计算机数据库丛书

NEW:

FoxPro

编程经验与技巧

学苑出版社

1

FoxPro 编程经验与技巧

（一）

学 苑 出 版 社

1994

(京)新登字 151 号

内 容 简 介

本系列丛书是 FoxPro 开发人员在多年的数据库开发中经验的结晶，它提供了大量高级、新颖的 FoxPro 开发技巧。通过阅读本书，可以大幅度地提高您的 FoxPro 编程能力。

欲购本书者，请与北京 8721 信箱联系，邮政编码 100080，电话 2562329。

计算机数据库丛书

FoxPro 编程经验与技巧(一)

编　　著：John L. Hawkins 等
翻　　译：亦　鸥　　木　杉
审　　校：希　望
责任编辑：徐建军
出版发行：学苑出版社　　**邮政编码**：100032
社　　址：北京市西城区成方街 33 号
印　　刷：兰空印刷厂
开　　本：787×1092　1/16
印　　张：6　　**字　　数**：138 千字
印　　数：1～5000 册
版　　次：1994 年 4 月北京第 1 版第 1 次
ISBN7－5077－0886－1/TP・28
本册定价：8.00 元

目　录

两个新的“ISX”函数

Malcolm C. Rubel

本月我们来共同讨论两个有趣的“ISX”函数，一个用于 FoxPro for Windows，另一个用于两种版本。希望这些讨论能对读者学习 FoxPro 编程有帮助。

字模按比例分隔了吗

在 FoxPro for Windows 中，使用按比例分隔的字模比使用单空格分隔模速度慢，而且更为复杂。我们应该尽可能地避免特别的处理。因此，发现自己是否在使用按比例分隔的字模是很重要的。通常仅通过观察屏幕输出来辨认出字模是否被按比例分隔，这样在程序中需要作些改动以能够辨认这种情况，而不需程序员的帮助。

逻辑上的起点在 FoxPro 语言参考手册或屏幕帮助行中。我们感兴趣的是字模的特征，因此最相似的入选者是 FONTMETRIC()函数。FONTMETRIC (6)返回指定以象素来计量的字模的平均字符宽度。FONTMETRIC (7)返回以象素来计量的最大的字符宽度。这样，只需比较两个返回值。若二者相同，表明字模是单分隔的；若返回值不同，表明字模是按比例分隔的。我编写的 ISPROPOR()函数的核心代码如下：

```
RETURN(FONTMETRIC (6,<font _ name>,10)#;
       FONTMETRIC(7, <font _ name>,10))
```

把这段代码写入用户自定义函数(UDF)时，若字模按比例分隔则返回 TRUE；若字模是单空格分隔或不存在，函数返回 FALSE。效果是很好的。

看起来我们已经得到了所需要的函数，它符合要求。我们提出这样问题，“是否有其他更合理的途径来编写这个函数?”我思索寻找了很长时间，相信会找到这个函数。偶然的机会我想到了另一个方案，决定编写第二个函数并进行尝试。

FoxPro for DOS 中，TXTWIDTH()和 LEN()函数通常返回相同的结果，运行速度也几乎相同。但在 FoxPro for Windows 中却存在着明显的差异。TXTWIDTH()的运行速度几乎比 LEN()函数慢 8 倍，即使字模是单空格分隔时情况仍是如此。我尝试着用 TXTWIDTH ()函数来重新编写我自己的 ISPROPOR() 函数，意在能显示出使用这种方式慢了多少。函数代码如下：

```
RETURN(TXTWIDTH('i' <font _ name>)#;
       TXTWIDTH('W',<font _ name>)
```

