Tree view check boxes: A sordid history

devblogs.microsoft.com/oldnewthing/20171204-00

December 4, 2017

Raymond Chen

Tree view check boxes were not part of the original design of tree view controls, so people who needed check boxes rolled their own. At some point, somebody decided to do people a favor and move that code into the tree view, but did so in a way that preserved the way an external consumer would add check boxes, rather than integrating it with the tree view. I don't know whether this was due to simplicity of implementation (just taking existing code and moving it into the tree view control), or in order to preserve maximum compatibility with the existing code base of people using the external helper library (so you can tell people, "Just set this style and everything works the same as the library you are already using"). Or most likely both.

One part of the craziness enters the picture because the tree view needs to know how big to make the check boxes. If you have images associated with each item (which the tree view control calls the "normal image list"), then the check boxes should match the size of the images. But if you don't have any images, then the check boxes should match the system small icon size. The tree view cannot see into the future to know whether you are ever going to set the normal image list, so you need to give the tree view a signal to say "Okay, you can create the check boxes now."

Another part of the craziness is the fact that an item with a state image index of zero gets no state image. Since zero is the default state image index, it would mean that by default, even if you enable check boxes, you won't get any check boxes. Which is kind of unexpected. Therefore, the tree view control, as a courtesy, sets all the state image indices to 1 when it enables check boxes, thereby setting all the items into the unchecked state, which is a much more reasonable default than "no check box at all."

Given that the check boxes need to be created after the normal image list is set, it doesn't make sense to set the style at control creation time, because the control can't actually create the check boxes at creation time since you clearly haven't satisfied the prerequisites yet.

In retrospect, the use of a window style to trigger the creation of check boxes was a bad idea. It probably should have been a message like `TVM_ ACTIVATECHECKBOXES`. That makes it clearer that you need to prepare the tree view control, and then send the message to tell the control, “Okay, I’m all set. Go create the check boxes.”

But it’s a style, and we have to live with that. The tree view control responds to *changes* in the `TVS_ CHECKBOXES` style to initialize check box operations, because that’s how things worked manually: When you manually created a tree view with check boxes, you created the tree view, and then added the check boxes.

That’s why you have to set the style programmatically rather than doing so in the dialog template. If you set it in the dialog template, then the style is set at creation and no change is ever observed. If you were manually creating the tree view with check boxes, you had to add the check boxes later, and that’s how the code was migrated into the tree view control.

And the act of showing the tree view control for the first time causes all the check boxes to reset because the rendering code has a special test that says, “If the `TVS_ CHECKBOXES` style is set, and I never observed it being set, then go and initialize the check boxes right now, before it’s too late!” And one of steps in initializing the check boxes is setting all of the check box states get to unchecked. If it didn’t do that, then there wouldn’t be any check boxes at all, because the default state image for a tree view item is “blank”.

But this means that when you add an item to the tree view, and you say that you don’t want a state image, the tree view will say, “You said you wanted check boxes, but now you’re adding an item with no check box at all. I’m going to assume you simply forgot to specify that you wanted an unchecked check box, so I’ll give you an unchecked check box.”

If you really did want to add an item with no check box at all, then you need to add it normally (whereupon it will be given an unchecked check box), and then clear the state image.

```
// First add the item normally.
// The tree view control will assign the unchecked check box
// state if you didn't specify a state or specified a state
// of zero.
HTREEITEM hti = TreeView_InsertItem(hwndTV, &tvis);

// Then explicitly remove the state image.
TreeView_SetItemState(hwndTV, hti,
                      INDEXTOSTATEIMAGEMASK(0),
                      TVIS_STATEIMAGEMASK);
```

There’s another quirk of the `TVS_ CHECKBOXES` style: In versions prior to Windows Vista, the check boxes are created when the normal image list changes from null to non-null. And if you change it twice, then it creates the check boxes twice (and leaks the first one).

This leak was fixed in Windows Vista in version 6 of the common controls. From Windows Vista onward, if you use version 6 of the common controls, the check boxes will not be recreated if you set the normal image list more than once.

Given all these crazy behaviors, the best way to set up a tree view with check boxes is to do the following, in order:

- Create the control without the `TVS_ CHECKBOXES` style.
- Set the normal image list, if you want one.
- 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.
- If in the future you need to change the normal image list, change it from a non-null image list to another non-null image list. Do not change it to null, and then change it to non-null. This avoids the leak mentioned above.
- Add your items.
- If you need to add an item with no check box at all, first add it normally (whereupon it gets an empty check box assigned automatically), and then remove the check box manually, as noted above.

Wait, what are these `TVS_ EX_ XXXCHECKBOXES` extended styles? We'll look at those next time.

Raymond Chen

Follow