

2005 年 4 月刊

我从哪儿找到重绘? — Doug Hennig 著 CY 译

Page.3

Doug Hennig 继续他的关于 VFP9 报表监听器的讨论，通过一组类以输出报表的内容到游标，并利用这个游标提供一个“活动”的报表预览界面。

从 Visual FoxPro 中访问 Hotmail 和 MSN 帐号 — Anatoliy Mogylevets 著 Fbilo 译

Page.15

在某些情况下，用 Visual Foxpro 代码发送和接收 Hotmail 消息是可能的。在这篇文章中，Anatoliy Mogylevets 讲解了一些类，这些类使用了微软 ServerXMLHTTP 和 XMLDOM 对象来实现对 Hotmail 服务器的 WebDAV 访问。你还可以学到一些 HTTP 请求的奥秘，包括请求的头。

来自 VFP 开发团队的 TIPS — 微软 VFP 开发团队 著 LQL.NET 译

Page.34

这个月来自 VFP 开发团队的 TIPS 向你展示了 VFP9 报表方面的一些新功能，这些功能在你现在买的 VFP9 中均已兑现。

用一个全局性的解决办法去解决一个局部问题不是明智的。为什么这么说呢？因为当 Andy Kramek 和 Marcia Akins 调用以前写的一些代码的时候发现了问题。一些命令如 SET EXACT 和 SET UDFPARMS 的影响我们大家都知道，而 SET COMPATIBLE 带来的结果却是令人吃惊的。来看看 Andy 最近碰上的麻烦事儿。

我从哪儿找到重绘？

原著: Doug Hennig

翻译: CY

Doug Hennig 继续他的关于VFP9报表监听器的讨论，通过一组以输出报表的内容到游标，并利用这个游标提供一个“活动”的报表预览界面。

当 你在VFP8以前里运行报表时，输出就象是一个黑框：在预览窗口里你有一个小控件，没有任何关于什么在哪儿重绘的信息，并且不提供“活动”的预览界面（可以捕捉在它上面的点击事件以提供某些特定对象的动作）。

在过去的两个月中，我已经讨论过VFP9里新的ReportListener类，和如何利用它以先前无法实现的办法来控制报表的输出。我所展示的ReportListener子类，可以实现某些类型的可视输出，比如HTML，带动态格式的报表，等等。

本月，监听器的输出对用户来说不再是任何可见的地方，取而代之的是，把它发送到到游标，于是我们可以跟踪什么在哪儿重绘。有了这个信息可以提供所有有趣的应用，比如，活动的预览报表界面，动态生成的内容表，文本查找的能力，有条件的高亮显示报表里的对象，等等。

DBFListener

DBFListener包含在DBFListener.PRG里，是_ReportListener的子类，定义在_ReportListener.VCX里的报表监听器子类，在VFP根目录的FFC子目录下。我在*FoxTalk 2.0* 2005年1月栏目里讨论过，“监听报表”。因为它是_ReportListener子类，DBFListener可以用于报表示单的监听器或是作为监听器链的成员，每一个在报表运行时提供某些功能。DBFListener的ListenerType设置为3，以忽略输出到预览窗口和打印机。在报表运行前，BeforeReport事件里的代码创建了一个游标或表来保存重绘的报表内容。若要创建表，设置lUseCursor 属性为.F. 以及cOutputDBF为所要创建的表文件的文件名和路径（如果你没有指定文件名，就会以SYS(2015)的名来用在WINDOWS的临时目录下）。若要创建游标，设置lUseCursor为.T. 以及cOutputAlias为所使用的游标的别名（如果没有指定别名，就使用SYS(2015)的名）。

在任何一种情况下，表或游标都有栏目对应于FRX里记录号的报表对象，OBJTYPE 和 OBJCODE值来源于FRX（它指明对象的类型），以及重绘对象的左距、顶距、宽度和高度，以及传递给Render方法的“连续类型”参数（参见VFP帮助里Render主题讨论这个参数），还有对象的内容（如果它是栏或标签），以及它所出现的页号。

```
function BeforeReport
local lcTable
with This
```

* 如果表名或别名没有指定，就创建默认名

```
if empty(.cOutputDBF)
.cOutputDBF = addbs(sys(2023)) + sys(2015) + ;
'.dbf'
endif empty(.cOutputDBF)
if empty(.cOutputAlias)
.cOutputAlias = ;
strtran(juststem(.cOutputDBF), ' ', '_')
endif empty(.cOutputAlias)
```

* 如果游标已经打开，先关闭它。如果表已经存在，先清除它。

```
use in select (.cOutputAlias)
if file(.cOutputDBF)
erase (.cOutputDBF)
erase forceext(.cOutputDBF, 'FPT')
erase forceext(.cOutputDBF, 'CDX')
endif file(.cOutputDBF)
```

* 创建表或游标

```
lcTable = iif(.lUseCursor, 'cursor ' + ;
.cOutputAlias, 'table ' + .cOutputDBF)
create &lcTable (FRXRECNO I, OBJTYPE I, ;
OBJCODE I, LEFT I, TOP I, WIDTH I, HEIGHT I, ;
CONTTYPE I, CONTENTS M nocptrans, PAGE I)
index on PAGE tag PAGE
endwith
```

* 执行常规动作

```

dodefault()
endfunc

```

当报表里的每一个对象重绘时，Render事件就会触发。DBFListener里这个事件的代码加入一条记录到游标里。由于文本栏和标签是以Unicode来传递的，它将不得不用STRCONV()转换为普通文本以使得可用。Render 使用定义在父类里的SetFRXDataSession和ResetDataSession帮助方法，以来回切换FRX游标的数据工作期。这使得Render可以从FRX里的当前对象获取OBJTYPE 和OBJCODE的值。注意：这个代码当前并不对图像作任何事，因为现在还没有确定该对它做什么。

```

function Render(tnFRXRecNo, tnLeft, tnTop, tnWidth, ;
tnHeight, tnObjectContinuationType, ;
tcContentsToBeRendered, tiGDIPlusImage)
local lcContents, ;
liObjType, ;
liObjCode
with This
if empty(tcContentsToBeRendered)
lcContents = ''
else
lcContents = strconv(tcContentsToBeRendered, 6)
endif empty(tcContentsToBeRendered)
.SetFRXDataSession()
go tnFRXRecno in FRX
liObjType = FRX.OBJTYPE
liObjCode = FRX.OBJCODE
.ResetDataSession()
insert into (.cOutputAlias) ;
values (tnFRXRecNo, liObjType, liObjCode, ;
tnLeft, tnTop, tnWidth, tnHeight, ;
tnObjectContinuationType, lcContents, ;
.PageNo)
endwith
endfunc

```

Destroy方法(此处未列出)关闭了游标或表,并删除了表文件(如果lDeleteOnDestroy属性为.T.)。

当DBFListener用于作为报表的监听器时,没有任何事情出现;报表没有预览,打印,输出为HTML,或其它任何别的东东。然而,在报表运行完成后,就有一个包含每个报表元素和何处重绘的信息的表或游标。

SFPreview

一旦你已经有了在游标或表里的需要重复绘的报表内容,你就可以利用它来作很多事情。我将在本文里为你展示一二。

SFPreviewForm是一个表单类,它提供了一个与VFP自带的预览窗口具有不同性能的报表预览对话框。它在报表对象被点击时发起事件,并支持其他性能,比如文本查找。它使用某些诡计来实现的:因为预览页是一个GDI+图像,当你在图像上点击文本时不会任何事情发生。

SFPreviewForm通过在预览界面上为每个重绘对象创建一个形对象以支持报表对象事件。当然,这些形对象可以捕捉事件,诸如鼠标的移动或点击,使得它可以有活动的预览界面。形对象并没有直接加入到表单,而是一个放置在表单上的容器时。因为必须为你在预览时所指向的每一页创建不同的形组,可以很容易来删除容器的(它可以一次性删除所有的形对象),并且创建一个新的而不是先移去原来的形再加入新的形。

在SFPreviewForm 里的主要方法是 DisplayPage。这个方法显示了报表的当前页,并以同样的大小和位置为页面上的每一个对象创建了形对象。然而DisplayPage是如何知道哪个报表对象是出现这个页上?当然,是通过查找DBFListener所创建的游标。

```
lparameters tnPageNo
local lnPageNo, ;
lcObject, ;
loObject
with This
```

* 如果我们还没有初始化,就先做吧。

```
if vartype(.oListener) = '0'
if not .lInitialized
.InitializePreview()
endif not .lInitialized
```

* 确认我们已经有空的形容器。

```
.AddShapeContainer()
```

* 确认已经指定特定的页号

```
if between(tnPageNo, .nFirstPage, .nLastPage)
lnPageNo = tnPageNo
else
lnPageNo = .nFirstPage
endif between(tnPageNo, .nFirstPage, .nLastPage)
```

* 选择输出游标并为指定页上的每一个报表对象创建形

```
select (.cOutputAlias)
seek lnPageNo
scan while PAGE = lnPageNo
.AddObjectToContainer()
endscan while PAGE = lnPageNo
```

* 设置当前页号并重绘页

```
.nCurrentPage = lnPageNo
.DrawPage()
```

* 标记是否在首页或末页

```
.lFirstPage = lnPageNo = .nFirstPage
.lLastPage = lnPageNo >= .nLastPage
```

* 刷新工具栏（如果需要）

```
.RefreshToolbar()
```

* 如果没有监听器对象，将无法处理

```
else
```

```

messagebox('There is no listener object.', 16, ;
.Caption)
endif vartype(.oListener) = '0'
endwith

```

这个代码是从调用InitializePreview开始的，如果预览还没有被初始化，然后调用AddShapeContainer以加入一个容器到表单上以用于放置形对象。我们没有在这里见到AddShapeContainer，它只是移去任何已经存在的容器并从类里加入一个新容器，这个类的类名和类库是由cContainerClass 和cContainerLibrary属性所指定的。然后DisplayPage确认已经指定有效的页号，并遍历所要重绘输出的游标，为当前页上每一个对象加入一个形对象到容器里。然后它设置nCurrentPage属性为页号，并调用DrawPage来在表单上为当前页显示预览图像。DisplayPage 更新了lFirstPage 和 lLastPage，使得工具栏上的按钮可以正确设置为允许和禁止（例如，当lLastPage为.T.时，Last Page按钮为禁止），并刷新工具栏。

