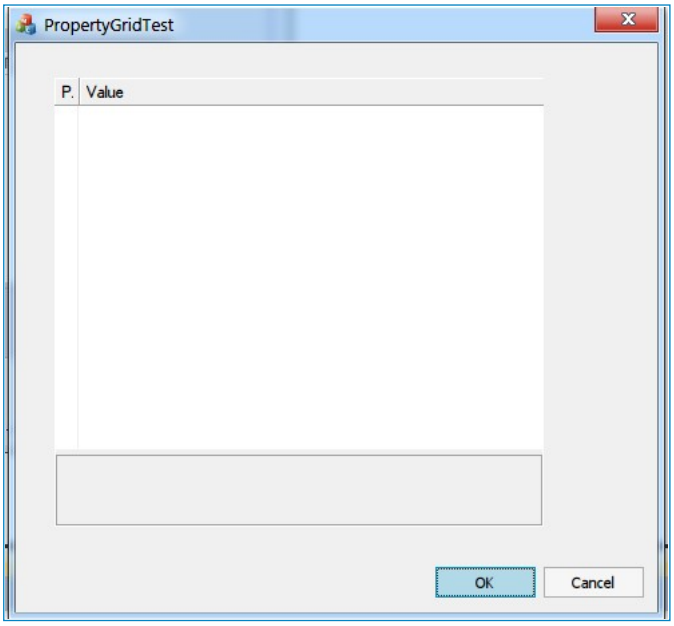
"Description Rows Count"指的是下面的描述部分有几行。

"Enable Description Area"表示是否启动下面的描述功能

"Enable Header"表示是否启动表头

"Mark Modified Properties"表示是否着重显示更改项

可以按照需求来进行设置。这里先使用默认的设置。先编译运行一下，比较简陋。

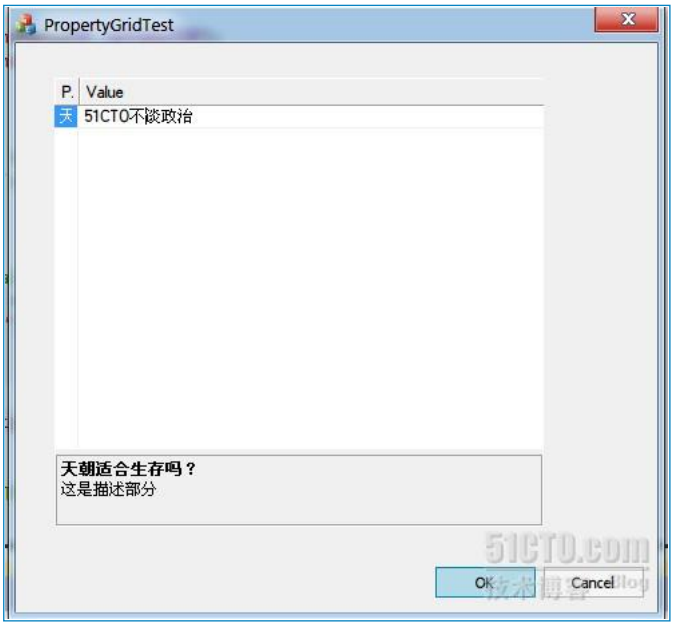



好，接下来该添加东西进去了。

在OnInitDialog中添加如下代码，我会一行一行解释。

```
1. CMFCPropertyGridProperty * pProp1 = new CMFCPropertyGridProperty(
2.         _T("天朝适合生存吗？"),
3.         _T("51CTO不谈政治"),
4.         _T("这是描述部分"));
5.
6. m_propertyGrid.AddProperty(pProp1);
```

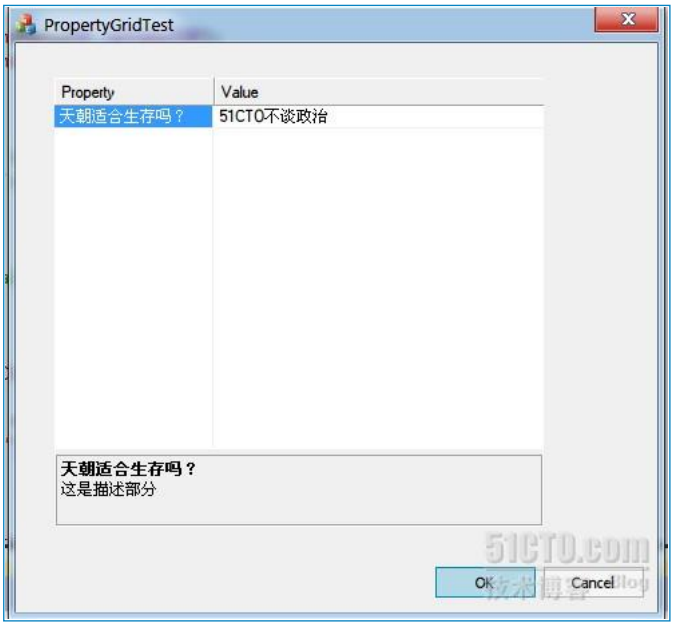
第一行是建立一个新的项目，也是最普通的项目，CMFCPropertyGridProperty。与这种项目同级的还有CMFCPropertyGridColorProperty、CMFCPropertyGridFontProperty以及CMFCPropertyGridFileProperty，等会都可以试一下。调用构造函数时传入的三个参数分别是条目名称、默认选项及描述文字。运行一下就知分晓。




