树形控件TreeCtrl PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/472/2021_2022__E6_A0_91_E5_BD_A2_E6_8E_A7_E4_c67_472303.htm 树形控件可以用于树形的结构，其中有一个根接点(Root)然后下面有许多子结点，而每个子结点上有允许有一个或多个或没有子结点。MFC中使用CTreeCtrl类来封装树形控件的各种操作。通过调用BOOL Create(DWORD dwStyle, const RECTamp.m_list,TVSIL_NORMAL). m_tree.InsertItem("Parent1",0,1).//添加， 选中时显示图标1，未选中时显示图标0 此外CTreeCtrl还提供了一些函数用于得到/修改控件的状态。 HTREEITEM GetSelectedItem( ).将返回当前选中的结点的句柄。BOOL SelectItem(HTREEITEM hItem).将选中指明结点。 BOOL GetItemImage(HTREEITEM hItem, intamp. nSelectedImage) / BOOL SetItemImage(HTREEITEM hItem, int nImage, int nSelectedImage)用于得到/修改某结点所使用图标索引。 CString GetItemText(HTREEITEM hItem) /BOOL SetItemText(HTREEITEM hItem, LPCTSTR lpszItem).用于得到/修改某一结点的显示字符。 BOOL DeleteItem(HTREEITEM hItem).用于删除某一结点，BOOL DeleteAllItems( ).将删除所有结点。 此外如果想遍历树可以使用下面的函数： HTREEITEM GetRootItem( ).得到根结点。HTREEITEM GetChildItem(HTREEITEM hItem).得到子结点。 HTREEITEM GetPrevSiblingItem/GetNextSiblingItem(HTREEITEM hItem).得到指明结点的上/下一个兄弟结点。HTREEITEM GetParentItem(HTREEITEM hItem).得到父结点

。 树形控件的消息映射使用ON_NOTIFY宏，形式如同：ON_NOTIFY(wNotifyCode, id, memberFxn)，wNotifyCode为通知代码，id为产生该消息的窗口ID，memberFxn为处理函数，函数的原型如同void OnXXXTree(NMHDR* pNMHDR, LRESULT* pResult)，其中pNMHDR为一数据结构，在具体使用时需要转换成其他类型的结构。对于树形控件可能取值和对应的数据结构为： TVN_SELCHANGED 在所选中的结点发生改变后发送，所用结构：NMTREEVIEW TVN_ITEMEXPANDED 在某结点被展开后发送，所用结构：NMTREEVIEW TVN_BEGINLABELEDIT 在开始编辑结点字符时发送，所用结构：NMTVDISPINFO TVN_ENDLABELEDIT 在结束编辑结点字符时发送，所用结构：NMTVDISPINFO TVN_GETDISPINFO 在需要得到某结点信息时发送，（如得到结点的显示字符）所用结构：NMTVDISPINFO 关于ON_NOTIFY有很多内容，将在以后的内容中进行详细讲解。 关于动态提供结点所显示的字符：首先你在添加结点时需要指明lpszItem参数为：LPSTR_TEXTCALLBACK。在控件显示该结点时会通过发送TVN_GETDISPINFO来取得所需要的字符，在处理该消息时先将参数pNMHDR转换为LPNMTVDISPINFO，然后填充其中item.pszText。但是我们通过什么来知道该结点所对应的信息呢，我的做法是在添加结点后设置其lParam参数，然后在提供信息时利用该参数来查找所对应的信息。下面的代码说明了这种方法： char szOut[8][3]={"No.1","No.2","No.3"}. // 添加结点 HTREEITEM hItem = m_tree.InsertItem(LPSTR_TEXTCALLBACK,...)

m_tree.SetItemData(hItem, 0). hItem = m_tree.InsertItem(LPSTR_TEXTCALLBACK,...) m_tree.SetItemData(hItem, 1). //处理消息 void CParentWnd::OnGetDispInfoTree(NMHDR* pNMHDR, LRESULT* pResult) { TV_DISPINFO* pTVDI = (TV_DISPINFO*)pNMHDR. pTVDI->item.pszText=szOut[pTVDI->item.lParam].//通过lParam得到 需要显示的字符在数组中的位置 *pResult = 0. } 关于编辑结点的显示字符：首先需要设置树形控件的TVS_EDITLABELS风格，在开始编辑时该控件将会发送TVN_BEGINLABELEDIT，你可以通过在处理函数中返回TRUE来取消接下来的编辑，在编辑完成后会发送TVN_ENDLABELEDIT，在处理该消息时需要将参数pNMHDR转换为LPNMTVDISPINFO，然后通过其中的item.pszText得到编辑后的字符，并重置显示字符。如果编辑在中途中取消该变量为NULL。下面的代码说明如何处理这些消息： //处理消息 TVN_BEGINLABELEDIT void CParentWnd::OnBeginEditTree(NMHDR* pNMHDR, LRESULT* pResult) { TV_DISPINFO* pTVDI = (TV_DISPINFO*)pNMHDR. if(pTVDI->item.lParam==0).//判断是否取消该操作 *pResult = 1. else *pResult = 0. } //处理消息 TVN_BEGINLABELEDIT void CParentWnd::OnBeginEditTree(NMHDR* pNMHDR, LRESULT* pResult) { TV_DISPINFO* pTVDI = (TV_DISPINFO*)pNMHDR. if(pTVDI->item.pszText==NULL).//判断是否已经取消取消编

辑 m_tree.SetItemText(pTVDI->item.hItem,pTVDI->pszText).// 重置显示字符 *pResult = 0. } 上面讲述的方法所进行的消息映射必须在父窗口中进行（同样WM_NOTIFY的所有消息都需要在父窗口中处理）。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com