InitializePreview是在第一次调用DisplayPage时被调用的，以确保当前属性被初始化正确。正如在这个方法的注释所指明的，复杂的是如果你在报表运行时使用了RANGE子句，比如RANGE 6,7，这些页会被重编号为6和7，但是当你在调用监听器的OutputPage方法在表单上重绘预览图像时，第一页是1，第二页是2，以此类推。为了克服这个在编码安排上可能的错误，InitializePreview设置nFirstPage 和 nLastPage属性为首页和末页（此例为6和7），并且nPageOffset为从“实际”页号里减去的值以获得输出的页号。

InitializePreview也把报表页的高度和宽度放入到nMaxWidth 和 nMaxHeight属性。这些值可用于设置报表预览容器的大小，如果它比表单大，就会出现滚动条，因为表单的ScrollBars 属性被设置为3-Both。

注意这里有些复杂的。首先，当表单是使用像素时，页的高度和宽度是以1/960英寸为单位。幸好，可以很容易的在1/960英寸和像素间转换：除以10，因为报表引擎是以96DPI来重绘的。第二个复杂的是，DBFListener对象不是运行报表的首个监听器，它的GetPageWidth 和 GetPageHeight方法无法返回正确的值。幸好，_ReportListener通过设置定制的SharedPageWidth 和 SharedPageHeight为相应的值来处理这个问题。

最后，InitializePreview清除用于文本查找的某些属性（我们将在稍后看到），打开用于形的类库，它将为报表对象加入到表单，并标记初始化已经完成，于是这个方法不会再被调用第二次。

```

with This

```

* 设置首页和首页偏移值。即使我们在RANGE子句没有输出第一页，页号从1开始，一直到OutputPage。

```

.nFirstPage = .oListener.CommandClauses.RangeFrom

```



```
.nPageOffset = .nFirstPage - 1
```

- * Width 和 Height值为报表的1/10，因为这些值是以1/960英寸为单位，而报表引擎的分辨率是96DPI，
- * 我们的监听器可能是后继者，于是使用相应的Shared属性（如果存在）。
- * 同样，利用SharedOutputPageCount（如果这个监听器是首个监听器并且没有后继者，那么它没有被填充）
- * 或OutputPageCount（调整偏移值）来获取上一页的页号。

```
if pemstatus(.oListener, 'SharedPageWidth', 5)
.nMaxWidth = .oListener.SharedPageWidth/10
.nMaxHeight = .oListener.SharedPageHeight/10
if .oListener.SharedOutputPageCount > 0
.nLastPage = ;
.oListener.SharedOutputPageCount + ;
.nPageOffset
else
.nLastPage = .oListener.OutputPageCount + ;
.nPageOffset
endif .oListener.SharedOutputPageCount > 0
else
.nMaxWidth = .oListener.GetPageWidth()/10
.nMaxHeight = .oListener.GetPageHeight()/10
.nLastPage = .oListener.OutputPageCount + ;
.nPageOffset
endif pemstatus(.oListener, 'SharedPageWidth', 5)
```

- * 清除查找设置

```
.ClearFind()
```

- * 打开相应的类库（如果需要）

```
if not '\ ' + upper(.cShapeLibrary) $ ;
set(' CLASSLIB')
.lOpenedLibrary = .T.
```

```
set classlib to (.cShapeLibrary) additive
endif not '\'
```

* 标记已经完成

```
.lInitialized = .T.
endwith
```

DrawPage是从DisplayPage里调用，以在表单上重绘当前的预览页图像，调用监听器的OutputPage方法，传递给它页号（调整为相对于起始页偏移），那个容器用于作为图像的占位符，其值为2，它指明输出到VFP控件。DrawPage也调用HighlightObjects来高亮显示我们所需要的任何报表对象，我将稍后讨论。注意表单的Paint事件也调用DrawPage，因为当表单在重绘时（比如在缩放时），占位符容器被重绘，因而预览图像丢失，因此用DrawPage来恢复它。

```
with This
if vartype(.oListener) = '0'
.oListener.OutputPage(.nCurrentPage - ;
.nPageOffset, .oContainer, 2)
.HighlightObjects()
else
messagebox('There is no listener object.', 16, ;
.Caption)
endif vartype(.oListener) = '0'
endwith
```

AddObjectToContainer，是从DisplayPage里调用的，它加入定义在cShapeClass里的形类（类库是在先前InitializePreview里打开的cShapeLibrary所指定的），以作为当前报表页对象的形容器。这个形的大小和位置是基于游标里的HEIGHT，WIDTH，TOP，和 LEFT 栏，虽然我们早已经见过，这些必须除以10以转换为像素。

```
local lcObject, ;
lcObject
with This
lcObject = 'Object' + transform(recno())
.oContainer.AddObject(lcObject, .cShapeClass)
```

```

loObject = evaluate('.oContainer.' + lcObject)
with loObject
.Width = WIDTH/10
.Height = HEIGHT/10
.Top = TOP/10
.Left = LEFT/10
.nRecno = recno()
.Visible = .T.
endwith
endwith
return loObject

```

处理事件

CshapeClass属性是被默认设置为“SFReportShape”。SFReportShape是Shape的子类，在它的Click, RightClick, 和DblClick事件代码里，它调用了表单的OnObjectClicked方法，并传递给它表示这个形的报表内容游标的记录号，和表明发生事件动作的数值（1 为 Click, 2 为 DblClick, 或 3 为 RightClick）。这可以让SFPreviewForm接收通知是否有报表对象被点击。

OnObjectClicked通过发起对应于点击类型的相应事件来处理在形对象上的点击。有助于使用 RAISEEVENT() 的是任何对象都可以利用BINDEVENT() 绑定ObjectClicked, ObjectDblClicked, 或 ObjectRightClicked, 以实现所希望的动作，而无需对SFPreviewForm作子类。你甚至可以有多个动作，如果你愿意，因为多个对象可以绑定于同一个事件。绑定于这些事件的任何对象可以接收在报表内容游标里的当前记录的SCATTER NAME对象来作为参数。

```

lparameters tnRecno, ;
tnClickType
local loObject
select (This.cOutputAlias)
go tnRecno
scatter memo name loObject
do case
case tnClickType = 1
raiseevent(This, 'ObjectClicked', loObject)
case tnClickType = 2
raiseevent(This, 'ObjectDblClicked', loObject)
otherwise

```

```
raiseevent(This, 'ObjectRightClicked', loObject)
endcase
```

这里是SFPreviewForm里的其他方法。利用这个类演示了一个工具栏，类库名是指定在ToolbarClass和 cToolbarLibrary里（如果lShowToolbar 为 .T.）。FirstPage, PreviousPage, NextPage, 和 LastPage 方法调用了DisplayPage, 并传递相应的数值以显示所需要的页。SaveFormPosition 和 SetFormPosition在报表运行时保存和恢复预览表单的大小和形。我将在下月讨论其余这些方法。

活动的预览界面

让我们来试试。TestSFPreview.PRG 使用 DBFListener来作为Customers报表的监听器，然后例化了SFPreviewForm, 设置属性为相应的值，并告诉它来显示第一页。

```
loListener = newobject('DBFListener', ;
'DBFListener.PRG')
report form Customers object loListener
```

* 在定制的预览器里显示报表

```
loForm = newobject('SFPreviewForm', 'SFPreview.vcx')
with loForm
.cOutputAlias = loListener.cOutputAlias
.Caption = 'Customer Report'
.oListener = loListener
.FirstPage()
endwith
```

然后TestSFPreview.PRG例化一个对象以处理在预览界面上的点击，并绑定不同的点击事件。最后，它显示了Debug Output窗口（因为它是由点击处理类的点击事件所响应的），并显示一个预览表单。

```
loHandler = createobject('ClickHandler')
bindevent(loForm, 'ObjectClicked', loHandler, ;
'OnClick')
bindevent(loForm, 'ObjectDblClicked', loHandler, ;
'OnDblClick')
bindevent(loForm, 'ObjectRightClicked', loHandler, ;
```

```
'OnRightClick')
```

* 显示高度输出窗口和预览表单

```
activate window 'debug output'  
loForm.Show(1)
```

这里是点击处理类的部分说明（OnDb1Click 和 OnRightClick方法并没有展示，是因为它们与OnClick是很相似的）。传递对象的ObjType属性指明何种报表对象被点击（例如，5表示标签，8表示栏），并且Contents属性包含了上述标签或栏的内容。

```
define class ClickHandler as Custom  
procedure OnClick(toObject)  
do case  
case inlist(toObject.ObjType, 5, 8)  
debugout 'You clicked ' + ;  
trim(toObject.Contents)  
case toObject.ObjType = 7  
debugout 'You clicked a rectangle'  
case toObject.ObjType = 6  
debugout 'You clicked a line'  
case toObject.ObjType = 17  
debugout 'You clicked an image'  
endcase  
endproc  
enddefine
```



图1: SFPreviewForm, 组合了DBFListener, 提供了活动预览界面。

当你运行TestSFPreview.PRG时, 将会见到预览表单如图1所示。虽然这看起来很像VFP自带的预览表单, 试着点击不同的报表对象。你将会看见对象响应到Debug Output窗口的信息(我决定发送到那儿而不是用WAIT WINDOW, 因为后者会干扰Db1Click事件)。这个简单的例子并没有做很多事, 但是可以推测出可能: 跳转到报表的另一个节或或另一个不同的报表, 运行VFP表单, 根据被右击的特别报表对象提供了快捷键菜单以显示不同的选项, 支持书签, 等等。

你也可能注意到工具栏包含有看似有趣的按钮。这些是用于在报表内查找文本。我将在下月讨论这个主题。

总结

通过输出报表的内容到表或游标, DBFListener为我们提供了报表里每一个重绘对象的信息, 它可以被用于许多用途。我希望你正开始看到VFP9 ReportListener类为我们所提供的难以置信的可能。