我推测小写“i”的宽度和大写“W”的宽度象其他任何两个字符那样分隔。若宽度不同，按比例分隔性的答案就显而易见了。

测试两种版本的函数时，存在明显的速度差，但这并不在我所预测之中。使用

TXTWIDTH()的函数版本比使用 FONTWIDTH()的版本要快 30 倍左右。这表明 FONTWIDTH()函数是很慢的。我改写了函数,下面给出了 ISPROPOR()函数的最终版本:

```
FUNCTION ispropor
PARAMETERS pt _ font
PRIVATE pt _ check,pt _ rvalue

*   ISPROPOR ([<C font>])
*   Note:Function returns TRUE if the passed font
*   is proportionally spaced.

* * *   if no parameter use the current window font
m. pt _ font=IIF (PARAMETERS()#1,;
                  WFONT(1),m. pt _ font)

* * *   if parameter was passed as a null string,get
* * *   the system window font.
m. pt _ font =IIF(EMPTY(m. pt _ font),WFONT(1,''),;
                  m. pt _ font)
m. pt _ rvalue=. f.

IF TYPE('m. pt _ font') = 'C'AND _ windows

m. pt _ rvalue=TXTWIDTH('i',m. pt _ font,1)# 1 OR;
                        TXTWIDTH('W',m. pt _ font,1)#1
ENDIF
RETURN(m. pt _ rvalue)
```

任何事情都是这样,只有付出努力才有可能成为幸运儿。得到这个答案表明 FoxPro 向我证实了这一点。

处理路径语句

下一个例子和路径及缺省目录有关。我希望函数在给定当前 SET PATH 和 SET DEFAULT 设置时,能够告诉用户输入到程序中的路径/目录语句是否使目录中的文件可见。遇到的这个问题是用户能给出一个相关的 PATH 语句,或在 PATH 语句中没有驱动器字母时在路径中能够包括驱动器字母。我编写了上百行程序试图在实际比较字符串之前得出结果,以发现目录是否在路径语句中。函数显得混乱一些,诚实地说,它有时并不起到作用。

我最终放弃了已编写的程序而又重新开始。另一种方案是使用 FCREATE()函数建立一个具有传递路径/目录的文件,要检验目录实际存在后使用 ISDIR()函数,这个函数几个月前我曾经提供过。文件建立后,就可以使用 FoxPro 的 FILE()函数来察看文件在 FoxPro 中是否可见。若可见,表明目录在路径中,这时返回值是 TRUE。清除程序在返回调用的程序前将修改文件。其优点在于:函数并不关心相关的路径和驱动器字母,而发生了它应有的作用。

其后我开始考虑试图在目录中建立文件的问题。应该知道建立和删除文件需要一系列

的 DOS 操作，运行是缓慢的。若是在网络上，需要书写关于那个目录的优先权吗？是否存在其他方法？

事实证明是存在另一个解决方法。可以使用 ADIR()函数来获取一个目录中已存在的文件，然后使用 FILE()。我对两种方法都进行了尝试，结果表明 ADIR()这种方法要快 2—4 倍。尽管如此，对此函数我仍不满意。若是目录存在且在 FoxPro 的路径语句中，而目录中不存在任何文件，可能会得到错误的负数返回值。应该采取什么手段来解决这一问题呢？

函数的最终版本是将两种方式结合起来。首先使用了 ADIR()来获得文件名；若 ADIR()返回零，表明没有文件，这时使用 FCREATE()来检验空的驱动器。这样函数既取得了 ADIR()的速度，又采取了 FCREATE()函数提供的安全措施。下面是函数代码：

```
ISVISIBL() Function
FUNCTION isvisibl
PARAMETERS pt_path
PRIVATE pt_fh,pt_array,pt_rvalue
*   ISVISIBL(<C path>)
*   Note: Program returns TRUE if directory and
*   path passed are "visible" to FoxPro
m.pt_rvalue=.f.
IF TYPE('m.pt_path')='C'
  m.pt_path=ALLTRIM(m.pt_path)
  *** clear up passed path
  IF RIGHT(m.pt_path,1)#'/'
    m.pt_path =m.pt_path+'/'
  ENDIF
  *** is directory a valid dos dir?
  IF ISDIR(m.pt_path)
    DECLARE pt_array[1]
    *** find a file, any file
    *** check for file using ADIR first
    IF ADIR (pt_array,m.pt_path+'*.*')#0
      m.pt_rvalue=FILE(pt_array[1])
    ELSE
      *** if no files in directory
      *** create a test file
      m.pt_fh=FCREATE(m.pt_path+'jjxxzztt.bzh')
      IF m.pt_fh>0 &&successful
        =FCLOSE(m.pt_fh)
        m.pt_rvalue=FILE('jjxxzztt.bzh')
        DELETE FILE(m.pt_path+'jjxxzztt.bzh')
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  RETURN (m.pt_rvalue)
```

