VB 动态控件数组的设计与实现

张　庄　(福建省经济信息中心　350003)

摘要:一个好的应用程序标准之一是少占计算机内存,本文提出一种动态与数组的设计方法使编写的应用程序很好地解决少占计算机内存的问题。

关键词:VB　计算机内存　动态数组

一、前言

设计应用程序的一个很重要的概念,就是尽可能地让应用程序少占计算机内存,以避免系统资源的浪费。这里所谓的「系统资源」,主要是针对内存而言。例如,你设计的应用程序允许用户启动许多个窗口,那么这些窗口将占用大量的内存;如果在设计阶段,将这些表格窗口(form)分开设计,不但耗费许多内存,且用户仅能启动所设计绘制的表格窗口。此时,有一权宜之计就是利用「动态控件数组」,动态载入所需的表格窗口,不使用时即将其释放。

建立控件数组方法有两种:第一种是在设计阶段时建立,属于静态方式;另一种是执行阶段,动态地建立控件数组。所谓「动态地建立控件数组」是利用程序码来建立控件数组,而且是在程序执行时建立的。一般而言,欲使用的控件数组,如果在设计阶段建立,那么会有许多的内存被控件数组所占用,这样会造成内存的浪费;而使用动态控件数组,目的就是避免这些内存的浪费。动态建立控件数组会节省内存空间,就是在程序模组(module)内,先声明一个空的控件数组,等到要使用该控件数组时,再以 ReDim 重新声明欲使用的控件数组元素或数目;而当不再使用某一个或某些控件数组的元素时,可以利用 Unload 将它释放掉。本文拟就「动态控件数组」的建立做一概略说明,并以一实际的例子来阐述程序的编写技巧。

二、动态控件数组的建立与释放

由于「动态控件数组」的概念是起缘于一般程序语言的「动态数组」,因此在说明「动态控件数组」之前,首先必须了解一般「动态数组」的建立及程序编写方法。所谓「动态数组」是指该数组在程序未执行之前并未配置任何内存给该数组,而是等到程序执行到该声明数组时,才会要求系统配置内存给该数组使用。一般声明动态数组的方法是在表格 general 中以 Dim 来声明一个空的动态数组,然后在程序中再以 ReDim 来声明其大小。例如:

1. 在表格 general 中:以 Dim 来声明一个空的动态数组

```
Dim arr( ) As Integer
```

2. 在程序中:以 ReDim 来声明其实际大小

```
ReDim arr(20) As Integer
```

程序执行时,若使用动态数组,则可利用 ReDim 语句重新定义数组的大小或是用 Erase 语句将数组清除。而静态数组却只能使用 Erase 语句将数组元素设为初值,而无法将静态数组所占用的内存空间释放掉。由于静态数组是在程序加载时即要求内存空间,因此若一个程序使用大量的静态数组时,会占用掉较多的内存。当程序加载主内存中,若发生内存不足的现象时,会出现 "Out of Memory" 的信息。若程序中使用动态数据,则当程序执行到需要要求内存空间给数组使用时,若发生内存不够的现象,才会出现 "Out of Memory" 信息而结束程序。

同样地,动态控件数组也有以上的特性,只是释放数组的方法有些差异罢了。其实,使用控件数组的主要目的是在简化程序,使得我们编写的程序精致化、更具价值性;而使用动态控件数组的主要目的是要节省内存空间,避免一些没有使用到的控件占用内存,同时还可以看到程序本身简化了,而且也显得比较严谨。用下面的一个例子,来说明控件数组的优点。

假设要设计一个计算器的程序,我们都知道计算器的面上有 0 至 9 的十个按键,要把所按的数字显示在屏幕(以 TextBox 替代)上。如果这十个数字按键不使用控

件数组，程序必须编写如下：

```
Private Sub CmdNumber0-Click()
  Text1.Text=Text1.Text & 0
End Sub
Private Sub CmdNumber1-Click()
  Text1.Text=Text1.Text & 1
End Sub
  :
  :
Private Sub CmdNumber9-Click()
  Text1.Text=Text1.Text & 9
End Sub
```