从 Visual FoxPro 中访问 Hotmail 和 MSN 帐号

原著: Anatoliy Mogylevets

翻译: Fbilo

在某些情况下，用 Visual Foxpro 代码发送和接收 Hotmail 消息是可能的。在这篇文章中，Anatoliy Mogylevets 讲解了一些类，这些类使用了微软 ServerXMLHTTP 和 XMLDOM 对象来实现对 Hotmail 服务器的 WebDAV 访问。你还可以学到一些 HTTP 请求的奥秘，包括请求的头。

这篇文章中讲到的内容对系统的要求是 Windows XP/2000/2003、Visual FoxPro 8.0 或者 9.0、Microsoft ServerXMLHTTP 和 XMLDOM COM 对象，以及一个 Hotmail 或者 MSN 帐号。注意这里讲述的功能也许不能用于免费的 Hotmail 帐号。

介绍

任何人都可以通过使用标准的 MSN 主页访问一个 Hotmail 帐号，该主页上有着一个非常直观的接口。但是，你能够在在一个 FoxPro 应用程序中自动化或者集成这种访问吗？你当然可以在一个表单上寄宿一个浏览器控件了，可要发送一封新的电子邮件或者一个消息的话，你还是得在浏览器窗口中点击一系列的步骤。

Outlook Express 能够集成 Hotmail 帐号，而且从表面上看起来，OE 处理 Hotmail 的邮件帐号看起来跟处理普通的 SMTP/POP3 邮件帐号没什么不同。因此，我们可以推断出肯定存在着一种让非浏览器应用程序与 Hotmail 服务器交互的途径。如果你的机器上装有一个数据包嗅探器（sniffing）应用程序的话，当通过 Outlook Express 同步一个 Hotmail 帐号的时候就能够很容易的发现正在发生什么事情。

一个数据包嗅探器会捕捉到所有离开或者进入你的计算机的 TCP 包，并能够帮助你理解 HTTP、FTP、SMTP、POP3 以及其他协议。我自己的机器上已经装有一个数据包嗅探器，所以我把它设置为 TCP 包捕捉模式，然后按下在 Outlook Express 工具栏上的“发送/接收”按钮。图 1 显示了这么做的结果。

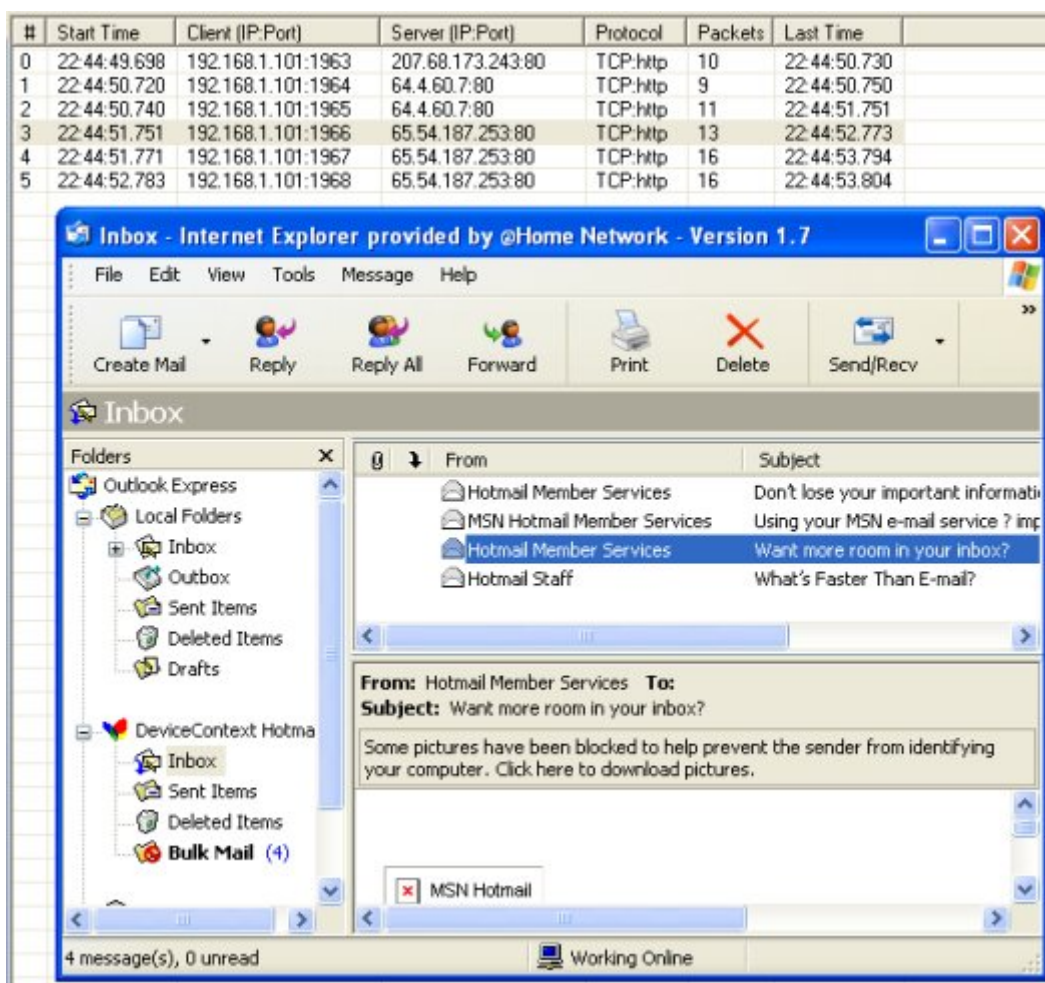


图 1、Outlook Express 与 Hotmail 服务器交换 TCP 包

从图 1 中你可以看到，本地计算机连续的连接到三台远程服务器上的 80 端口。这是第一条线索——这是在使用 HTTP 协议。通过这种途径，你可以用“nslookup”命令来查找这些 IP 地址的域名。让我们看看这些 TCP 包中的内容。

图 2 显示这是本地服务器发送给位于 services.msn.com 的一台远程服务器的一个 HTTP 请求 (Request)。

#	Start Time	Client (IP:Port)	Server (IP:Port)	Time Offset	Pac...	Dat...	Data
0	22:44:49.698	192.168.1.101:1963	207.68.173.243	22:44:49.698	62	0	
1	22:44:50.720	192.168.1.101:1964	64.4.60.7:80	22:44:50.720	62	0	
2	22:44:50.740	192.168.1.101:1965	64.4.60.7:80	22:44:50.720	54	0	
3	22:44:51.751	192.168.1.101:1966	65.54.187.253:	22:44:50.720	784	730	PROPFIND /svcs/hotr
4	22:44:51.771	192.168.1.101:1967	65.54.187.253:	22:44:50.720	60	0	
5	22:44:52.783	192.168.1.101:1968	65.54.187.253:	22:44:50.720	500	446	HTTP/1.1 302 Object


```

PROPFIND /svcs/hotmail/httpmail.asp HTTP/1.1
Depth: 0
Content-Type: text/xml
Brief: t
User-Agent: Outlook-Express/6.0 (MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.0.3705; .NET CLR 1.1.4322; TmstmpExt)
Host: services.msn.com
Content-Length: 357
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: mh=MSFT; MC1=V=2&GUID=28966619c0d64d8c9788faca5ec1e5; SPEED=B

<?xml version="1.0"?>
<D:propfind xmlns:D="DAV:" xmlns:h="http://schemas.microsoft.com/hotmail/"
xmlns:hm="urn:schemas:httpmail:">
  <D:prop>
    ..<h:adbar/>
    ..<hm:contacts/>
    ..<hm:inbox/>
    ..<hm:outbox/>
    ..<hm:sendmsg/>
    ..<hm:sentitems/>
    ..<hm:deleteditems/>
    ..<hm:drafts/>
    ..<hm:msgfolderroot/>
    ..<h:mxpoll/>
    ..<h:sig/>
  </D:prop>
</D:propfind>

```

图 2、Outlook Express 使用 HTTP 请求来与 Hotmail 服务器进行通讯

你也许听说过 HTTP GET 和 POST 请求，但什么是 PROPFIND 请求？而且为什么这个请求的主题是 XML 格式的呢？

在 Google 上搜索一下 PROPFIND 会很快发现它与 WebDAV（Web Distributed Authoring and Versioning，分布式 Web 验证和版本）协议有关。那么现在我就知道了 Outlook Hotmail 客户端是建立在什么基础之上的了。WebDAV 协议是对 HTTP/1.1 协议的一个扩展，它允许你读取和修改远程数据，或者去建立可写的 Web 应用程序。

WebDAV 客户端（Outlook Express）向 WebDAV 服务器（Hotmail 服务器）发送 HTTP 请求来试图在服务器上触发一个活动。客户端会对请求中的 HTTP 头(header)和 XML 格式的主体（body）中的信息进行编码。

服务器对命令进行解码，并执行被请求的操作——比如清空被删除了的消息——然后向客户端发送一个应答。一个典型的应答如图 3 所示。

```

HTTP/1.1 207 Multi-Status
Connection: close
Date: Fri, 11 Mar 2005 04:25:06 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
P3P:CP="BUS CUR CONo FIN IVDo ONL OUR PHY SAMo TELo"
Content-Length: 406
Expires: Mon, 01 Jan 1999 00:00:00 GMT
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/xml
X-Dav-Error: 200 No error
HMServer: H: BAY18-DAV10.phx.gbl V: WIN2K3 10.05.1504.0009 1 D: Mar 4 2005
17:02:52 S: 0

<?xml version="1.0" encoding="Windows-1252"?>
<D:multistatus xmlns:D="DAV:" xmlns:m="urn:schemas:mailheader:"
xmlns:hm="urn:schemas:httpmail:" xmlns:c="urn:schemas:contacts:"
xmlns:h="http://schemas.microsoft.com/hotmail/">
<D:response>
<D:href>http://bay18.oe.hotmail.com/cgi-
oin/hmdata/devicecontext@hotmail.com/folders/trAsH/</D:href>
<D:status>HTTP/1.1 200 OK</D:status>
</D:response>
</D:multistatus>

