April, 1999

Advisor Answers

Updating combos and lists

Visual FoxPro 6.0, 5.0 and 3.0

Q: How can I update a listbox? I use a list to show a table in a multi-user application. If someone else adds a new line, I want to update the listbox by pressing a button. I tried putting the following code in the Valid of the button:

```
neu_count = COUNT()
ThisForm.List2.NumderOfElements = neu_count
ThisForm.List2.Refresh()
```

When I click the button, NumberOfElements changes, but the listbox doesn't shows the new item until I click on it. Is there a way to refresh the list without giving it focus?

-- Roland Klingler (via Advisor.COM)

A: The term "refresh" is definitely overused in Visual FoxPro. In particular, it's the name of a function that rereads data from the network and the name of a method that visually updates forms and controls.

The Refresh method of a control is generally called when the Value of the control has changed. However, Refresh doesn't refill listboxes and comboboxes when their RowSources change. That's the job of the Requery method (also an overloaded term in VFP). Requery goes to the RowSource of the list or combo, says, in effect, "give me everything you've got" and recreates the internal list of the control. (Regardless of the RowSourceType, combos and lists maintain their own internal representation of the items in the list. With many of RowSourceTypes, this internal list is not automatically updated when the underlying data changes.)

Don't confuse the Requery method with the same-named REQUERY() function. Its job is similar, but different. The REQUERY() function applies to views – it re-runs the original query on which the view is based (hence the name "re-query") and refills the view's cursor. With parameterized views, REQUERY() checks the current parameter value, so it's the key to gathering a different subset of the source data.

Back to lists. In most cases, all you need to refill a list or combo from a method of the list itself, is:

```
This.Requery()
```

If the call comes from a method outside the list, use something like:

```
ThisForm.lstMyList.Requery()
```

In some cases, you may need to do some work before the list is refilled. For example, if RowSourceType is 2-Alias, and RowSource is a cursor created with a SQL SELECT, you probably want to run the query again before you requery the list. Otherwise, nothing changes. Similarly, if RowSourceType is 5-Array and you built the array with a query INTO ARRAY or with a function like ADIR(), you may want to execute the query or function again.

In this situation, I like to take advantage of the fact that any custom code in the Requery method is executed before the list is refilled. So, I put the query (or whatever other code I need to execute to populate the RowSource itself) in the Requery method. Then, the form and its controls don't have to worry about the RowSourceType. Calling Requery is sufficient to refill both the RowSource and the list or combo.

For example, say we have a list called lstEmps with RowSourceType set to 2-Alias and a RowSource of MyEmps, a cursor based on the Employee table. The list's Requery might look like:

```
* Run the query to create MyEmps
SELECT LastName, First, Title ;
   FROM Employee ;
   WHERE StartDate >= ThisForm.dStartDate ;
   INTO CURSOR MyEmps
```

When lstEmps.Requery() is called, the query executes, then the list is refilled using the data now contained in MyEmps.

Views offer another situation where putting code in Requery is useful. You may have a combo or list displaying the data from a parameterized view, where the parameter is the ControlSource for another control on the form. When the other control is changed, you need to REQUERY() the view, and then refill the combo.

For example, consider the same scenario as above, except that instead of a cursor, we have a view for the employees of interest, defined as:

```
CREATE SQL VIEW vMyEmps AS ;
   SELECT LastName, First, Title ;
      FROM Employee ;
      WHERE StartDate >= ?dStartDate
```

Assume dStartDate is the ControlSource of another control (perhaps a textbox). In the Valid or LostFocus of that control, we can issue lstEmps.Requery. The code in Requery this time should be:

```
REQUERY("vMyEmps")
```

The REQUERY() function runs first, refilling the view's cursor. Then the list is refilled based on the data now in the cursor.

One final note. The NumberOfElements property isn't really intended for the kind of situation you describe. Its purpose is to provide backward-compatibility with the lists and combos from FoxPro 2.x. Together with FirstElement, it lets you specify that only a subset of the data in an array should appear in a list or combo rather than every element.

The mechanism used by FirstElement and NumberOfElements is a little complex and not worth going into here, except to say that it only lets you choose a portion of a single column. It does not allow you to specify that a rectangular sub-portion of an array should appear in the list or combo.

For most situations, Requery is the best way (often, the only way) to repopulate a list or combo.

-- Tamar