函数简单、快捷，并且在任何情况下都能够运行。不知读者怎样看待？

结论

本期我们讨论的这两个函数还有大量的改进之处，我只希望这些例子能对读者起到抛砖引玉的作用。盼望与您共同探讨。

在屏幕构造器中增加程序段

Miriam Liskin

本期编程基础讨论之前，先设计了用户自定义函数(UDF)用于格式化数据。某种意义下，这些"输出的"UDFS 是可选的，它们允许用户生成外观更好的报告，但它们并不真正为 FoxPro 应用程序的正常功能所必需。

然而，一些编程知识对使用 FoxPro Screen Builder 是绝对必要的。一旦用户掌握了怎样向设计的屏幕增加代码，在解决一些问题时会游刃自如。用户可以超越简单的 BROWSE 和 EDIT 屏幕，而设计成用于输入和修改数据的更有用和具有吸引力的格式。

本文所举的例子都来自于 MAILLIST 表格的数据入口屏幕。FoxPro for DOS(FPD)版本的 MAILDOS 示于图 1，FoxPro for Windows(FPW)版本的 MAILWIN 示于图 2。

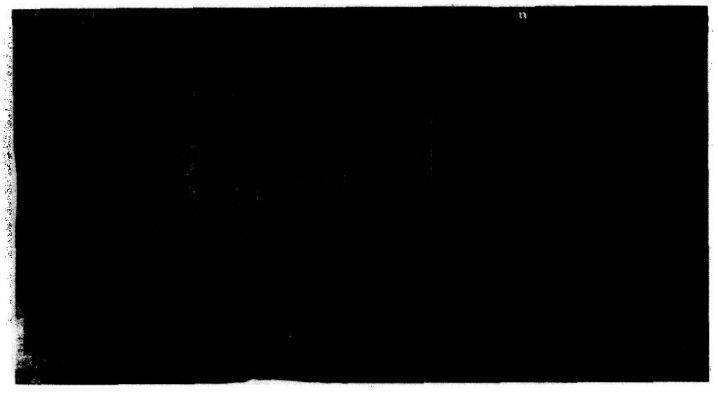

图 1　一个在 FoxPro for DOS Screen Builder 中构造的数据入口屏幕

屏幕构造器

若读者从未使用过屏幕构造器，可能需要一个简短的适应过程，使自己熟悉设计屏幕的基本步骤——指定窗口特征以按一定形状放置文本和数据区域。可以从与 FoxPro 一同提供的《怎样起动》和《用户手册》中找到有关细节。

与报告不同，FoxPro 并不直接执行屏幕。它首先读取用户使用 Screen Builder 定义的指定特性，然后生成程序。用户可以象运行其它 FoxPro 程序那样来运行生成的程序。用户必须生成屏幕程序，这可以通过在 Screen Builder 中选择 Program 菜单中的 Generate 选择项来完成。初学者对于 Generate 对话盒(见图 3)可能有些望而生畏，但象这种简单的屏幕只需在一