```

图 3、Hotmail 服务器通过发送一个由头和 XML 主体组成的应答来回答客户端的 HTTP 请求。

从应答中，WebDAV 客户端可以搞清楚请求的操作是否已经成功。

在通过我的数据包嗅探器观察了一段时间的用 Outlook Express 访问我的 Hotmail 帐号的过程以后，我很快有了足够的捕捉到的数据包，可以用来分析客户端和服务端之间的交互，并将之替换成 Visual FoxPro 代码。

在这里，我必须声明，Hotmail 的 WebDAV 访问是不使用证书的。《时代·PC 世界》杂志最近估计活跃的使用 WebDAV 功能来访问 Hotmail 帐号的人数有 940 万。目前，这个协议还是不需要证书的，但微软看起来正准备把这个功能做成只有付费的 Hotmail 用户才能使用，以免该功能被垃圾邮件发送者所滥用。一般来说，如果你可以通过 Outlook 访问你的帐号，你就可以用这篇文章中的 VFP 代码来做到同样的事情。

关于 HTTP 请求的一些话

在 Internet 上，一个请求是由客户端通过一个 TCP/IP 连接发送给服务器的一个消息。HTTP 请求的第一行包括动词（也叫做方法）、Internet 资源的标志符（URL）、以及该 HTTP 协议的版本。

动词的列表包括 GET、POST、HEAD、PUT、TRACE、以及定义在 HTTP/1.1 规范中的其它词。WebDAV 自己又添加了一套新的动词，包括 PROPFIND、PROPPATCH、MOVE、DELETE、以及其他动词。在 HTTP 请求中的动词表示要在由 URL 指定的资源上执行的操作。

HTTP 请求的下一部分包含着一些头部字段，通常称之为“请求头”。客户端使用它们来将关于请求和客户端自身的信息传递给服务器。

可选的主体 (body) 部分结束这个请求。主体 (body) 可以包括客户端 **Post** 给服务器的一个文件、或者一块格式化过的数据 (XML、HTML 等等)。象你可能会想到的那样, 主体的体积可能会变得非常大。这里是一个例子。当我打开我的浏览器的时候, 它会向 **Google** 发送一个 **HTTP** 请求, 因为我浏览器中设置的主页是 **Google**。

从图 4 你可以看到使用了动词 **GET**, 而 **URL** 是在 **www.google.ca** 上的 **“/”**。通过读取这个请求头中的 **User-Agent** 部分, 服务器就能知道客户端所使用的哪种操作系统和浏览器。某些 **Web** 服务器会根据 **User-Agent** 的值来生成与客户端的硬件和软件相适应的应答。这就是为什么从 **Pocket PC** 上看到的 **Google** 搜索页跟在台式机上看到的是不同的。

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-ca
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
.NET CLR 1.0.3705; .NET CLR 1.1.4322; Google-TR-1)
Host: www.google.ca
Connection: Keep-Alive
Cookie: PREF=ID=aeb7dcce3b7f1dae:LD=en:TM=1110244807:LM=
1110244807:S=V_HK2cPUAxKAD2Fm
```

图 4、一个浏览器用来返回 **Google** 搜索页所发送的 **HTTP** 请求

注意, 这个请求是没有主体块的, 因为 **GET** 请求被认为是不需要有主体的。**Google** 通过将它的搜索页的 **HTML** 代码回送给浏览器来对这个请求提供服务。

有几种在 **Visual FoxPro** 代码中打开和直接或者间接发送 **HTTP** 请求的办法: 使用 **WinINET API** 库、**WinHTTP**、**Winsock**、以及 **URL** 标记(Moniker); 使用 **Winsock** 控件; 或者使用一个第三方库或者控件。至于这篇文章中讲到的 **WebmailClient** 类, 我采用了微软的 **ServerXMLHTTP COM** 对象。更准确的说, 我找不到可以代替这个 **COM** 对象的任何其它东西。

你可以试试使用 **MSXML2.XMLHTTP** 对象来代替, 不过这样的话, 你的 **FoxPro** 代码将必须自己去实现(Implement)服务器部分(重定向和验证)。微软的 **ServerXMLHTTP** 对象包含在 **Microsoft XML Parser(MSXML) 3.0** 及以上版本中的。

使用微软 **ServerXMLHTTP** 和 **XMLDOM** 对象

ServerXMLHTTP 对象是向远程服务器发送各种 **HTTP** 请求和接收应答的理想选择。只用它, 你就可以在 **Visual FoxPro** 中建立一个 **Hotmail** 客户端的数据交换部分。缺点是 **Windows 95/98/Me** 不支持这个对象。

这里是如何建立一个 **ServerXMLHTTP** 对象实例的办法:

```
oHttp = CreateObject("MSXML2.ServerXMLHTTP")
```

该对象支持 XML 数据交换,而且它的名字里有个“server”——并且它的确工作得象一个 Server:跟 Hotmail 建立一个连接、重定向、建立 cookies、并对数据加密和解密。

另一个 Microsoft 对象 XMLDOM 是用来分析从 Hotmail 服务器上接收到的 XML 数据的。

封装 Hotmail WebDAV 功能的 Visual FoxPro 类

在这篇文章里讲述的类库包含四个由 VFP 的 Session 类继承而来的子类（见表 1）。

表 1、封装 Hotmail WebDAV 功能的 Visual FoxPro 类库

类	说明
WebmailClient	连接到 Hotmail 服务器,并请求一个文件夹的列表。它有一些发送 HTTP 请求的方法,知道怎样去清除放有已删除了的数据项的文件夹（老外说话罗嗦,不就是废件箱嘛!）,知道怎样去发送一封 Email。它的 FOLDERS 属性是一个 WebmailFolder 对象的集合。
WebmailFolder	代表一个 Hotmail 文件夹。这个类有一个方法,可以用来返回在这个文件夹中一个所有邮件的信封消息的列表。MESSAGES 属性是一个 WebmailMessage 对象的集合。
WebmailMessage	代表一个 Hotmail 邮件。可以把邮件返回、删除或者移动到另一个文件夹中去。HEADERS 属性是一个邮件头的集合。为了简化设计,这个对象不会分析邮件的主体部分或者附件。
MessageHeader	代表一个邮件的头部,比如 From、To、Cc、Content-Type、Return-Path 等等内容。

连接到邮件服务器

到目前为止,为了说得简单点,我一直使用“Hotmail”这个词来表示 Hotmail 和 MSN 的 email 帐号。在这两者之间唯一的区别是邮件服务器的 URL 地址:

```
#DEFINE URL_MSN http://oe.msn.msnmail.hotmail.com/cgibin/hmdata
#DEFINE URL_HOTMAIL http://services.msn.com/svcs/hotmail/httpmail.asp
```

为了打开与服务器的通讯,客户端会发送一个 PROPFIND HTTP 请求和一个 XML 主体,主体中包含着一个用于服务器的应答的模板。图 5 显示了一个连接到 Hotmail 帐号的请求的内容:

```

PROPFIND /svcs/hotmail/httpmail.asp HTTP/1.1
Depth: 0
Content-Type: text/xml
Brief: t
User-Agent: Outlook-Express/6.0 (MSIE 6.0; Windows NT 5.1; SV1; .NET
CLR 1.0.3705; .NET CLR 1.1.4322; TmstmpExt)
Host: services.msn.com
Content-Length: 357
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: mh=MSFT; MC1=V=2&GUID=28966619c0d64d8c9788faca5ec1ebe5; SPEED=B

<?xml version="1.0"?>
<D:propfind xmlns:D="DAV:"
xmlns:h="http://schemas.microsoft.com/hotmail/"
xmlns:hm="urn:schemas:httpmail:">
  .<D:prop>
    ..<h:adbar/>
    ..<hm:contacts/>
    ..<hm:inbox/>
    ..<hm:outbox/>
    ..<hm:sendmsg/>
    ..<hm:sentitems/>
    ..<hm:deleteditems/>
    ..<hm:drafts/>
    ..<hm:msgfolderroot/>
    ..<h:mxpoll/>
    ..<h:sig/>
  .</D:prop>
</D:propfind>

```

图 5、当初始化一个连接时，Hotmail 客户端发送给 Hotmail 服务器的第一个 HTTP 请求

这里是在这个 HTTP 请求中涉及到的 WebmailClient 类的几个方法中的 FoxPro 代码：

```

THIS.http = CREATEOBJECT("MSXML2.ServerXMLHTTP")
THIS.xmlGateway = CREATEOBJECT("Microsoft.XMLDOM")
* 省略后面的代码

PROCEDURE ConnectTo(cEmail, cPwd)
  LOCAL cServer
  THIS.email = LOWER(ALLTRIM(m.cEmail))
  THIS.pwd = LOWER(ALLTRIM(m.cPwd))

  DO CASE
    CASE "@hotmail.com" $ THIS.email
      cServer = URL_HOTMAIL
    CASE "@msn.com" $ cEmail
      cServer = URL_MSN
    OTHERWISE
      * email 帐号无效

```

```

        RETURN .F.

    ENDCASE

    * 连接到 Hotmail 服务器...

    THIS.SendPropfind(cServer, XML_GATEWAY)

    IF THIS.response != RESPONSE_OK
        * 连接失败

        RETURN .F.
    ENDIF

    * 把应答交给 xml 分析器

    THIS.xmlGateway.Load(THIS.http.ResponseXml)

    RETURN .T.

