Include Browses in Regular READ Windows - Painlessly - with InBrowse!

The purpose of Ken Levy’s GENSCRNX pre and post-processor for FoxPro’s GENSCRN was to enable the developer to include features in screens that couldn’t be done without irreparably modifying the .SPR by hand. A side effect of GENSCRNX was to enable the developer to do things that could be done in other ways, but to do them much with much less effort. This month’s cool tool, a driver for GENSCRNX that enables you to embed true Browse windows in standard READ windows, satisfied both descriptions.

Simply put, a Browse can’t be included in the READ of a window - it’s not a standard GET like data entry fields or radio buttons. You can create an effect that gets close, but it takes a lot of waving of hands and crossing chicken bones, and the results are never all that satisfying. With Christophe Biehlmann’s InBrowse driver, however, you can include one or more Browses in a window along with standard GETs and any other of the usual window objects, and have the entire package integrated like you’ve wished. And it’s not difficult.

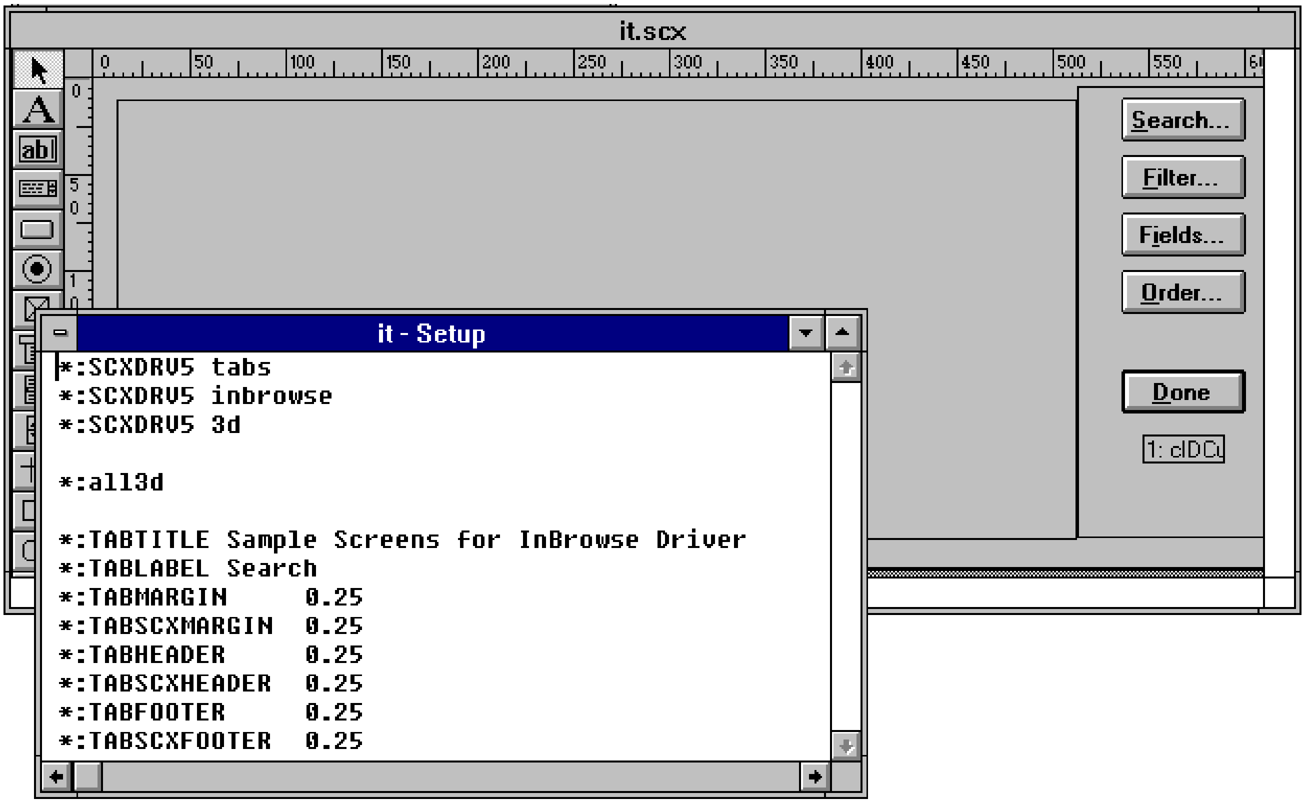
Not to belabor the entire explanation of how GENSCRNX works again (you can find a detailed explanation in the ____ and ______ issues of FoxTalk), but for new readers, the essence is this: The FoxPro _GENSCRN directive is pointed to GENSCRNX.PRG instead of FoxPro’s standard GENSCRN.PRG. GENSCRNX acts as a “WHEN” and a “VALID” for GENSCRN, modifying the .SCX table before GENSCRN is called, calling GENSCRN, and then modifying the resultant .SPR file. At certain points while running, GENSCRNX looks for the existence of other programs, and executes them (just like subroutines) if it finds them. These programs are called drivers. The InBrowse driver is simply a .PRG that is called by GENSCRNX at a specific point in the generation cycle.