在此你可以看到一个单调而冗长的程序，且它们所处理的是一些千篇一律的事件。现在使用控件数组，把数字按钮的名称(Name)设成 CmdNumber，并依其显示的标题(Caption)设定其属性 Index(即按钮上显示 1，则设定该按钮的 Index 属性值为 1)。那么，就可将上述较无技巧性的冗长程序修改如下：

```
Private Sub CmdNumber-Click(Index As Integer)
  Text1.Text=Text1.Text & Index
End Sub
```

看看以上的程序，运用了控件数组之后，整个程序变得那么短，而且变得非常精致，这就是善用控件数组的好处。至于如何设计「动态控件数组」呢？首先在模组中或表格的 general 中声明控件数组类型，然后在程序中重新声明其该数组大小。声明控件数组类型的定义如下：

语法：Dim | ReDim | Static | Global 数组类型名称 As [New] 数据类型

例如：Dim MyForm() As New Form1

上述语法中除了 New 之外，Dim、ReDim、Static 和 Global 的使用和一般变量的声明方法完全一样。其中使用 New 关键字，表示当 MyForm 这个控件数组变量在程序中首次被使用时，会自动产生一个 Form1 的表格(即复制一个与 Form1 相同的表格)。此时 MyForm 是一个空的数组，尔后在程序中再以 ReDim 来声明其大小。上例中 MyForm 这个控件数组变量所参考的是一个已经存在的表格，如果想参考任意表格或任意控件项，可声明成：

```
Dim AnyForm() As Form
Dim AnyControl() As Control
```

至于控件类型，则可分为「一般类型」及「特定类型」两种。一般的控件类型又可区分成三种：第一种是 Form，其参考控件是应用程序中的任一表格(包括 MDI 表格及其子窗口)；第二种是 Control，其参考控件是应用程序中的任一控件项；第三种是 MDIForm，其参考控件是应用程序中的 MDI 表格。而特定控件类型的种类非常多，除了 Visual Basic 本身所提供的控件外，还有许多第三者所发展的 VBX 或 OCX，都有其特定控件类型。这些特定控件类型，可以在属性窗口上获悉。如图 1 所示，在这个窗口中显示了 C_OK 控制项的特定型态为 CommandButton。

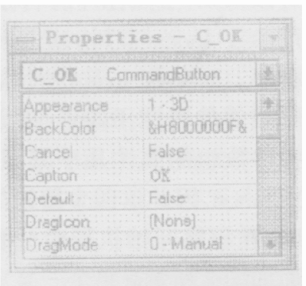

图 1　在属性窗口上显示 C_OK 控制项的特定型态为 CommandButton

三、动态控件数组的应用实例

本文动态控件数组的应用例子是一个信息窗口的显示程序，在本程序内使用了四个命令钮(CommandButton)的控制项，而这些控制项彼此间的行为非常类似，且可共用相同的事件程序，因此以静态控件数组方式来设计，以简化程序。另外所要显示的信息窗口，则以动态控件数组方式来设计本程序共使用一个模组(module)——dynamic.bas 及两个表格窗口(form)：dynamic.frm 和 msgshow.frm 。dynamic.frm 为程序的主表格窗口；msgshow.frm 为「关于本程序」的表格窗口，是本程序的一个控件。而 dynamic.bas 模组主要是声明一个空的控件变量数组，供其后某程序中重新声明使用。以下就设计本程序时所运用到的技巧，加以概略说明：

1.动态控件数组的声明

首先在 dynamic.bas 模组中声明一个空的控件变量数组 MsgBox()，而其所参考的控件是事先设计好的一个信息窗口 MsgForm。其声明方法如下：

```
Global MsgBox() As New MsgForm
```

要便用动态控件数组的话，必须在程序中重新声明，才能使用该数组。在本程序中，将此声明放在 Private Sub MsgEffectCmd-Click(Index As Integer)这个程序中。而欲启动的信息窗口，则参考静态控件数组 MsgEffectCmd 的 Index 值，以决定欲显示的窗口及其显示效果。动态控件数组重新声明的方法如下：

```
ReDim MsgBox(Index) As New MsgForm
```