PROCEDURE SendPropfind(href, xml)
    THIS.SendHttpRequest(PROPFIND, m.href,;
        "Cache-Control: no-cache;" +;
        "Content-Type: text/xml", m.xml)

PROCEDURE SendHttpRequest
    LPARAMETERS cMethod, href,cHeaders, cBody)
    STORE "" TO THIS.response, THIS.statustext,;
        THIS.responseheaders

    LOCAL ex As Exception
    WITH THIS.http
        TRY
            .Open(cMethod, m.href, 0,;
                THIS.email, THIS.pwd)
        CATCH TO ex
    ENDTRY

    IF VARTYPE(m.ex) = "O"
        THIS.errorno = ex.ErrorNo
        THIS.errormessage = ex.Message
    RETURN .F.

```

```

ENDIF

THIS.AddHeaders(cHeaders)

TRY
    .Send(m.cBody)
    THIS.response = SUBSTR(
        .GetResponseHeader(X_DAV_ERROR),1,3)
    THIS.statustext = THIS.http.StatusText
    THIS.responseheaders =;
        THIS.http.GetAllResponseHeaders()
CATCH
ENDTRY

IF VARTYPE(m.ex) = "O"
    THIS.errorno = ex.ErrorNo
    THIS.errormessage = ex.Message
    RETURN .F.
ENDIF
ENDWITH

```

```

HTTP/1.1 207 Multi-Status
Connection: close
Date: Sun, 13 Mar 2005 23:02:45 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
P3P:CP="BUS CUR CONo FIN IVDo ONL OUR PHY SAMo TELo"
Content-Length: 2429
Expires: Mon, 01 Jan 1999 00:00:00 GMT
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/xml
X-Dav-Error: 200 No error
HMServer: H: BAY18-DAV9.phx.gbl V: WIN2K3 10.05.1504.0009 i D:
Mar 4 2005 17:02:52 S: 0

```

图 6、Hotmail 服务器应答的头部字段们。注意其中的 X-Dav-Error 头。


```

<?xml version="1.0" encoding="Windows-1252"?>
<D:multistatus xmlns:D="DAV:" xmlns:m="urn:schemas:mailheader:"
xmlns:hm="urn:schemas:httpmail:" xmlns:c="urn:schemas:contacts:"
xmlns:h="http://schemas.microsoft.com/hotmail/">
<D:response>
<D:href>http://bay18.oe.hotmail.com/cgi-
bin/hndata/devicecontext@hotmail.com/</D:href>
<D:propstat>
<D:prop>
<h:adbar>AdPane=Off*AdSvr=H*Other=GetAd?PG=HOTOEB?SC=LG?TF=_BLANK?HM=
045746405a5e4a120a57012c524e1003496b6851102852153f4b13595e5054530b6c044
f6f66</h:adbar>
<hm:contacts>http://contacts.msn.com/cgi-
bin/hndata/devicecontext@hotmail.com/abdata/</hm:contacts>
<hm:inbox>http://bay18.oe.hotmail.com/cgi-
bin/hndata/devicecontext@hotmail.com/folders/ACTIVE/</hm:inbox>
<hm:sendmsg>http://bay18.oe.hotmail.com/cgi-
bin/hndata/devicecontext@hotmail.com/sendmsg/</hm:sendmsg>
<hm:sentitems>http://bay18.oe.hotmail.com/cgi-
bin/hndata/devicecontext@hotmail.com/folders/s&VeD/</hm:sentitems>
<hm:deleteditems>http://bay18.oe.hotmail.com/cgi-
bin/hndata/devicecontext@hotmail.com/folders/tr&asH/</hm:deleteditems>
<hm:drafts>http://bay18.oe.hotmail.com/cgi-bin/

```

图 7、Hotmail 服务器应答的 XML 主体。这个特定的应答返回的是服务器和 Hotmail 帐号的主 URL。

Hotmail 服务器用头部（见图 6）和一些 XML 数据（见图 7）来应答。注意其中的 X-Dav-Error 头的值是“200 No Error”。所有成功的 Hotmail 事务都会从服务器上返回这个值。

你可以从这段代码中看到，THIS.xmlGateway(XML 分析器)会加载这个应答的 XML 部分，其中包含着对一些特殊文件夹——包括收件箱、已发送邮件箱、废件箱、以及其它东西——的连接。稍后我们将在其它一些 HTTP 请求中使用其中的一些连接。

请求文件夹列表

同一个 PROPFIND 动词被用来返回 Hotmail 帐号中的文件夹的列表。请求的 XML 主体部分事实上是文件夹属性们的列表，它为 Hotmail 服务器提供了一个模板（见图 8）。服务器会用每个文件夹的属性来填充到这个模板中去，并在应答的主体部分中返回它们。这里是用于处理这个请求的

WebmailClient 代码：

```

THIS.xmlFolders = CREATEOBJECT(
    "Microsoft.XMLDOM")

THIS.folders = CREATEOBJECT("Collection")

* 这里的代码忽略

PROCEDURE GetFolders

```


* 读取可用的文件夹，并生成一个 WebmailFolder 对象的集合

= ClearCollection(THIS.folders)

LOCAL cFoldersHref, xmlFolder,;

nFolderCount, nFolderIndex, oFolder

cFoldersHref = THIS.xmlGateway.;

SelectSingleNode(MSGFOLDERROOT_NODE).Text

THIS.SendPropfind(m.cFoldersHref, XML_FOLDERS)

IF THIS.response <> RESPONSE_OK

RETURN .F.

ENDIF

THIS.xmlFolders.Load(THIS.http.ResponseXml)

nFolderCount = THIS.xmlFolders.;

SelectSingleNode(MULTISTATUS).;

ChildNodes.Length

FOR nFolderIndex=0 TO nFolderCount-1

xmlFolder = THIS.xmlFolders.;

SelectSingleNode(MULTISTATUS).;

ChildNodes(nFolderIndex)

oFolder = CREATEOBJECT("WebmailFolder",;

THIS, xmlFolder)

THIS.folders.Add(oFolder,;

oFolder.foldername)

oFolder=NULL

NEXT

```

PROPFIND /cgi-bin/hmdata/devicecontext@hotmail.com/folders/ HTTP/1.1
Depth: 1,noroot
Content-Type: text/xml
Brief: t
X-Timestamp: folders=1055250907,ACTIVE=1055250923
Accept-CharSet: Windows-1252
User-Agent: Outlook-Express/6.0 (MSIE 6.0; Windows NT 5.1; SV1; .NET
CLR 1.0.3705; .NET CLR 1.1.4322; TmstmpExt)
Host: bay18.oe.hotmail.com
Content-Length: 265
Connection: Keep-Alive
Cache-Control: no-cache

<?xml version="1.0"?>
<D:propfind xmlns:D="DAV:" xmlns:hm="urn:schemas:httpmail:">
  <D:prop>
    ..<D:isfolder/>
    ..<D:displayname/>
    ..<hm:special/>
    ..<D:hassubs/>
    ..<D:nosubs/>
    ..<hm:unreadcount/>
    ..<D:visiblecount/>
    ..<hm:special/>
  </D:prop>
</D:propfind>

```

图 8、Hotmail 客户端发送这个 HTTP 请求来获得 Hotmail 帐号中文件夹的列表

在返回的消息中，每个文件夹都通过它的名字、文件夹的 URL、存储有邮件的数量、以及其它一些参数来描述。目前 Hotmail 服务器仅支持一层的文件夹，所以 HASSUBS 和 NOSUBS 属性是没用的。WebmailClient 对象使用 XML 分析器来生成它的 FOLDERS 集合——这是一个 WebmailFolder 对象的集合。下面的代码显示的是为 WebmailFolder 类定义了的属性：

```

DEFINE CLASS WebmailFolder As Session

  webmail=Null

  href=""

  foldername=""

  specialfolder=.F.

  messagecount=0

  unreadcount=0

  messages=0 && collection

PROCEDURE Init(oWebmail, xmlFolder)
  * 这里的代码忽略
ENDDFINE

```

读取存储在一个文件夹中邮件们的信封

WebmailFolder 类包含着一个能够读取邮件的信封的 **GetMessageEnvelopes**。由于它不会把每个邮件的全部内容都读下来，所以它所花的时间相对较短。读信封的 **HTTP** 请求同样使用 **PROPFIND** 动词，并且请求的 **XML** 主体部分是一个用于邮件信封的属性列表的模板（见图 9）。这里是 **WebmailFolder** 类的 **GetMessageEnvelopes** 方法：

```
PROCEDURE GetMessageEnvelopes
= ClearCollection(THIS.messages)

THIS.webmail.SendPropfind(THIS.href,;
    XML_MESSAGES)
IF THIS.webmail.response != RESPONSE_OK
    RETURN
ENDIF

LOCAL xmlMessages, xmlMessage, nMessageCount,;
    nMessageIndex
xmlMessages = CreateObject("Microsoft.XMLDOM")
WITH xmlMessages
    .Load(THIS.webmail.http.ResponseXml)
    nMessageCount = .SelectSingleNode(;
        MULTISTATUS).ChildNodes.Length

    FOR nMessageIndex=0 TO nMessageCount-1
        xmlMessage = .SelectSingleNode(;
            MULTISTATUS).ChildNodes(;
                nMessageIndex)

        oMessage = CREATEOBJ("WebmailMessage",;
            THIS, xmlMessage)
        THIS.messages.Add(oMessage,;
            oMessage.msgid)
        oMessage = Null
    NEXT
ENDWITH
```

```

PROPFIND /cgi-bin/hndata/devicecontext@hotmail.com/folders/ACTIVE/
HTTP/1.1
Cache-Control: no-cache
Content-Type: text/xml
Accept-Language: en-ca
Content-Length: 286
Accept: */*
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Host: bay18.oe.hotmail.com
Connection: Keep-Alive

<?xml version="1.0"?>
<D:propfind xmlns:D="DAV:" xmlns:hm="urn:schemas:httpmail:"
xmlns:m="urn:schemas:mailheader:">
  .<D:prop>
    ..<D:isfolder/>
    ..<hm:read/>
    ..<m:hasattachment/>
    ..<m:to/>
    ..<m:from/>
    ..<m:subject/>
    ..<m:date/>
    ..<D:getcontentlength/>
  .</D:prop>
</D:propfind>

```

图 9、Hotmail 客户端发送这个 HTTP 请求来获得在 Hotmail 帐号收件箱文件夹中邮件信封的列表

服务器以头部、一个放有邮件信封列表的 XML 主体来作为应答。WebmailFolder 对象使用 XML 分析器来生成它的 MESSAGES 集合，这是一个 WebmailMessage 对象的集合。

```

DEFINE CLASS WebmailMessage As Session

```

```

  folder=NULL

```

```

  href=""

```

```

  msgid=""

```

```

  readstatus=.F.

```

```

  sender=""

```

```

  recipient=""

```

```

  subject=""

```

```

  created=""

```

```

  contentlength=0

```

```

  rawcontent=""

```

```

  rawheaders=""

```

```

  headers=0

```

```

PROCEDURE Init(oFolder, xmlMessage)

```

```

  * 这里的代码忽略

```

```

ENDDEFINE

```

返回一封邮件

WebmailMessage 类包含的 **GetMessageContent** 方法能返回整封邮件。跟前面的那些请求不同，这个方法使用 **GET** 动词并且没有 **XML** 主体。邮件的 **URL** 就放在请求中第一行动词的后面，而 **Hotmail** 服务器的应答的主体则包含着未加工的邮件内容。这里是 **WebmailMessage** 类的 **GetMessageContent** 方法：

