



'Just For You' Consulting Services, Inc.  
1941 Lake Ave.  
Scotch Plains, NJ 07076  
Ph: (908) 233-4447 Fax: (908) 233-8033

## **Pushing vs. pushing data into controls**

Imagine you want to provide a combo box from which you can choose a value. Once a value has been chosen you want to fill in various other controls on the form with data found in the row that corresponds to the value the user just chose. The methods for doing this differ, depending on whether those other fields are bound. If the other controls are bound, you must push data into them; otherwise, they can pull the new data in themselves. You might think of the pull methods as being passive, since the data just flows in, and the push method as being active, since you must provide code to copy the data from the list or combo box.

## **Pulling data into unbound controls**

If the controls to be filled in are unbound, it means their ControlSource Property is otherwise empty, and you can use an expression to pull in data from the combo box where the user just made a choice. All references in this example work equally well for list boxes and combo boxes.

To pull in data from the combo box, follow these steps:

1. Fill the comb: Create a query (or use an existing table) that contains the data you want to present, plus any other fields you want filled in automatically once the user makes a choice. Later steps will be simpler if you make sure the value to be displayed in the combo is the first fields in the table or query, but that isn't imperative.
2. Prepare the combo: set the ColumnWidths property so that the correct column is visible and the other columns are invisible. If the first column is the one you want displayed and you have five columns total, your ColumnWidths would be ;0;0;0;0. This tells access to use the default width for the first column and 0 width (hidden) for the next four columns.
3. Prepare the other controls: Set the ControlSource property for each of the controls into which you want data pulled. In each case, set the Controlsource Property to =YourCombo.Column(n).

Where YourCombo is the ControlName Property of the combo and n is the column number (starting at 0) you want pulled in the control.

Once you've set up the form, Access takes care of the rest. Anytime the combo box changes, the other controls on the form will recalculate and



'Just For You' Consulting Services, Inc.

1941 Lake Ave.

Scotch Plains, NJ 07076

Ph: (908) 233-4447 Fax: (908) 233-8033

pull the value they need from the combo box. If you're pulling information from a list box, the information will be updated each time you move the selection bar in the list box. This is a convenient way for users to browse items as they move through the list box using the arrow keys. It takes absolutely no programming to accomplish a great deal!

## **Pushing Data into Bound Controls**

If you need to fill in controls that are bound, it means their ControlSource property is not empty, and you can't use the pull method described in the previous section. Instead, you'll need to push data into the controls once you've made a choice from the combo or list box. This is also simple, but it requires a bit of code.

The steps you follow to implement this method are the same as they were for the pull method. In this case, though you need to leave the ControlSource property of the text boxes alone. The assumption is that you're using this method because those controls are bound to data fields. To apply the push method, you attach code to the AfterUpdate event of the combo or list box, which will fill the appropriate controls.

The code is as follows for the AfterUpdate event.

```
Me("textbox")=YourCombo.Columns(n)
```

"textbox" = Bound Control Name

YourCombo=Combo Box Name

N=Column of Combo Box starting at 0

***This is copied from pages 395-396 – Access 2000 Developers Handbook  
Volume 1:Desktop Edition***

***Ken Getz, Paul Litwin, Mike Gilbert***