个窗口中修改单个表格的情况，用户只需选择缺省的设置然后选择 Generate 选项。

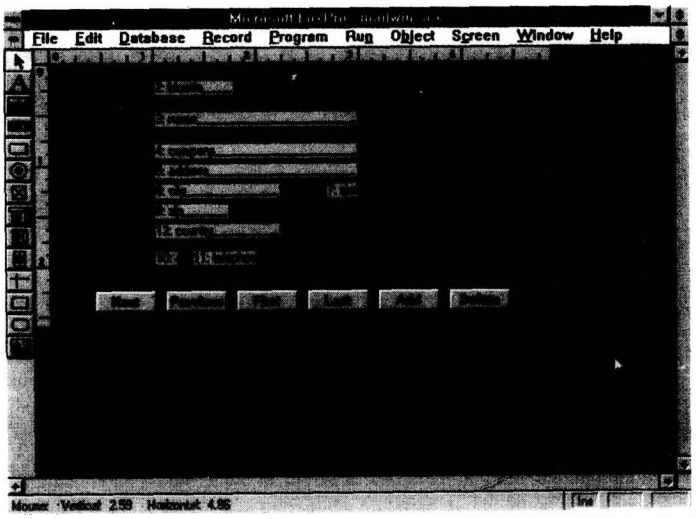

图 2　一个在 FoxPro for Windows Screen Builder 中构造的数据入口屏幕

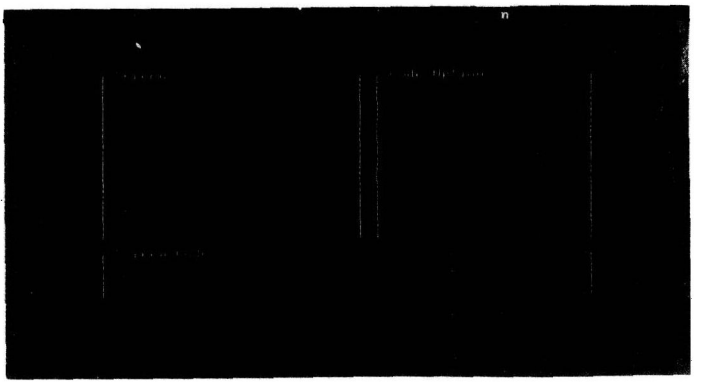

图 3　在 Screen Builder 中生成屏幕程序

屏幕文件的扩展名是. SPR，而不是. PRG，这样用户必须在运行程序 DO 命令中显式键入扩展名。例如：为运行图 3 所示的生成序列所生成的 MAILDOS. SPR 程序，用户需键入：

```
DO MAILDOS. SPR
```

由于 FoxPro 生成普通的程序以执行屏幕，所以用户易于插入定制代码到屏幕程序中。Screen Builder 也提供了数种方式以增加这些代码段。在本文中将讨论其中的一些方法。

图 4 给出了 FPD 中的 Field 对话盒。与 FPW 对话盒类似，不同之外仅在于校验盒被置换成命令按钮。其中的 Upper，Lower，Valid，Error 及 Message 选项将导出另一个对话盒，如图 5 所示，用户可以输入显式的表达式或 FoxPro 程序代码段到这个对话盒中。本期我们只讨论有关 When 和 Valid 的部分。