```
PROCEDURE GetMessageContent
= ClearCollection(THIS.headers)
LOCAL cResponseText, nPos

THIS.folder.webmail.SendHttpRequest("GET",;
    THIS.href, "", "")

IF THIS.folder.webmail.response = RESPONSE_OK
    cResponseText =;
        THIS.folder.webmail.http.ResponseText
    nPos = AT(dCRLF, cResponseText)
    THIS.rawcontent = SUBSTR(cResponseText,;
        nPos+4)
    THIS.rawheaders = SUBSTR(cResponseText,;
        1, nPos)
    THIS.ParseHeaders
ELSE
    STORE "" TO THIS.rawheaders,;
    THIS.rawcontent
ENDIF
```

在 **Outlook Express** 中，你可以通过在一个邮件上单击右键，然后选择“属性|详细信息|邮件来源”来看到邮件未处理过的内容。

未处理过的邮件顶上的部分包含各种邮件参数。下面的邮件主体部分也许不会象图 10 中显示的那么简单。主体通常都会包含以不同途径编码的几个部分（浅白的文本、或者是 **HTML** 部分、还有附件）。怎么去分析 email 邮件的所有类别应该是一个独立的 **FoxPro** 类的任务，在这个示例里，为了简化问题，**WebmailFolder** 类仅分析邮件的主体的一部分。它把邮件分成 **RAWHEADERS** 和 **RAWCONTENT** 两个部分，然后从 **RAWHEADERS** 来生成邮件头部的集合。

删除一封邮件

Hotmail 的删除其实就是把邮件移到废件箱里罢了。所以 `WebmailMessage.DeleteMessage` 方法不过是对同一个类的 `MoveMessage` 方法的一个封装。HTTP 请求的最上面一行包括 **MOVE** 动词和要被移动的邮件的 **URL**(见图 11)。其中的 **Destination** 头部包含着目标文件夹的 **URL**——在这里, 这个目标文件夹就是指废件箱。此外, 还包含有 **Allow-Rename** 头部。注意这个请求是不需要一个 **XML** 主体的, 因为所需的信息已经包含在顶上一行和头部中了。

```
Received: from ...
Message-ID: ...
Date: Thu, 10 Mar 2005 11:43:17 -0600
From: ...
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0;)
Gecko/20031205 Thunderbird/0.4
X-Accept-Language: en-us, en
MIME-Version: 1.0
To: ...
Subject: Test Message
Content-Type: text/plain; charset=us-ascii; format=flowed
Content-Transfer-Encoding: 7bit

Hi,
This is a test message. Please do not reply.
```

图 10、一个未处理邮件内容的例子

```
MOVE /cgi-
bin/hmdata/devicecontext@hotmail.com/folders/ACTIVE/MSG1110229512.
47 HTTP/1.1
Destination: http://bay18.oe.hotmail.com/cgi-
bin/hmdata/devicecontext@hotmail.com/folders/trash/MSG1110229512.4
7
Allow-Rename: t
User-Agent: Outlook-Express/6.0 (MSIE 6.0; Windows NT 5.1; SV1;
.NET CLR 1.0.3705; .NET CLR 1.1.4322; TmstmpExt)
Host: bay18.oe.hotmail.com
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

图 11、一个 Hotmail 客户端发送给 Hotmail 服务器的要求从收件箱中删除一封邮件的 HTTP 请求。注意这个请求中是没有 XML 主体的。

```
PROCEDURE DeleteMessage
LOCAL href
href = THIS.folder.webmail.GetTrashHref()
RETURN THIS.MoveMessage(m.href)

PROCEDURE MoveMessage(cDstHref)
```

```

LOCAL cHeaders
cHeaders = "Destination: " + cDstHref
        THIS.msgid + ";Allow-Rename: t"

WITH THIS.folder.webmail
        .SendHttpRequest("MOVE",;
                THIS.href, cHeaders, "")
        RETURN (.response = RESPONSE_OK)
ENDWITH

```

一个要求删除多封邮件的请求可以通过一系列的 **MOVE** 请求或者单个 **BMOVE** 请求来实现。所有对多个数据项操作的 **WebDAV** 动词都是以“**B**”来开头的。**BMOVE** 请求有一个包含着信件 ID 的 XML 部分（见图 12）。

```

BMOVE /cgi-bin/hmdata/devicecontext@hotmail.com/folders/ACTIVE/
HTTP/1.1
Destination: http://bay18.oe.hotmail.com/cgi-
bin/hmdata/devicecontext@hotmail.com/folders/trash/
Allow-Rename: t
Content-Type: text/xml
User-Agent: Outlook-Express/6.0 (MSIE 6.0; Windows NT 5.1; SV1;
.NET CLR 1.0.3705; .NET CLR 1.1.4322; TmstampExt)
Host: bay18.oe.hotmail.com
Content-Length: 231
Connection: Keep-Alive
Cache-Control: no-cache
<?xml version="1.0"?>
<D:move xmlns:D="DAV:">
  .<D:target>
    ..<D:href>MSG110229512.47</D:href>
    ..<D:href>MSG1109216161.5</D:href>
    ..<D:href>MSG1108697970.14</D:href>
    ..<D:href>MSG1107840393.27</D:href>
  .</D:target>
.</D:move>

```

图 12、一个 Hotmail 客户端发送给 Hotmail 服务器的从收件箱中删除多份邮件的 HTTP 请求。XML 主体包含着邮件 ID。

其它操作

另外一个“**B**”动词——**BDELETE**——是用于从废件箱里清除所有已被删除了的邮件的。请求的头上行包含着废件箱文件夹的 URL，而 XML 主体则是信件 ID 的列表。你可以在这篇文章的下载文件中 **WebmailClient** 类的 **EmptyTrashFolder** 方法中找到清空废件箱的代码。

另两个动词 **COPY** 和 **BCOPY** 可以被用来在文件夹之间拷贝邮件。

发送一封 Hotmail 邮件

微软声称，即使前段时间使用的“每天一百封”的发送邮件限制也没有改善 Hotmail 被垃圾邮件发送者所滥用的状况。因此，**thou shalt not spam!**(晕~~翻译不出来了)记住这一点，我将解释怎样发送 Hotmail 邮件。如果你对简单邮件传递协议（Simple Mail Transfer Protocol,SMTP）很熟悉，你可能会注意到 Hotmail 对邮件要求的格式与 SMPT 很相似。

不过，它们还是有一些区别的。在一个 SMTP 服务器上，信件必须一步一步的被发送（见表 2）。首先，你发送出一个 MAIL 命令，接着等待“250 OK”应答。然后，你发送一个也需要得到一个“250 OK”应答的 RCPT TO 命令来继续。再然后，DATA 命令被发送，如此等等...

表 2、发送一封 SMPT 邮件

元素	格式
首先来的是 MAIL 命令	MAIL FROM: <sender@msn.com>
	RCPT TO: <recipient@somewhere.com>
然后是邮件的头	From: <sender@msn.com>
	To: <recipient@somewhere.com>
	Subject: test message
	Date: Fri, 11 Mar 2005 15:23:00 -0500
	MIME-Version: 1.0
	Content-Type: text/plain; charset=us-ascii; format=flowed
	Content-Transfer-Encoding: 7bit X-Priority: 3
	X-MSMail-Priority: Normal
	X-Mailer: Microsoft Outlook Express 6.00.2900.2527
	X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2527
最后是邮件的主体	“测试邮件，请不要回复。”

与 SMTP 不同，Hotmail 服务器要求整个邮件被集中起来，并在单个 POST HTTP 请求中被发送给 Hotmail 服务器。这里是 WebmailClient.SendEmail 方法的代码：

```
PROCEDURE SendEmail
LPARAMETERS cRecipient, cSubject, cBody, ISaveSent
LOCAL href, cMessage, cHeaders

* 在 Hotmail 上的 SendMessage 的 URL
href = THIS.GetSendHref()

* 将邮件整合起来
```



```

cMessage = 'MAIL FROM:<' +;
    THIS.email + '>' + CRLF +;
    'RCPT TO:<' + cRecipient + '>' + CRLF +;
    " + CRLF +;
    'From: <' + THIS.email + '>' + CRLF +;
    'To: <' + m.cRecipient + '>' + CRLF +;
    'Subject: ' + m.cSubject + CRLF +;
    'Date: ' + TTOC(DATETIME()) + CRLF +;
    'MIME-Version: 1.0' + CRLF +;
    'Content-Type: text/plain;' +;
    'charset="Windows-1252"' + CRLF +;
    'X-Mailer: ' + VERSION() + CRLF +;
    " + CRLF +;
    cBody

* 附加的 HTTP 头
cHeaders = "Content-Type: message/rfc821;" +;
    "SAVEINSENT: " + IIF(m.ISaveSent, "t", "f")

* 发送邮件
THIS.SendHttpRequest("POST", m.href,;
    m.cHeaders, m.cMessage)

```

注意，在这个例子中的邮件整合例程被简化了，并且生成的是纯文本的邮件而不带任何附件的。

总结

通过观察一个数据包嗅探器，你可以判定其它基于 HTTP 的应用程序所用的协议和消息，然后用标准的微软 HTTP 和 XML 对象来模仿这些消息。你不仅可以使使用这种办法来访问你的 Hotmail 帐号（尽管它可能必须是一个付费的帐号），但你可以使用同样的原理来给你的应用程序增加更多的 Internet 功能。

下载：504ANATOLIY.ZIP

来自 VFP 开发团队的 TIPS

原著：微软 VFP 开发团队

翻译：LQL.NET

这个月来自 VFP 开发团队的 TIPS 向你展示了 VFP9 报表方面的一些新功能，这些功能在你现在买的 VFP9 中均已兑现。

ReportListener 事件和报表调试

下面的代码向你演示了如何将你自己的代码挂到 ReportListener 事件并在报表运行时触发。（这个程序在下载文件中的 IntroListener.prg 里）