2. 静态控件数组的运用

本程序内使用了四个命令钮控制项，而这些控制项主要是用来显示信息窗口，因为彼此间的行为非常类似，且可共用相同的事件程序，因此以静态控件数组方式来设计。在事件处理程序的参数中，多出了 Index 这个参数。由此我们可以得知：虽然控件的名称是相同的，但我们还是可以利用参数 Index 来判别是哪一个控件触发这个事件的处理程序。而参数 Index 所代表的值正是触发该事件处理程序控件的属性 Index。由于本程序所使用的四个命令钮控件数组的属性几乎都相同，唯一不同的就是它的属性 Caption，因此这四个按钮的 Caption，利用下列一小段程序，在程序开始执行时分别赋值。程序开始执行时的画面如图 2 所示。

```
Private Sub Form-Load()
  MsgEffectCmd(0).Caption = "文字技巧一"
  MsgEffectCmd(1).Caption = "文字技巧二"
  MsgEffectCmd(2).Caption = "文字颜色"
  MsgEffectCmd(3).Caption = "窗口展开"
End Sub
```

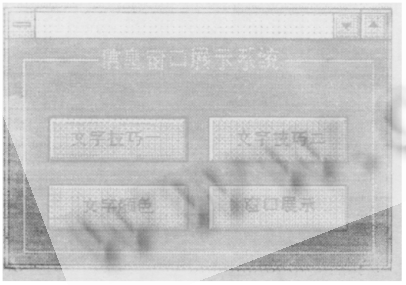

图 2　程序执行时的画面

3. 信息窗口显示技巧

本程序所列举的四种信息窗口显示技巧，是利用动态控件数组设计，因此程序执行之初，这四个窗口并未载入。我们是利用主表格窗口中的四个按钮之一，来启动某一个信息窗口。而这四个窗口所参照的表格窗口是 MsgForm，如图 3 所示：

四种信息窗口所欲显示的分别为「文字技巧一」、「文字技巧二」、「文字颜色」和「窗口展开」。其中「文字颜色」是利用随机数，使得窗口中每次出现的文字颜色不同；而「文字技巧一」、「文字技巧二」和「窗口展开」则是利用控件大小及位置的改变，来达到一种特殊的显示效果。由于牵涉到控件的大小和位置，因此各控件的 Height、Left、Top 和 Width 属性，必须经过一番设计，表 1 所示为信息窗口中，有特殊变化的几个控件的 Height、Left、Top 和 Width 属性值。

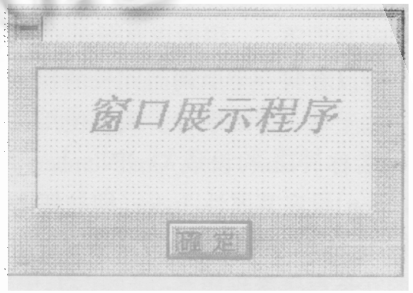

图 3　本程序所参照的表格窗口 MsgForm

表 1　控件的 Height、Left、Top 和 Width 属性值

	Height	Left	Top	Width
Picture	1500	* * *	* * *	3600
Label1	300	500	200	2700
Label2	300	900	700	2100
Label3	300	900	1100	2400

「文字技巧一」是利用 Label 的移动，使得 PictureBox 中的文字分别由左、右移到中央。设计这种效果时，一定得将 Picture 及 Label 的大小及位置计算精确，然后利用循环来达到这种动画效果。以下一小段程序即可使得文字，分别从 PictureBox 左、右外方移到 PictureBox 中央。

```
MsgBox(Index).Visible = True
For i = 1 To 100
  MsgBox(Index).label1.Left = 27 * i - 2200
  MsgBox(Index).Label3.Left = 3600 - 27 * i
Next
```