To Use InBrowse

Using InBrowse is extremely easy. Christophe has used Steven Black’s TABS driver to produce an sample application that demonstrates several different variations of InBrowse in one screen. I’ve also used TABS to show how InBrowse might be used to clean up a complex or busy maintenance screen by putting the Browse in one tab and the editing fields in another.

There are three steps to using InBrowse. The first is to tell FoxPro to run GENSCRNX and to put the GENSCRNX drivers somewhere so that they can be found. Christophe puts INBROWSE.PRG in the FoxPro directory while I place it (along with any other driver, such as TABS or 3D) in a special directory called DEVUTILS that is included in the FoxPro path upon startup.

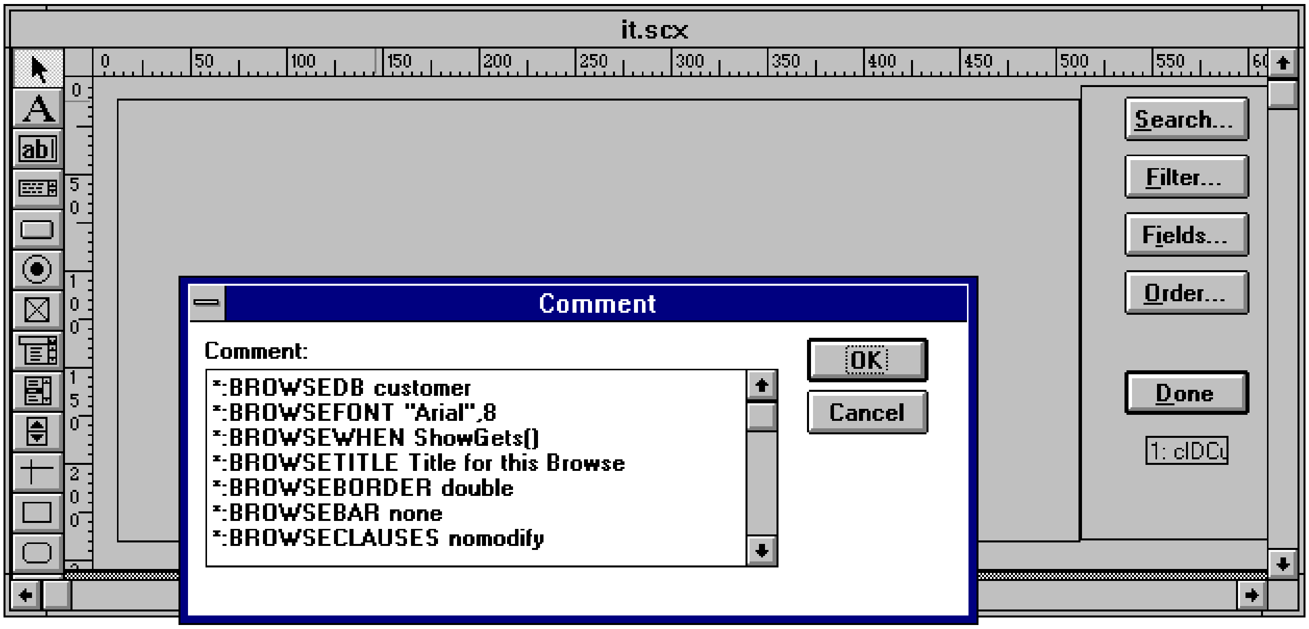
Next, you need to direct the screen set to look for the InBrowse driver when being generated. You can do this by either including the InBrowse directives in your CONFIG.FP(W) file, or simply placing them in the Setup code of the screen in which you want to place a Browse window. Both Christophe and I have chosen the latter so that the samples can be run “as is” without you having to move things around in your system.