你运行这段代码，会要你选择一个报表，然后你看着屏幕，可以在 WAIT WINDOWS 的反应中看到 listener 命令子句集的内容，屏幕也将为每页的 BeforeBand 事件显示输出。（请注意看当我们在页注脚区时它是如何处理传进来的 nBandObjCode 参数的）

然后，你把第一行的 DEBUGGING 常量改为.T.再运行一个至少 10 页的报表，你会看到报表会在第 10 页因 SUSPEND 命令而中断。然后你就可以在正在执行的代码中步进调试，你还可以在里面研究一下一些有意思的变量和对象。你随时可以 RESUME 恢复报表运行。

```
#DEFINE DEBUGGING .F.
CLEAR
LOCAL oListener AS ReportListener
oListener = NEWOBJECT("MyListener")
oListener.ListenerType=1 && Preview
REPORT FORM (GETFILE("FRX")) OBJECT oListener PREVIEW
DEFINE CLASS myListener AS ReportListener
    PROCEDURE AfterReport()
        ACTIVATE SCREEN
        ? PROGRAM()
        ? "***Report is done.***"
    ENDPROC
```

```

PROCEDURE BeforeReport
    ACTIVATE SCREEN
    ? "***Report is starting.***"
    WAIT WINDOW "Let's see REPORT FORM cmd object now."
    lnCount=AMEMBERS(laCmds, THIS.CommandClauses)
    FOR i = 1 TO lnCount
        ? laCmds[m.i], THIS.CommandClauses.&laCmds[m.i]
    ENDFOR
    ?
    WAIT WINDOW
ENDPROC

PROCEDURE BeforeBand(nBandObjCode, nFRXRecno)
    ** Check on Page Footer **
    IF nBandObjCode=7
        ? "Before Page "+TRANSFORM(THIS.PageNo)+ ;
        ? " is about to print."
    ENDIF
ENDPROC

PROCEDURE AfterBand(nBandObjCode, nFRXRecno)
    ** Page Footer **
    IF DEBUGGING AND nBandObjCode=7 AND THIS.PageNo=10
        ? "Suspending..."
        SUSPEND
        ? "Resuming..."
    ENDIF
ENDPROC
ENDDEFINE

```

在报表预览里链接报表

下面的代码样例演示了 VFP9 在报表预览窗口链接报表的能力：

```

*This demonstrates how reports can be chained in the
*preview now.
#define ListenerPrint      0

```

```
#define ListenerPreview    1
#define ListenerXML        4
#define ListenerHTML       5

rep1 = GETFILE("FRX", "Pick the first report")
rep2 = GETFILE("FRX", "Pick the second report")
*See how the range is now respected by the preview:
REPORT FORM (Rep1)  RANGE 1,4 ;
    OBJECT TYPE ListenerPreview NOPAGEEJECT
REPORT FORM (Rep2)  RANGE 1,4 ;
    OBJECT TYPE ListenerPreview NORESET
```

把你的情况告诉我们

2005 年 4 月开始我们对开发人员作了一次在线调查,调查他们正在进行的开发项目和所使用的工具。这是一个让我们更好地了解你的需求的好机会,这会影响今后 VFP 的发展。

在 4 月的第一个星期的某时你可以在 <http://msdn.com/vfoxpro> 看到调查信息。

附件: 504TEAMTIPS.ZIP

正确的设置选项

原著: Andy Kramek & Marcia Akins

翻译: LQL.NET & Fbilo

用一个全局性的解决办法去解决一个局部问题不是明智的。为什么这么说呢？因为当 Andy Kramek 和 Marcia Akins 调用以前写的一些代码的时候发现了问题。一些命令如 SET EXACT 和 SET UDFPARMS 的影响我们大家都知道，而 SET COMPATIBLE 带来的结果却是令人吃惊的。来看看 Andy 最近碰上的麻烦事儿。

Andy: 我想我发现 VFP9 的 BUG 了，但我真的无法相信这个事实因为我的代码看起来明显没错。

Marcia: 啥症状？

Andy: 哦，我调用了我的一个自定义函数用来实现精确匹配查询。这个函数保存了当前工作区（及其他一些相关设置）并在函数返回的时候恢复这个工作区。

Marcia: 打住。没事儿你改什么工作区阿？你用 SEEK() 函数带上“别名”参数不就结了么？

Andy: 哎，SEEK() 函数要检查索引标识是否被指定并且指定的标识必须存在，所以我就用 LOCATE 来代替 SEEK()。你知道，你不能“LOCATE IN”，但这没关系，我保存/恢复一下工作区就是了，这是相关代码：

```
LOCAL InSelect
*** 保存当前工作区
InSelect = SELECT()

*** 恢复工作区
SELECT (InSelect)

RETURN
```

Marcia: 嗯这代码看起来挺结实了，会有啥问题？

Andy: 当我把这段代码放到我的应用程序（这是我 FOXPRO2.6 的代码转到 VFP9 的应用程序）里，它

返回了一个错误的工作区。

Marcia: 啥意思？返回一个错误的工作区？

Andy: 嗯你来看。如果你在工作区 1 调用这段代码，它就返回工作区 2（那是空闲工作区）！咱看图 1，你可以看到变量 **InSelect** 显示为 2，但当前工作区（你可以从数据工作期窗口看到）明明就是 1。我没法从这段一清二白的程序中找到任何问题，我都快疯了。

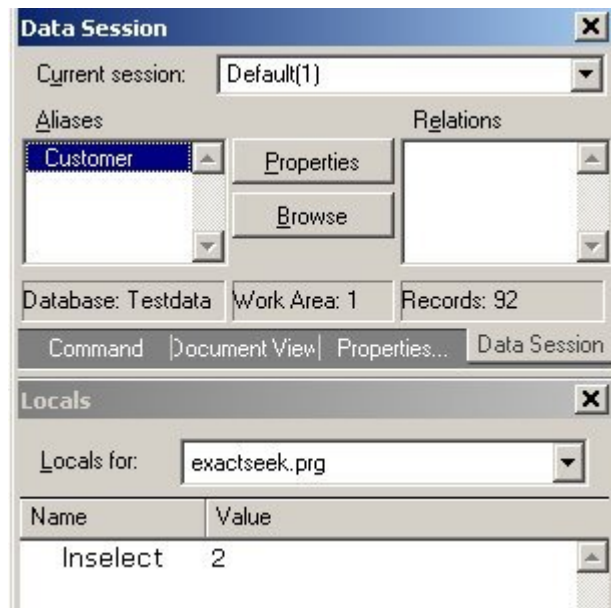


图 1：这是个 BUG！**SELECT()** = 2，但工作区 = 1。

Marcia: 稍安勿躁亲爱的。

Andy: 谢谢，快告诉我怎么办。如果我退出 VFP 重新进入，并在 **COMMAND** 窗口中做同样的事儿，一切 OK，但是我只要一在我的应用程序中运行这段就出现图 1 中的问题，结果就好像 **SELECT()** 返回的工作区值被加了 1。

Marcia: 让我们来分析下。如果你在你的应用程序和 **COMMAND** 窗口运行这段代码的结果不一样，那么就一定是这个应用程序环境中的某些东西发生问题了。

Andy: 嗯是的，那又怎么样呢？我不知道从哪儿着手查。

Marcia: 我通常从看帮助文件开始着手。让我们看看关于 **SELECT()** 函数帮助上是怎么说的，也许 VFP9 有什么改变。

Andy: 帮助文件上说 **SELECT()** 函数会返回当前工作区或最大的空闲工作区。用参数 **0** 获取当前工作区，**1** 返回空闲工作区。

Marcia: 这里没说不带参数会怎么样对吧，你代码里的 **SELECT()** 就没带参数。

Andy: 嗯，不过 **0** 是默认值，这时你不需要给参数。

Marcia: 谁说了？谁说 **0** 是默认值了？我只看到资料上这么说的：

如果 **SET COMPATIBLE** 设置为 **OFF**，**SELECT()** 返回当前工作区，如果 **SET COMPATIBLE** 设为 **ON**，**SELECT()** 返回空闲工作区。

Andy: 哦上帝！我打赌我一定在程序中的某处设了 **SET COMPATIBLE = ON**

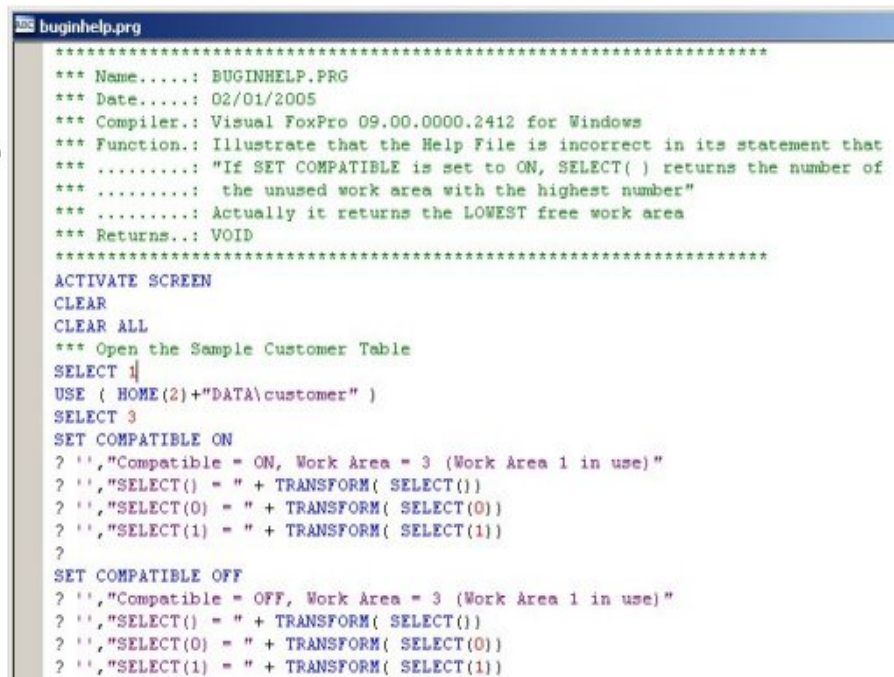
Marcia: 这就能很好地解释你所看到的问题了。现在你在程序里把 **SELECT()** 改成 **SELECT(0)** 试试。

