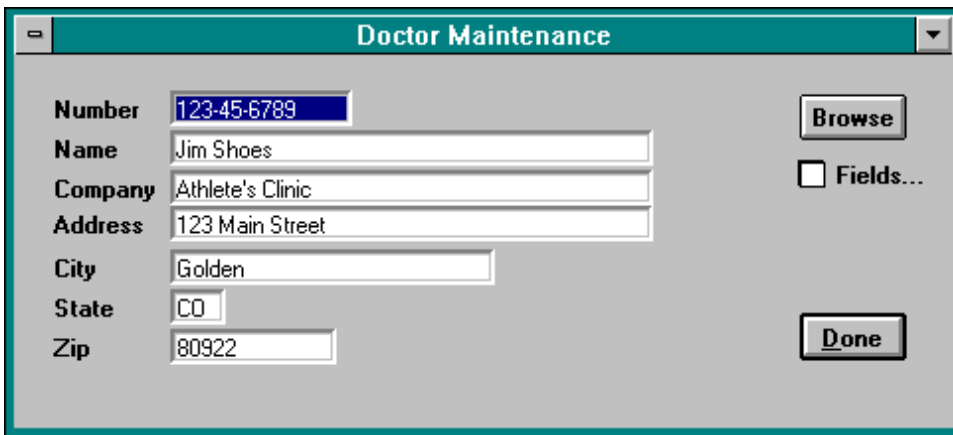


The Fox Hunt

By Whil Hentzen

Last month we discussed the use of a data driven Browse window from a data entry and maintenance screen. The screen had a Browse pushbutton that was used to display a Browse window through which the user could navigate from record to record. The trick was to populate the field list in the BROWSE command from a table that contained all of the fields in the table. This month, we're going to enhance this functionality by including the ability for the user to select which fields they want to appear in the Browse window.

We're going to use two mechanisms and a trick to do this. The first mechanism is a "Fields" checkbox on the maintenance screen, placed in close proximity to the Browse pushbutton. If the user just presses the Browse pushbutton, a standard Browse window will be brought forward, showing all of the columns in the table. However, if the user checks the Fields checkbox and then presses the Browse pushbutton, a "mover" dialog will be brought forward that allows them to select which fields will appear in the Browse. This is our second mechanism. The mover dialog consists of two listboxes, side by side. The left side listbox contains all of the available choices and the right side listbox contains all of the selected choices. The user can move an item from one side to the other by using one of the pushbuttons in the mover dialog or just by double-clicking on the element in the listbox itself.

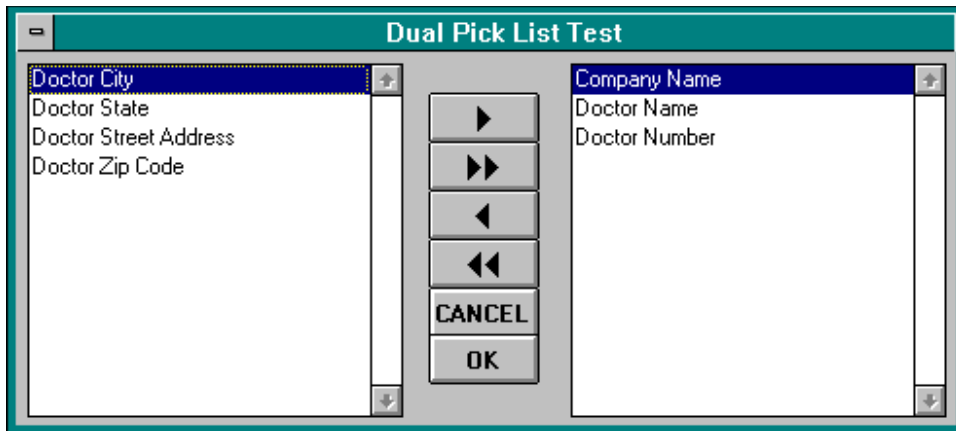


The screenshot shows a window titled "Doctor Maintenance" with a teal header bar. On the left, there is a form with the following fields: "Number" (123-45-6789), "Name" (Jim Shoes), "Company" (Athlete's Clinic), "Address" (123 Main Street), "City" (Golden), "State" (CO), and "Zip" (80922). On the right side of the form, there is a "Browse" button, a checkbox labeled "Fields...", and a "Done" button.

And the trick? The trick is that we're not going to write the mover dialog ourselves. Instead, we're going to avail ourselves of the wide variety of good utilities that are available on CompuServe's FoxForum. The one we're going to use is called DUALPICK and was written by Jeff Neeley early this year. He spent a couple of days putting this dialog together and then released it to the public for free. Why reinvent the wheel if you don't have to?

Let's see what we have to do in order to change the code we used last month. (All of the source code and design surfaces, including DUALPICK, are included on this month's source code diskette.)

First, we have to detect whether the user pressed the Fields checkbox, and if they did so, we need to bring forth the mover. If they didn't, of course, we could just use the code from last month to populate the BROWSE command. This code belongs to the Valid snippet of the Browse pushbutton. The first few lines are the same as we saw last month - they determine which fields in SYSCOL belong to this table and thus are available for browsing.



There's one subtle assumption here. What we've done is create a second array that only holds the English names of the fields in question. It wouldn't be too friendly to show a listbox with items like "CNODO" and "CNAORG" in it, would it?

DUALPICK uses two arrays, lArray and rArray, to populate the left and right listboxes. We fill lArray from the result of the SELECT command, and then call DUALPICK. For the purposes of this article, we're going to assume that DUALPICK is a black box - we send it some data and expect some information in return. In this case, we are expecting an array, rArray, when DUALPICK is finished.

Now that we have our array of selected values, it's time to build the BROWSE FIELDS string. And it's time to pay attention closely! The values in rArray are simply the English names of the fields - not the column names that we'd need. What we'll have to do is find the English name in our original array, a2PossFields, and then determine what the associated column name is. The ASCAN function does the trick nicely.

Note that we still build the BROWSE FIELDS string, together with column headings from the a2PossFields array's English names column. And the rest of the code - creating the Browse window and displaying the records - is the same as last month. We could make this more robust. For example, what if the user doesn't pick any items in the mover? We'd probably want to trap for that situation - and either return to the data entry screen or browse all fields by default.

Furthermore, the use of a third party tool is not without its problems. First, there might be aesthetic things, like different fonts, sizes, colors and other physical attributes, from the one you use in the rest of your app. Second, you might have noticed that DUALPICK automatically alphabetizes the elements in the selected listbox. Kinda shoots the use of our COLUMNPOS field in SYSCOL in the foot, doesn't it? But that's why I selected this particular utility for use with this article - DUALPICK comes with all of the source code and design surfaces, so if you wanted, you could enhance it as you saw fit. The advantage is having a working tool that does most of what you need without spending the time building it from scratch.