饿滴神啊，肿么这个样子。不过该有的全有，只需要设置一下就行。这里得提一笔，微软似乎非常喜欢把第一列的宽度设置为“只能容得下一个普通的5号小宋体的宽度”，不光是CMFCPropertyGrid，连CListCtrl也是如此，需要动点特殊的手段才能调整过来。在这段代码的前面加这么几句：

```
1. HDITEM item;
2. item.cxy=120;
3. item.mask=HDI_WIDTH;
4. m_propertyGrid.GetHeaderCtrl().SetItem(0, new HDITEM(item));
```

如此再运行，就会比较好看了。

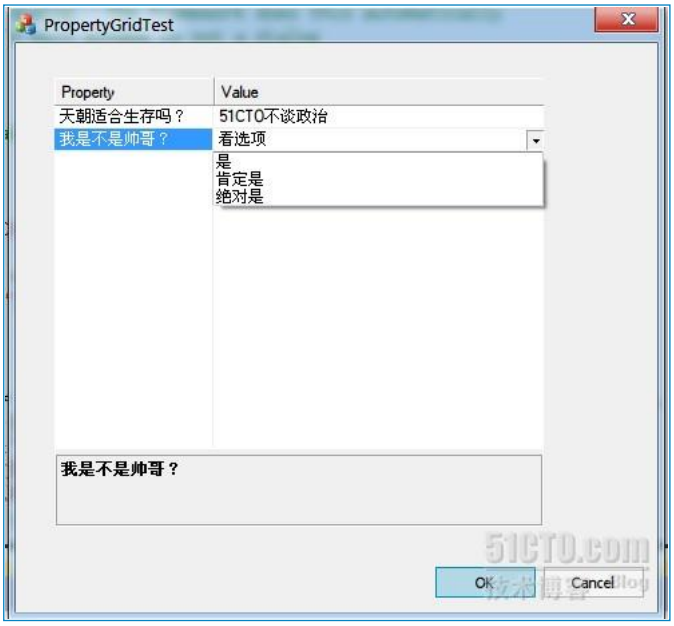



好，接下来我们看一下右边的value部分还能换成神马东西。

如同vs2010里提供的“属性”栏一样，这CMFCPropertyGridCtrl应该也支持下拉菜单，
好，就来添加下拉菜单看看。修改刚才的代码：

```
1. CMFCPropertyGridProperty* pProp2 = new CMFCPropertyGridProperty(
2. _T("我是不是帅哥？"),
3.     _T("看选项"),
4. _T(""));
5. pProp2->AddOption(_T("是"));
6. pProp2->AddOption(_T("肯定是"));
7. pProp2->AddOption(_T("绝对是"));
8. pProp2->AllowEdit(FALSE);  //不允许对选项进行编辑
9.
10. m_propertyGrid.AddProperty(pProp2);
```