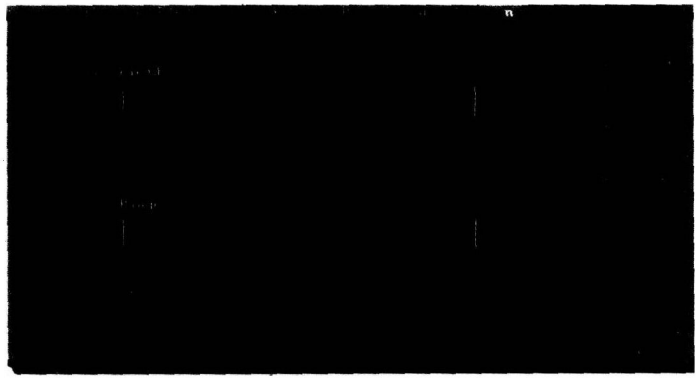

图 4　FoxPro for DOS 中的 Field 对话盒

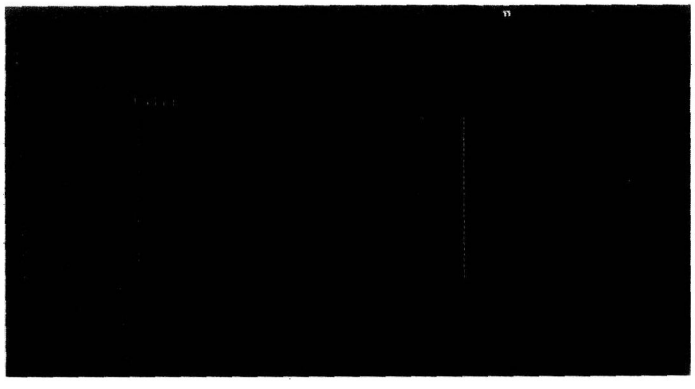

图 5　Zip 区域的 Valid 程序段

程序段的编写

顾名思义，When 程序段是预编辑条件，它决定了用户编辑区域的时间。Valid 程序是一个后定义条件，它决定了用户输入的数据是否能被接受。用户可以使用一个显式表达式或书写一段定制的程序，但在两种情况下结果都必须是一个逻辑值(.T. 或.F.)。若 When 表达式或代码段段值为.F.。FoxPro 将不允许用户将光标移入区域。若 Valid 表达式或代码段返回 False，FoxPro 将不允许用户将光标移出区域。例如：在邮政清单数据入口屏幕中，使用了 Valid 表达式 NOT EMPTY（地址）和 NOT EMPTY（城市）来验证 ADDRESS 和 CITY 区域，防止用户将其遗留为空白。若用户输入的数据验证失败时，可以通过进入 Error 程序段来定制错误消息显示。

尽管图 5 中无线按钮标签以 Procedure，用户编写的程序代码在生成的屏幕程序中成为 UDF。这一 UDF 应返回逻辑值.T. 或.F.（这一规则仅存在一种例外情况，但现在可以忽略不予讨论）。在用户的代码段中不要包含 FUNCTION 语句，但要指定 RETURN 数值。

用户可以直接将程序段健入位于 Procedure 和 Expression 无线按钮之下的滚卷盒中。但对于多于一行的情况，通过选择 Edit 来打开一个编辑窗口还是较容易的。麻烦的是 FoxPro 在对话盒之后打开编辑窗口，因此用户在开始编辑之前必须选择 OK 以关闭对话盒。用户一次可以打开多于一个的编辑窗口，并且在程序段之间剪取和粘贴代码。

用户可以使用 When 和 Valid 代码段，来指定比使用显式表达式更为复杂的预编辑和后编辑条件规则。例如，考虑定义有效的邮政号码的规则：长度必须为 5 或 10，每一位置必须存在一个数字，只有在 10 位邮政号码中第六个字符必须是一个破折号。这就是 U. S. 的邮政号码。Canada 的邮政号码包括 7 个字符，其顺序是形状字符、数字、字母、空格、数字、字母、数字。其他的国家应用其他规则。的邮政号码验证程序段：

```
mreturn=.T.                          && UDF return value
DO CASE
   CASE EMPTY(country)               && US zip codes
      IF LEN (TRIM(zip)) <> 5 AND;
         LEN (TRIM(zip)) <> 10
         mreturn=.F.                 && Wrong length
      ENDIF
      IF LEN(TRIM(zip))=10 AND
         SUBSTR(zip,6,1) <> "-"
         mreturn=.F.
      ENDIF
   CASE UPPER (country)="CANADA"
      IF LEN (TRIM (zip)) <> 7
         mreturn= .F.                && Wrong length
      ENDIF
ENDCASE
RETURN mreturn
```

这一版本的验证程序并不采用另一个可行的步骤，即验证每个 U. S. 邮政号码字符位置为数字占据，对于 Canada 邮政号码则是一个数字或字母。读者可以设想一下怎样完成这一版本，包括使用 SUBSTR()函数来抽取每个字符，使用 ISALPHA()和 ISDIGIT()函数来测试结果子字符串的内容。

