May, 1998

Advisor Answers

Visual FoxPro 5.0 and 3.0

Q: I am trying to use the command:

```
oObj=CREATEOBJECT("frmObject",aArray)
```

In the Init method of frmObject I have:

```
LPARAMETERS aIndex
```

But aIndex is not an Array. What am I missing? Using PUBLIC aArray[10] and CREATEOBJECT() without parameters, it works.

–Eurico Chagas-Filho (via Advisor.Com)

A: This is one of those cases where the overall complexity of the problem is hiding a minor mistake. In FoxPro, arrays must be passed by reference. Passing an array by value actually passes only the first element of the array.

Since CREATEOBJECT() is a function call and, by default, function calls pass parameters by value, you have to do something special to make this work.

There are two possibilities. The one I prefer is to simply precede the array name with "@", indicating this it's to be passed by reference. In that case, your CREATEOBJECT line would look like:

```
oObj = CREATEOBJECT("frmObject",@aArray)
```

The other alternative is to SET UDFPARMS TO REFERENCE. This tells VFP to pass all parameters to functions by reference, and this is exactly why I don't like it. This solution is too blunt and too likely to cause problems elsewhere in your application.

While we're on the subject of arrays, parameters and objects, it's worth noting that you can't pass array properties as parameters at all. That is, if you've added a property to an object and defined it as an array, say aMyProp[5], you can't pass it as a parameter to another method or a function like this:

```
THIS.MyMethod(@THIS.aMyProp)
```

or

```
SomeFn(@THIS.aMyProp)
```

That gives an error. Instead, you have to copy the array to a local array and pass that. If the method or function changes the array and you need to store the changes, you have to copy the results back into the property. The sequence looks like this:

```
LOCAL aHold[1]

ACOPY(THIS.aMyProp,aHold)
```

```
THIS.MyMethod(@aHold)
ACOPY(aHold,THIS.aMyProp)
```

You should be aware that, in general, object properties can't be passed by reference. I suspect this prohibition is for philosophical reasons based on the idea that no outside object should be able to reach into another object in such an indirect way.

If you need the effect of passing by reference, the solution is the same as with arrays. Copy the property to a local variable, pass the variable by reference and recopy the value to the property afterward. However, when you find you want to pass a property by reference, it's worth thinking about whether that's a sign of something not quite right in your design.

–Tamar