「文字技巧二」亦是利用 Label 的移动，使得 PictureBox 中的文字分别由上、下移到中央。这种效果的设计与「文字技巧一」相同，必须利用循环来达到这种动画效

果。以下一小段程序即可使得文字，分别从 PictureBox 上、下外方移到 PictureBox 中央。

```
MsgBox(Index).Visible = True
For i = 1 To 100
  MsgBox(Index).label1.Top = 7 * i - 500
  MsgBox(Index).Label2.Top = 1800 - 11 * i
  MsgBox(Index).Label3.Top = 1800 - 7 * i
Next
```

「文字颜色」的技巧，首先在按下该按钮时，即利用 Randomize Timer 产生随机数，然后指定随机数，以 RGB() 函数使得窗口中每次出现的文字颜色不同（如图 4 所示）。这个技巧的主要作用是使得我们目前出现此信息窗口内的文字颜色，与上次出现的不尽相同，让欲显示的信息窗口更富变化，也较多采多姿。

```
MsgBox(Index).Visible = True
j = Int(Rnd * 256)
MsgBox(Index).label1.ForeColor = RGB(((j * 3) Mod 125), 255 - j, j)
MsgBox(Index).Label2.ForeColor = RGB(j, j Mod 125, ((j * 3) Mod 256))
MsgBox(Index).Label3.ForeColor = RGB(255 - j, ((j * 3) Mod 256), j)
```

图 4　利用随机数改变信息窗口内的文字颜色

「窗口展开」的设计方法很简单，只要知道信息窗口的大小，我们即可利用一个循环使其逐渐展开。因为本程序的信息窗口是动态参考一个事先设计好的表格控件 MsgForm，而该表格的的大小是 4200 * 2800（单位：twip），因此我们可以利用下列的一段程序使得我们欲显示的窗口渐渐变大，并保持在屏幕的中央。

```
For i = 0 To 40
  MsgBox(Index).Height = i * 70
  MsgBox(Index).Width = i * 105
  MsgBox(Index).Left = (Screen.Width - MsgBox(Index).Width)  2
  MsgBox(Index).Top = (Screen.Height - MsgBox(Index).Height)  2
  MsgBox(Index).Visible = True
Next
```

4. 动态控件数组的释放

本程序动态载入的信息窗口，其本身是一个表格，因为是参照原先设计好的表格 MsgForm，因此几乎所有的属性都与 MsgForm 相同，在 Visual Basic 我们称这些动态加载的信息窗口为「复本」（Instances）。在本程序中，每个动态加载的信息窗口虽然都可拥有自己独特的属性，但是却共用相同的程序码。而且不同的复本之间可以共用整体变数，至于模组层次变数（例如 Index），当然被限定在复本（或某个程序）之内。

由于同一个表格允许有许多复本存在，因此就会产生不知要处理那一个复本的问题。例如 Unload 叙述就必须很明确指定某一个准备要被释放的表格名称。所幸的是 Visual Basic 提供了一个 "Me" 关键字，这个关键字代表目前正在执行的表格窗口。因此我们可以利用 Unload Me，将目前正在执行的表格复本给释放掉。

四、结语

在 Windos 的环境下，有许许多多的应用软件，几乎都用到了「动态控件数组」的概念。例如 Microsoft Word 这个文书处理的应用软件，在执行之初，它会自动动态加载一个未命名的 MDIForm - -「文件一」，而且会将功能选单的选项全部载入；当我们将「文件一」释放掉时，大部分功能选单的选项也会被释放掉，也就是说这个软件会根据用户的需求，自动加载或释放某些功能选项及表格窗口。Microsoft Word 这种人性化的设计，不但考虑到用户的需求，而且也顾及了系统资源的维护。相信经过了该文章的介绍，有心从事 VisualBasic 程序设计者，一定可以更加了解 Windows 下，一些应用软件的设计理念；也可使自行编写的程序更加精致化与人性化。最后，若对本文有任何问题，欢迎寄 E - mail 给我，E - mail 地址是 zhzh@sun1.fjic.gov.cn

（来稿时间：1999 年 6 月）