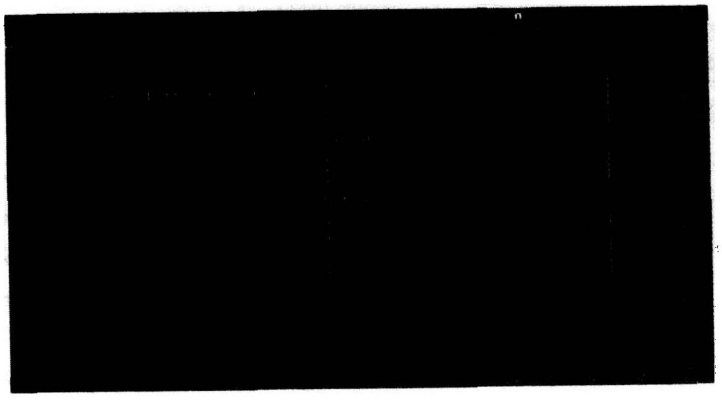

图 6　定义按钮

使用 When 和 Valid 程序段来指定预编辑和后编辑条件在原因上应该说是很直观的。然而，这些程序具有很广的应用性。设想一下 FoxPro 评估 When 和 Valid 代码的时间——用户试图进入或退出编辑区域时。这个时序的一个重要的结果是用户可以在任何需要完成这一操作的时间来使用这些程序。而不管它们在编辑区域中做些什么。

基于这个概念的一个功能很强的应用程序是在屏幕中放置命令按钮，以克服一个明显的局限——Screen Builder 在表格操作时未提供一个自动的方法。解决方法之一是使用 Menu Builder 来定义 System 菜单的一个新版本，但更容易和直接的方法是使用如图 1 和图 2 中所用的按钮。

用户一旦知道怎样使用判断结构，就可以为操作选择项使用一组按钮而不是分立的按钮。定义按钮是很容易的。在 Screen Builder 中，首先选择 Screen 菜单中的 Push Button（或使用热键 Ctrl＋H）来调用如图 6 所示的对话盒，其中包括了一些按钮的指定项，，通常并不使用所有的选择项。用户必须完成以下操作（并不必按照所列出的顺序）：

- 定义按钮提示。
- 选择 **Horizontal** 或 **Vertical** 以指定组按钮的原点。
- 若想改变按钮间缺省的间隔使用 **Spacing** 选择项。
- 输入存储选择按钮的内存变量名。
- 在 Valid 程序段中编写代码以执行按钮的操作。

上述的前三个步骤应该是很明显的，第四个步骤看起来不是那么必要。尽管用户很少使用按钮来收集数据。用户放置在屏幕上的对象（FoxPro for Windows 中的图形对象除外）都需要获得一个数值。对于按钮而言，这个数值存储在内存变量中。用户运行屏幕程序并选择按钮时，FoxPro 将向基于按钮位置的变量赋值——首先是 1，其次是 2，如此下去。

验证变量数值并完成合适的操作的程序段在按钮的 Valid 段中。当用户选择了按钮从而向按钮收集数据的变量输入数据时，就开始执行这段程序。

在检查邮件清单数据入口屏幕中的按钮的 Valid 代码段前，让我们先查看一个记录导向命令：

```
Next        SKIP
Previous    SKIP －1
First       GOTO TOP
Last        GOTO BOTTOM
Add         APPEND BLANK
Delete      DELETE
```

若用户使用的是 SET DELETED ON ，FoxPro 将忽略删除的记录，用户很可能在完成 DELETE 操作后 SKIP 到下一条记录。

用户还必须考虑 Screen Builder 生成的屏幕程序的另外一个特征：在移动记录指针后 FoxPro 并不自动重新显示。因此，用户选择了 Next 或 Previous 按钮后，屏幕上仍保留着先前的记录数据，直到用户在区域中移动了光标时，显示修改了的区域。幸运的是，对这个问题有一个很简单的解决方法——SHOW GETS 命令，它使 FoxPro 重新显示当前屏幕之上的所有输入（SHOW GETS 由@…GET 命令派生而来，GET 命令用于输入数据）。下面是按钮的 Valid 代码段：

```
DO CASE
  CASE moption =1
    SKIP
  CASE moption=2
    SKIP -1
  CASE moption=3
    GOTO TOP
  CASE moption=4
    GOTO BOTTOM
  CASE moption=5
    APPEND BLANK
  CASE moption =6
    DELETE
    SKIP
ENDCASE
SHOW GETS
RETURN. T.
```