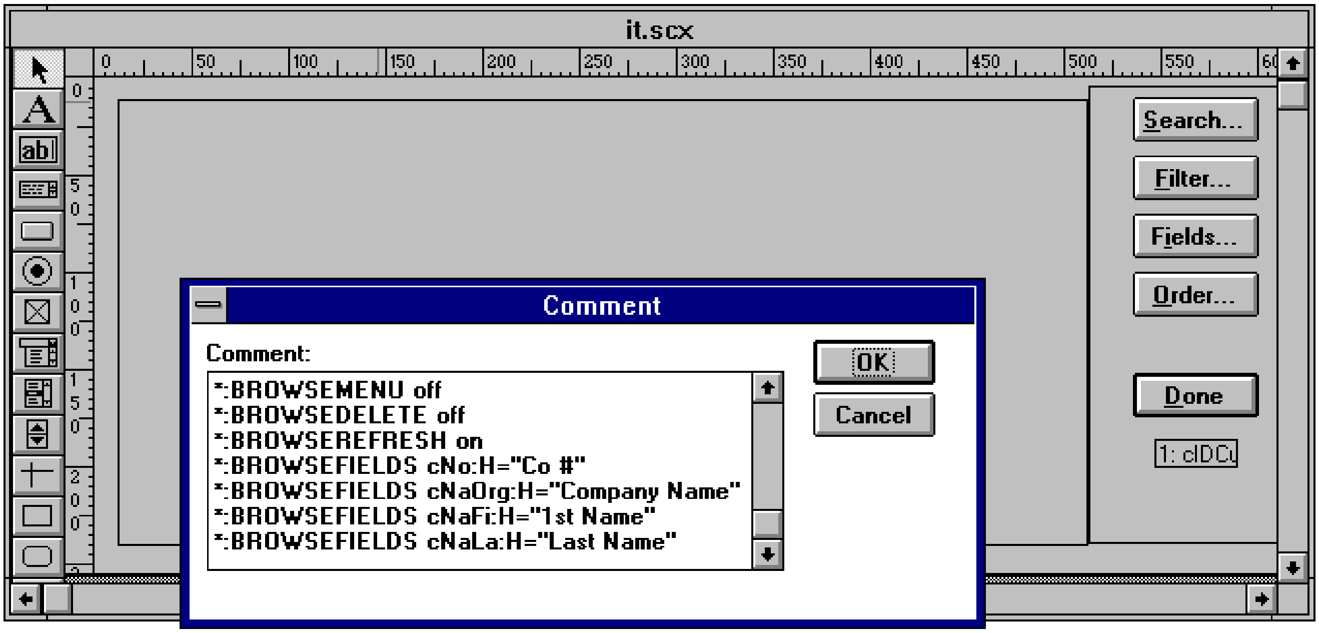
```
*:SCXDRV5 tabs
*:SCXDRV5 inbrowse
*:SCXDRV5 3d

*:all3d

*:TABTITLE Sample Screens for InBrowse Driver
*:TABLABEL Search
*:TABMARGIN     0.25
*:TABSCXMARGIN  0.25
*:TABHEADER     0.25
*:TABSCXHEADER  0.25
*:TABFOOTER     0.25
*:TABSCXFOOTER  0.25
```

The Setup snippet of the main screen only contains a few directives. Note the relative position of the TABS, INBROWSE and 3D directives.]

The third step is to draw a plain box on the screen that will hold the Browse window. This box contains the specific directives that control how the Browse is created and what functionality it has. The two screen shots show some of the comments I've used to create the sample application.