然后运行，就会如愿以偿地出现下拉框了。

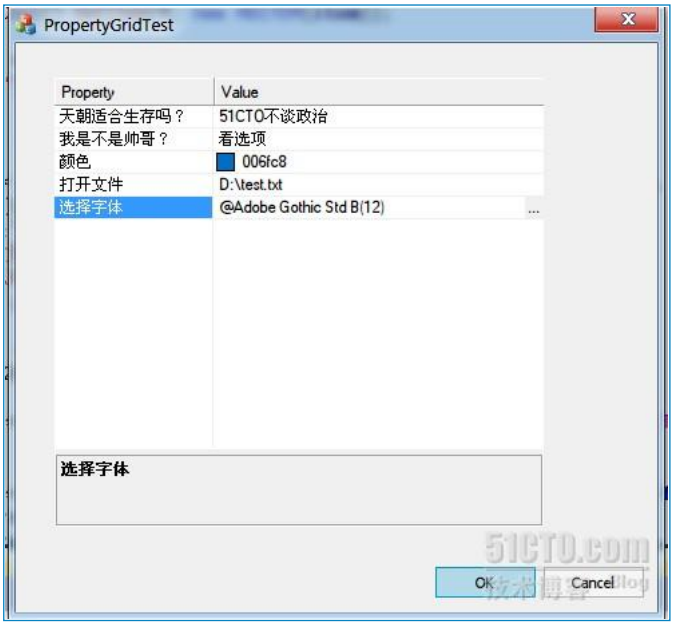



接下来是另外三个同级的项目：

```
1. CMFCPropertyGridColorProperty * pProp3 = new CMFCPropertyGridColorPro
   perty(
2.     _T("颜色"), RGB(0, 111, 200));
3. m_propertyGrid.AddProperty(pProp3);
4.
5. CMFCPropertyGridFileProperty * pProp4 = new CMFCPropertyGridFilePropert
   y(
6. _T("打开文件"), TRUE, _T("D:\\test.txt"));
7. m_propertyGrid.AddProperty(pProp4);
8.
9. LOGFONT font = {NULL};
10. CMFCPropertyGridFontProperty * pProp5 = new CMFCPropertyGridFontProp
    erty(
11. _T("选择字体"), font);
12. m_propertyGrid.AddProperty(pProp5);
```

注：每一种类型的项目都有2个或3个重载函数，可以自己根据需求慢慢挖掘，在这里就不赘述了。

运行效果如下：

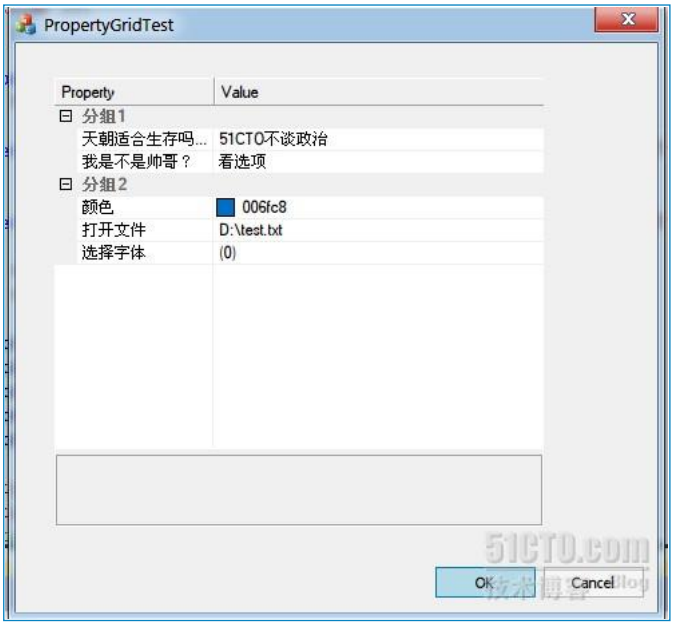



这么些不同种类的东西乱七八糟堆在一起，是不是有点不科学？那么就引入下一个概念：分组。回到第一张图，vs2010的“属性”栏分了三个组，分别是Apperance、Behavior和Misc，看起来就清晰多了，我们也可以。

好，来重新构建一下我们的代码：

```
1. CMFCPropertyGridProperty * group1 = new CMFCPropertyGridProperty(_T("分组1"));
2. CMFCPropertyGridProperty * group2 = new CMFCPropertyGridProperty(_T("分组2"));
3.
4. group1->AddSubItem(pProp1);
5. group1->AddSubItem(pProp2);
6. group2->AddSubItem(pProp3);
7. group2->AddSubItem(pProp4);
8. group2->AddSubItem(pProp5);
9.
10. m_propertyGrid.AddProperty(group1);
11. m_propertyGrid.AddProperty(group2);
```

编译运行效果如下：




至此，静态创建CMFCPropertyGridCtrl的方法就结束了。

还有一种方法是动态创建，与CStatic、CEdit等控件无二，在创建之后也可以利用自带的函数修改控件的属性，如：

1. CMFCPropertyGridCtrl * propertyGrid = new CMFCPropertyGridCtrl;
2. propertyGrid->Create(WS_CHILD | WS_BORDER | WS_VISIBLE, CRect(400, 100, 600, 200), this, WM_USER + 100);
3. propertyGrid->EnableHeaderCtrl(TRUE); //使用表头
4. propertyGrid->SetVSDotNetLook(); //使用样式
5. propertyGrid->MarkModifiedProperties(); //着重显示更改过的部分

更多属性留待各位发掘一下啦。

转载于:https://blog.51cto.com/serious/956984

**相关资源：** CMFCPropertyGridCtrl使用_CMFCPropertyGridCtrl-C++代码类资源...

显示推荐内容