使用无线按钮

用户可以使用无线按钮(也称单选按钮)以收集数据而不需编程,但这样做需要在数据存储方式做些不情愿的变通。若无线按钮收集的区域是一个字符域,用户在选择按钮时,FoxPro 将输入选择按钮提示的文本到区域中。用户在显示一个新记录时还将显示相对应的选择按钮,提供对应按钮提示的区域内容。不幸的是,这一限制带来很大的约束力。例如:MAILLIST 表格中的 ADDRTYPE 区域只有一个字符的长度。若要加长区域以容纳提示的全部文本("Home","Work"等等),将浪费大量的磁盘空间,而且致使 BROWSE 或 EDIT 命令的数据入口更为杂乱。

解决这个问题的方法是使用无线按钮来收集内存变量,而不是区域,并书写一段定制代码以完成必要的翻译,缺省时,FoxPro 假定内存变量是数字型的,并赋给基于选择按钮位置的数值。用户可能已推测出向表格中输入数据的过程应该在 Valid 代码中完成,在按钮选择后执行这段代码。下面是邮件清单数据入口屏幕中按钮的 Valid 代码段:

```
  DO CASE
CASE maddrtype=1
      REPLACE addrtype WITH "H"
    CASE maddrtype=2
      REPLACE addrtype WITH "W"
    CASE maddrtype =3
      REPLACE addrtype WITH "M"
    CASE maddrtype=4
      REPLACE addrtype WITH "X"
  ENDCASE
```

在按钮选择后为显示合适的按钮,每次在屏幕上显示新记录时,必须向基于区域数值的内存变量赋以合适的数值。下面给出程序:

```
  DO CASE
```

```
CASE addrtype= "H"
     maddrtype = 1
   CASE addrtype ="W"
     maddrtype=2
   CASE addrtype="M"
     maddrtype =3
   CASE addrtype="X"
     maddrtype =4
 ENDCASE
```

用户若把这段程序放入与无线按钮相关的代码段中，逻辑位置应是 When 代码。每次用户在屏幕上移动按钮时都将执行这段程序。然而，无论新记录何时在屏幕上显示，用户都需要修改早先显示的按钮。

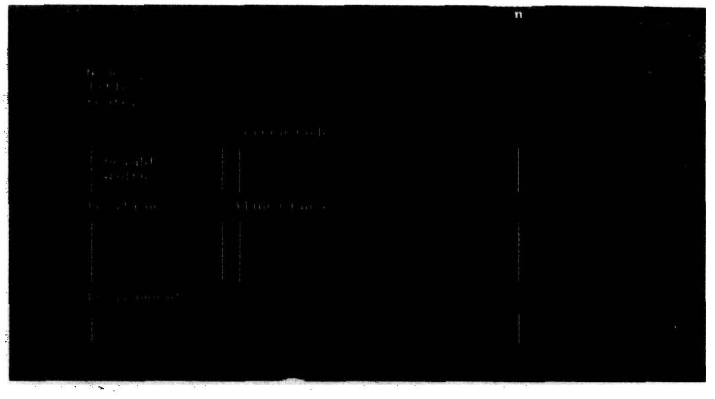

图 7　Screen Builder Layout 对话盒

修改无线按钮显示的最佳位置应该是在连接整个屏幕的代码段中，而不是屏幕中的任何单个对象。FoxPro 提供了 7 个这样的代码段，通过从 Screen 菜单(图 7)中选择 Layout 选择项而调用的对话盒可以访问它们。为到达 FoxPro for Windows 中的模拟对话盒。可以从 Layout 对话盒中选择标签为 Code 的按钮。有关这 7 个程序段的深层次的讨论远远超出了本文的范围，但在执行它们时有助于理解：