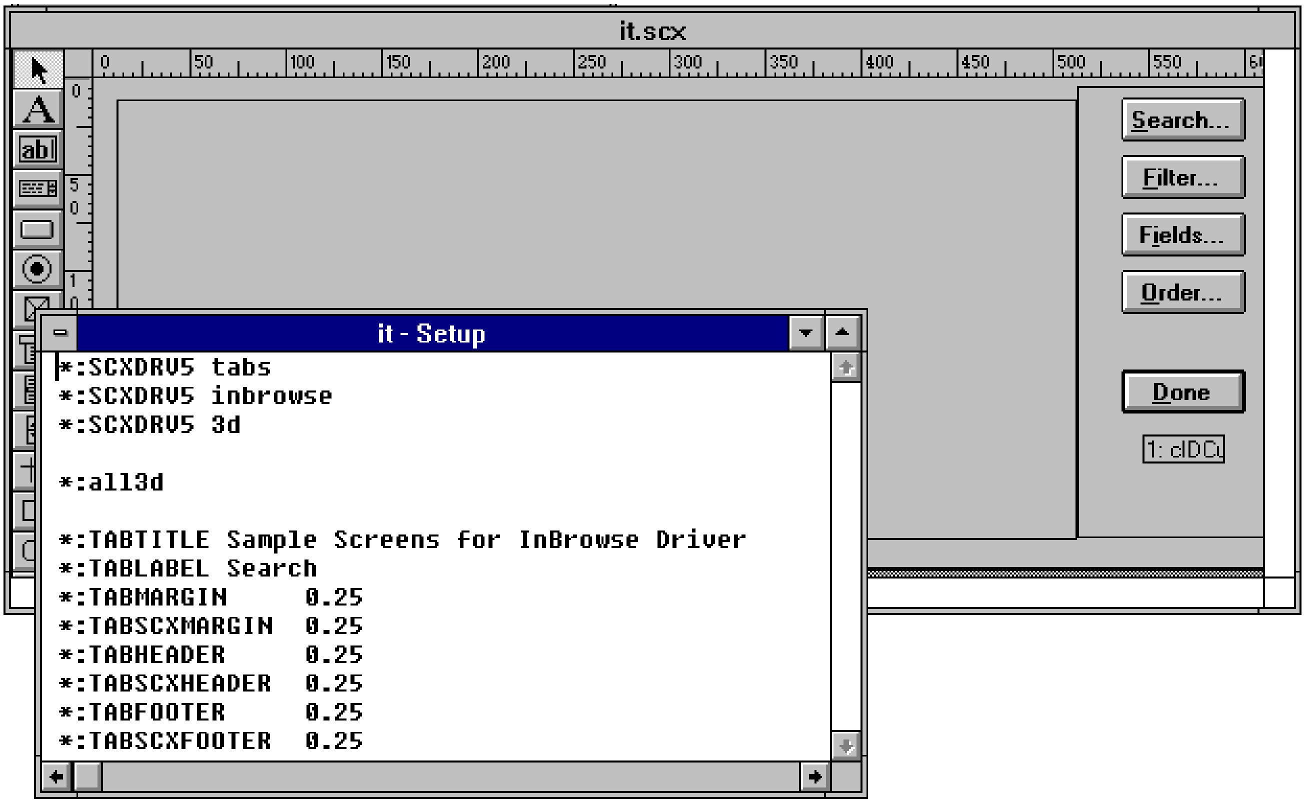
```
*:BROWSEDB customer
*:BROWSEFONT "Arial",8
*:BROWSEWHEN ShowGets()
*:BROWSETITLE Title for this Browse
*:BROWSEBORDER double
*:BROWSEBAR none
*:BROWSECLAUSES nomodify
```



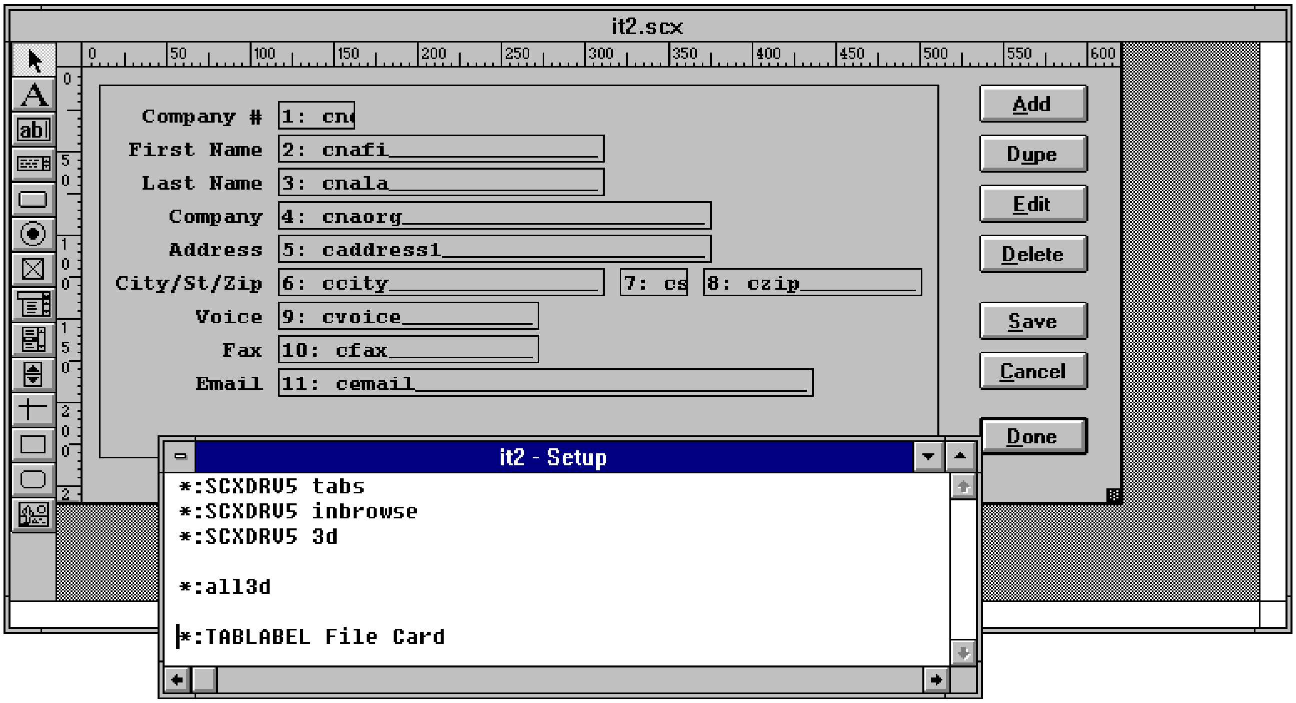
```
*:BROWSEMENU off
*:BROWSEDELETE off
*:BROWSEREFRESH on
*:BROWSEFIELDS cNo:H="Co #"
*:BROWSEFIELDS cNaOrg:H="Company Name"
*:BROWSEFIELDS cNaFi:H="1st Name"
*:BROWSEFIELDS cNaLa:H="Last Name"
```

