May, 2004

Advisor Answers

Disabling items in a combo box

VFP 8/7/6

Q: Is it possible to disable some items in a combo box?

A: As with so much in FoxPro, it depends. Specifically, some RowSourceTypes allow you to disable items while others do not. The basic rule is that you can disable items when the combo box "owns" the data, but not when it gets its data from another source. Also, note that the same rules apply to list boxes.

A little background on VFP's combo and list boxes is useful here. These controls are incredibly flexible in how they get populated. Two properties of a combo or list box, RowSourceType and RowSource, determine where the data comes from. As their names imply, the value of RowSourceType determines the interpretation of RowSource. Table 1 shows the list of RowSourceTypes and the corresponding interpretations of RowSource.

Table 1. Population combo and list boxes—The data in a combo or list box can be drawn from a variety of places. The RowSourceType and RowSource properties combine to determine what's included.

RowSourceType	RowSource
0-None	Should be empty. If anything is listed, it's treated as if RowSourceType=1.
1-Value	Contains a comma-separated list of items to put in the combo or list. If ColumnCount is greater than 1, the data is divided into columns.
2-Alias	Contains the alias for an open table. ColumnCount determines the number of listed fields displayed in the combo or list.
3-SQL Statement	Contains a query to be executed when the combo or list is created and each time it's requeried. The resulting cursor is treated as when RowSourceType=2.

4-Query	Contains the name of a query file (.QPR) to be executed when the combo or list is created and each time it's requeried. The resulting cursor is treated as when RowSourceType=2.
5-Array	Contains the name of an array from which the data is drawn. ColumnCount determines how many columns of the array are displayed.
6-Fields	Contains a comma-separated list of fields from which the data is drawn. (Include an alias only on the first field in the list.) ColumnCount determines how many of the listed fields are displayed. Field names only; expressions cannot be used here.
7-Files	Contains a file specification. Files in the current directory that match the specification are listed in the combo or list.
8-Structure	Contains the alias for an open table. Field names from that table populate the combo or list.
9-Popup	Contains the name of an existing popup (created with DEFINE POPUP). The bars from the popup are listed in the combo or list.

With many of the RowSourceTypes, the data comes from something outside the control and the combo or list doesn't have control over the data. However, with RowSourceTypes 0-None, 1-Value and 5-Array, you can think of the combo or list as owning the data. In those cases, you can disable individual items.

The mechanism for disabling an item is simple. Precede it with a backslash ("\"). For example:

```
RowSourceType = 1
RowSource = "\Men,Barney,Fred,Ricky,\Women,Betty,Lucy,Ethel"
```

There's one complication. Sometimes, you need a data item that begins with a single backslash. In that case, double the backslash—the item will show up enabled with a single backslash.

–Tamar

Positioning items in a listbox

VFP 8/7/6

Q: I have a listbox where the initially selected item is near the middle of the list. I'd like to have that item appear at the top of the list when the form opens.

A: This is one of those capabilities that most people seem to have overlooked. List boxes give you control over the positioning of items using the TopIndex or TopItemID property. Set either one and the item you specify is displayed at the top of the list.

How do you know whether to set TopIndex or TopItemID? The two properties control the same item, but they use two different numbering mechanisms. Both lists and combos support the two mechanisms.

The Index mechanism (represented by properties like List, ListIndex, and TopIndex and methods including AddItem and RemoveItem) is based on the position of the items in the list. The first item has index 1, the second has index 2, and so forth. Because items can be added to and removed from the list, the index for a particular item can change.

The ItemId mechanism (represented by properties like ListItem, ListItemId and TopItemId and methods including AddListtem and RemoveListItem) assigns every item in the list a unique ID. That ID never changes as long as the item remains in the list. When you use RowSourceType 0 and populate the list manually, you can assign the ID for a particular item.

In most cases, including setting the top item, it doesn't matter which mechanism you use to address the list or combo. The major difference between the two is in the behavior of the AddItem and AddListItem methods. Briefly, AddItem always adds a new row while AddListItem lets you add or change data in an existing row as well.

So, if you want to specify which item is at the top of the list based on its position in the list, set TopIndex. To specify based on the item's unique ID (which you might have set to the primary key of a record), use TopItemID.

There is one complication. TopIndex and TopItemID are read-only at design-time, so you have to set these properties in code. To

complicate matters, the list's Init method is too early—changes to these properties in Init are ignored.

To my surprise, even the form's Init method is too soon. You have to wait until at least the form's Activate method. Since you probably want to set TopIndex or TopItemID only when you first enter the form, that means you'll need a logical property to ensure that's the case:

```
* In Form.Activate
IF This.lFirstTime
   This.lstMyList.TopItem = This.lstMyList.Value
   This.lFirstTime = .F.
ENDIF
```

Finally, for a combo box, TopIndex and TopItemID are read-only at both design-time and runtime.

–Tamar