- Setup　　一次，在用户开始运行屏幕程序时。
- Cleanup　　一次，屏幕程序终止时。
- Activate　　任何时候用户想进入窗口时。
- Deactivate　　任何时候用户想退出窗口时。
- When　　一次，在屏幕上显示数据后。返回的数值决定了运行继续还是停止。
- Valid　　任何时候用户想从屏幕程序退出时。
- Show　　用户发出 SHOW GETS 命令时。

正如前面对邮件清单屏幕中导向按钮的讨论，用户在移动记录指针后必须先出 SHOW GETS，以确保新的当前记录中的数据显示在屏幕上。将修改无线按钮的程序段放入 SHOW 代码段保证了同时修改显示按钮。

欢迎尝试

尽管本文引进了许多新概念，但不要泄气，更不要回避 Screen Builder 的复杂性。用户不必关心其中提供的大多数选项。现在可以做的是，编写代码段完成以下操作：决定是否能够编辑数据（在 When 程序段中）；验证数据（在 Valid 程序段中）；执行命令按钮功能（Valid 程序段中）；以及修改显示数值（在屏幕的 Show 程序中）。

增加类查换：一个两级查表系统

Tamar E. Granor

很多应用软件都允许用户从一个有效入口表中查找输入项。通常，弹出式菜单用于显示有效入口。当用户从上托菜单中选择时，已选定值被放入适当的段或变量。通常在一个应用软件中同样的项会出现在许多屏幕中(例如，一个两个字母的州/省的缩写)。同样还会有许多不同的项可在某一应用软件内查找。

用户可以使用类查找规程，而无需编写代码来在每个查找出现的位置提供查找。这样一种规程可组成几种不同的方式。一种方法是便用类查表，它包含可能被查找的每个段所有可能的值。在类查表中的每个入口都表示它所使用的段。该查找规程便可由应用于手边的段入口组成一个弹出式菜单。

查找项是某一个更大结构的一部分，而该结构还包括可能被查找的段。

例如，在输入雇员档案时，查找雇员管理程序的雇员号码会很有用。此时，有效入口表是雇员表的子集，并且随时可更改。图 1 就是此类查找的例子。

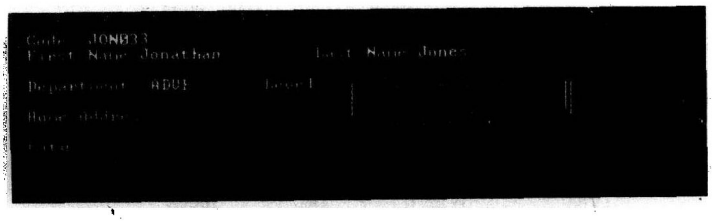

图 1　查找，在屏幕上方弹出菜单

另一个例子发生在跟踪不动产交易的应用软件中。对于一个给定的交易处，或者是某律师代表的团体。用户可以在已知的律师(或许已认可的)的律师中查找律师名。然而，律师档案本身也可由用户编辑。律师编辑屏幕可包括其他查换，例如状态段。

用户不想在一般的查表中包括管理程序或律师信息，因为这需要多个信息存放在另一表中。

管理程序和律师查找和状态查找的区别在于一个积极一个消极。管理程序和律师项为移动目标。对于使用的应用软件，这些表中的项都会改变。而状态项是静止的。尽管状态表经定义后可以改变，但它不作为应用软件的日常功能部分来改变。单个查表很适用于消极/静止项，但对于积极项用处不大。

另一个设计查找规程的方法是使用可表示被查找项所在的表和段的参数。这种方法也存在问题。首先，此方法会意味着关于指定段查找的信息每次出现在屏幕上都是特定的。其次，它要求在调用查找规程时查找信息是已知的。该方法不允许用户在菜单中使用一个类查找选项。