Tree view check boxes: The extended check box states

devblogs.microsoft.com/oldnewthing/20171205-00

December 5, 2017

Raymond Chen

Version 6 of the common controls in Windows Vista introduced some new check-box-related extended styles for the tree view controls. Unfortunately, the documentation for them is kind of spare.

> `TVS_EX_PARTIALCHECKBOXES`
> Include partial checkbox state if the control has the `TVS_CHECKBOXES` style.
>
> `TVS_EX_DIMMEDCHECKBOXES`
> Include dimmed checkbox state if the control has the `TVS_CHECKBOXES` style.
>
> `TVS_EX_EXCLUSIONCHECKBOXES`
> Include exclusion checkbox state if the control has the `TVS_CHECKBOXES` style.

Yeah, that doesn't really explain anything.

Fortunately, more information about what these check box states are for can be found in the documentation for the `NSTCSTYLE` enumeration.

> `NSTCS_PARTIALCHECKBOXES`
> Adds a checkbox icon on the leftmost side of a given item with a square in the center, that indicates that the node is partially selected. Maps to the `TVS_EX_PARTIALCHECKBOXES` tree view control style.
>
> `NSTCS_DIMMEDCHECKBOXES`
> Adds a checkbox icon on the leftmost side of a given item that contains an icon of a dimmed check mark, that indicates that a node is selected because its parent is selected. Maps to the `TVS_EX_DIMMEDCHECKBOXES` tree view control style.
>
> `NSTCS_EXCLUSIONCHECKBOXES`
> Adds a checkbox icon on the leftmost side of a given item that contains a red **X**, which indicates that the item is excluded from the current selection. Without this exclusion icon, selection of a parent item includes selection of its child items. Maps to the `TVS_EX_EXCLUSIONCHECKBOXES` tree view control style.

Okay, so that explains what the intended purposes of these new styles are.

Of course, when you use those state images, you can use them to mean whatever you like. Though for consistency with the rest of Windows, you probably want to use them to mean what Windows uses them to mean, just like you should probably use the checked state to mean, y'know, that the thing is selected.

Don't forget that these are tree view extended styles, not window manager extended styles, so you set them by using the `TVM_SETEXTENDEDSTYLE` message or the corresponding `TreeView_SetExtendedStyle` macro.

The documentation for these extended styles says that they must be combined with `TVS_CHECKBOXES`, but that is not true; these extended styles imply `TVS_CHECKBOXES`; you don't need to set `TVS_CHECKBOXES`. In fact, it's worse than that. If you set `TVS_CHECKBOXES` first, and then set the extended styles second, you won't get the extended styles at all. That's because of the rules we spelled out last time:

> - Turn on the check boxes, either by setting the `TVS_CHECKBOXES` style (if all you want is unchecked and checked) or setting one or more of the `TVS_EX_XXXCHECKBOXES` styles (if you want other states, too).
> - Do not touch any of the checkbox-related styles any more. You get one chance, and that's it.

Okay, so that's really a documentation error, not a quirk.

Anyway, let's take these new extended styles for a spin. Save the scratch program we've been using up until now, because we're going to be reusing functions from it. Grab a new scratch program and make these changes:

```
#pragma comment(linker,"\"/manifestdependency:type='win32' \
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \
processorArchitecture='*' publicKeyToken='6595b64144ccf1df' language='*'\"")

BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  // Copy this from the old program.
    g_hwndChild = CreateWindow(WC_TREEVIEW,
    nullptr,
    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
    WS_CHILD | WS_VISIBLE,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    hwnd, nullptr, g_hinst, 0);

  // New code
  DWORD desiredStyles = TVS_EX_PARTIALCHECKBOXES |
                        TVS_EX_DIMMEDCHECKBOXES  |
                        TVS_EX_EXCLUSIONCHECKBOXES;

  TreeView_SetExtendedStyle(g_hwndChild,
        desiredStyles, desiredStyles);

  // Copy this from the old program
  PopulateTreeView(g_hwndChild);

  return TRUE;
}

void
OnDestroy(HWND hwnd)
{
  // Copy this from the old program
  ImageList_Destroy(TreeView_SetImageList(
    g_hwndChild, nullptr, TVSIL_STATE));
  PostQuitMessage(0);
}
```

The new image list states are added in the order above: Partial, then dimmed, then exclusion. If you omit one of the states, then the others move up to close the gap. For example, if you ask for partial and exclusion (but not dimmed), then the images are unchecked (1), checked (2), partial (3), and exclusion (4).

Raymond Chen

Follow