(you may want to trim them down) The Comment snippet of the Box drawing object that will hold the Browse contains the InBrowse directives.]

Since this sample has two tabs, the second screen in the set also requires directives in the Setup snippet in order that TABS can do it’s thing.



```
*:SCXDRV5 tabs
*:SCXDRV5 inbrowse
*:SCXDRV5 3d

*:all3d

*:TABTITLE Sample Screens for InBrowse Driver
*:TABLABEL Search
*:TABMARGIN     0.25
*:TABSCXMARGIN  0.25
*:TABHEADER     0.25
*:TABSCXHEADER  0.25
*:TABFOOTER     0.25
*:TABSCXFOOTER  0.25
```



```
    Company #  1: cn
   First Name  2: cnafi
    Last Name  3: cnala
      Company  4: cnaorg
      Address  5: caddress1
  City/St/Zip  6: ccity          7: cs  8: czip
        Voice  9: cvoice
          Fax  10: cfax
        Email  11: cemail
```

```
*:SCXDRV5 tabs
*:SCXDRV5 inbrowse
*:SCXDRV5 3d

*:all3d

*:TABLABEL File Card
```

The Setup snippet of the second screen contains TABS-specific directives.]

Then, create a project, add the main program, and generate the screens. If you're using TABS, you'll need to add additional tabs to the screen set. Running the generated program will result in the following two screens. You can maneuver through both tabs and from standard READ objects to the Browse and back either with the keyboard or the mouse. Note that the one compromise we've needed to make was to use CTRL-TAB to pop from the Browse to the other objects and back.

Options

The InBrowse directives are quite flexible. You can include any of the standard Browse clauses such as NOMODIFY or NOAPPEND with the *:BROWSECLAUSES directive, hide the delete column with the *:BROWSEDELETE OFF directive (handy if you have NODELETE also turned on), and specify which fields to display in the Browse with the *:BROWSEFIELDS directive. Additionally, the *:BROWSEWHEN clause enables you to perform a specific action when the Browse is entered and the *:BROWSEKEY and *:BROWSEFOR directives provide access to the standard Browse Key and Browse For clauses. And, of course, you can control the title of the Browse window, whether or not horizontal and vertical scroll bars display, and what the border around the edge of the Browse window looks like.

The most current version of InBrowse as of this writing, version 1.4, was released in late December, and we're still coming up the learning curve on it. I've just started working with it in order to create the type of interface presented by the example, and I'm still learning little tips and tricks.

For example, you must include a GET field in the screen that contains the Browse, and there must be a box around that GET with the *:BROWSEGET directive in the comment snippet. Since I didn't want a GET field in the Search tab of my screen, I used a GET field and then hid it from display with an *:IF directive. Then I prevented entry into it with a .F. expression in the WHEN of the GET and I was all set. (You can find the GET under the Done pushbutton in the Search tab.)

Furthermore, I've found that the *:BROWSEGET must surround any objects on the screen that you want access to when you're not in the Browse. I surrounded the hidden GET as well as all the pushbuttons. And since the Browse is essentially a navigation tool, I used a SCATTER MEMVAR as part of the WHEN directive to refresh the memory variables each time the Browse was exited and the user entered the File Card tab. It's a little crude, but it's a start.

InBrowse is another one of those gems of this FoxPro community - a public domain tool donated by Christophe. As a result, it's available on this month's Companion Disk as well as on FoxUser Forum on CompuServe (INBR14.ZIP), and support is available on FoxUser as well.