Andy: 马上试！在函数里检查一下设置，我发现 **COMPATIBLE** 值是 **ON**，然后我用代码参考工具查找，上帝！真的在我的启动程序中发现一句 **SET COMPATIBLE DB4**（这等于是 **SET COMPATIBLE ON**）！啊你解决了我的问题亲个亲个——这的确不是 **VFP9** 的 **BUG**。

Marcia: 呵呵很高兴我能帮你找到问题。有趣的是，这里发现一个 **BUG**，不过这个 **BUG** 不是代码里的 **BUG** 而是帮助文件的 **BUG**，看图 2。

```
Compatible = ON, Work Area = 3 (Work Area 1 in use)
SELECT() = 2
SELECT(0) = 3
SELECT(1) = 32767
```

```
Compatible = OFF, Work Area = 3 (Work Area 1 in use)
SELECT() = 3
SELECT(0) = 3
SELECT(1) = 32767
```



```
*****
*** Name.....: BUGINHELP.PRG
*** Date.....: 02/01/2005
*** Compiler.: Visual FoxPro 09.00.0000.2412 for Windows
*** Function.: Illustrate that the Help File is incorrect in its statement that
*** .....: "If SET COMPATIBLE is set to ON, SELECT( ) returns the number of
*** .....: the unused work area with the highest number"
*** .....: Actually it returns the LOWEST free work area
*** Returns..: VOID
*****
ACTIVATE SCREEN
CLEAR
CLEAR ALL
*** Open the Sample Customer Table
SELECT 1
USE ( HOME(2)+"DATA\customer" )
SELECT 3
SET COMPATIBLE ON
? "Compatible = ON, Work Area = 3 (Work Area 1 in use)"
? "SELECT() = " + TRANSFORM( SELECT() )
? "SELECT(0) = " + TRANSFORM( SELECT(0) )
? "SELECT(1) = " + TRANSFORM( SELECT(1) )
?
SET COMPATIBLE OFF
? "Compatible = OFF, Work Area = 3 (Work Area 1 in use)"
? "SELECT() = " + TRANSFORM( SELECT() )
? "SELECT(0) = " + TRANSFORM( SELECT(0) )
? "SELECT(1) = " + TRANSFORM( SELECT(1) )
```

图 2：不是代码里的 BUG 而是帮助文件的 BUG

Andy: 嗯我看到了。当 **compatible** 是 **ON** 的时候，**SELECT()**函数返回最小的空闲工作区号而不是帮助文件上说的返回最大的。不过我想这应该没什么影响，因为这仍然是一个空闲的工作区而你不需要在乎它的确切区号。

Marcia: 从 **VFP6**（可能更早）开始都会有这样的问题，帮助文件也一直就这么说，但是，我觉得在应用程序里不应出现 **SET COMPATIBLE** 这些命令，如果要用，你必须小心、严谨地处理好它。

Andy: 为什么这么说？

Marcia: 因为你碰到的这个问题就是一个典型的“用全局的办法来解决局部问题”例子。

Andy: 你说的“全局的办法”是什么意思？

Marcia: 因为 **SET COMPATIBLE** 会影响很多东西，而不仅仅是影响 **SELECT()**函数的行为（这将浪费你许多时间去查找原本并不存在的 **BUG**）。还有好多类似的命令会带来我说的这个问题，比如，你在比较字符串时使用精确匹配，你可能会在比较前使用 **EXACT = ON**。这是一个全局解决办法因为它影响当前工作期所有的字符串比较，直到你恢复它的设置，它的设置将影响你后面程序的运行。

Andy: 我懂你的意思了。这样一来，调试程序将是一个噩梦，因为直到你在程序中设 **EXACT** 为 **ON** 前，你的应用程序都运行得好端端的没有任何问题。

Marcia: 是的。更好的办法是用 **==** 操作符强制使用精确匹配，这才是正确的局部问题用局部办法解决。

Andy: 哦是这样。不过如果你想用 **SEEK()**函数，你就不能使用 **==** 了。实际上，这正是我创建自己的 **ExactSeek()**函数的原因——我在 **SEEK()**前设置 **EXACT** 为 **ON** 然后用完了就马上恢复它的原始值。

Marcia: 是的，可你这段代码真正的目的是什么呢？你是在通过保证特定的设置仅被用于单个自包含代码块的执行，来将解决方案局部化。也许另一个例子会有所帮助。**SET UDFPARMS** 怎么样？

Andy: 给我个提示。我甚至不清楚这个设置是干嘛用的。

Marcia: 参数可以通过传值或者传址的方式被传递。当一个参数被通过传址的方式传递给一个函数或者

过程的时候，任何在被调用代码中对参数值的改动都会影响到调用代码中的初值。相反，当参数被通过传值的方式来传递的时候，被调用的代码可以随便改动参数的值而不会对调用程序中的初值产生任何影响。

Andy: 是的，而且我还知道 **Visual FoxPro** 根据参数被传递的机制来解释被调用的代码。所以，当调用语法象这样的時候：

```
luRetVal = CallMyFunction( param1, param2 )
```

Visual FoxPro 把这当作是一个函数调用，而且是在用传值的方式传递参数。然而，如果同样的代码象下面这样被调用的话：

```
DO CallMyFunction WITH param1, param2
```

那么 **Visual FoxPro** 就会把这当成是一个过程调用，并且是在用传址的方式传递参数。

Marcia: 没错。不过，当你把一块代码当成是一个函数来调用的时候，其实是 **UDFPARMS** 的设置。正是这个设置的默认值被设置成了按传值的方式传递。如果你改动了 **UDFPARMS** 的设置，那么，即使你象调用函数那样调用代码，参数还是会被用传址的方式来传递的。

```
UDFPARMS = REFERENCE
luChangeVar Initialized to [Start]
luChangeVar after Function Call = Changed Once
luChangeVar after Procedure Call = Changed Twice
luChangeVar after Function Call by Reference = Changed Thrice

UDFPARMS = VALUE
luChangeVar Initialized to [Start]
luChangeVar after Function Call = [Start]
luChangeVar after Procedure Call = Changed Once
luChangeVar after Function Call by Reference = Changed Twice
```

```
shoudfparms.prg
*****
CLEAR
LOCAL luChangeVar, lcSet

FOR lnCnt = 1 TO 2
  lcSet = IIF( lnCnt = 1, "REFERENCE", "VALUE" )
  SET UDFPARMS TO &lcSet
  ?
  ? '', "UDFPARMS = " + lcSet
  STORE "[Start]" TO luChangeVar
  ? '', "luChangeVar Initialized to " + TRANSFORM( luChangeVar )
  ShoChange( luChangeVar )
  ? '', "luChangeVar after Function Call = " + TRANSFORM( luChangeVar )
  DO ShoChange WITH luChangeVar
  ? '', "luChangeVar after Procedure Call = " + TRANSFORM( luChangeVar )
  ShoChange( @luChangeVar )
  ? '', "luChangeVar after Function Call by Reference = " + TRANSFORM( luChangeVar )
NEXT
SET UDFPARMS TO VALUE

PROCEDURE ShoChange( tuInVal )
DO CASE
CASE LOWER( tuInVal ) == "[start]"
  tuInVal = "Changed Once"
CASE LOWER( tuInVal ) == "changed once"
  tuInVal = "Changed Twice"
OTHERWISE
  tuInVal = "Changed Thrice"
ENDCASE
RETURN
```

图 3: SET UDFPARMS 的效果

Andy: 哎呀！这可能会给大量的代码都造成麻烦，而且还很难调试出来。有谁会傻的竟然这么干呢？如果你需要用传址的方式传递参数的话（例如要传递一个数组作为参数），只要在参数名称前面加上一个@

前缀就可以了。事实上，我以前甚至不知道除了这样做以外的第二种方法。

Marcia: 现在你理解我说的“用全局的解决方案来处理局部问题”的意思了吧？

Andy: 够清楚了！不过我们有点跑题了。还是回头说 **SET COMPATIBLE** 吧！除了 **SELECT()** 函数以外还有什么受它影响的？

Marcia: 帮助文件中列出的有 32 项（其中包括一项只说了句“菜单命令”却又不告诉我们具体是哪个菜单命令的）。不过，其中只有大概一打的问题看起来是 **Visual FoxPro** 适用的——其它的各项关心的都是 **FoxPro 2.x** 问题。可那些适用的部分中包含着一些真正麻烦的特性改动。

Andy: 奇怪，这里提到的很多命令在它们的帮助文件中根本就没有与 **SET COMPATIBLE** 有关的解释。

Marcia: 比如说？

Andy: **APPEND MEMO**、**BROWSE**、**INKEY()**、**LASTKEY()**、**SET MESSAGE**、以及 **SET PRINTER**，大概有半打。从我的测试（当然是比较简单的测试）的情况来看，对它们来说，我找不到 **SET COMPATIBLE** 与否有什么影响。这我就奇怪了，为什么要提到它们呢？

Marcia: 我们先来处理那些能够确定的吧！最令我头疼的，是在处理数组的时候的 **DIMENSION** 和 **STORE** 命令的表现。

Andy: 那我先忍一下。最大的问题是什么？

Marcia: 在 **Visual FoxPro** 里，重定义一个已存在的数组会增加新的行和列，但是不会影响已经存在的内容，对吧？

Andy: 没错。此外，我通常还会把我的数组初始化为空字符串或者零来避免默认的表现（会把所有的元素设置为.F.）可能会带来的问题。用 **STORE** 命令来这么干很容易。

Marcia: 但这些表现都依赖于 **SET COMPATIBLE** 的设置。看图 4 和图 5。当 **COMPATIBLE = ON** 的时候，不仅重定义一个数组会把数组给初始化（不管你是使用 **DIMENSION** 还是 **DECLARE** 都没有区别）了，而且用 **STORE** 甚至会删除整个数组而代之以建立一个同名的变量。

Compatible = ON
 Resize existing array from 3 to 5 rows
 latest[1] = .F.
 latest[2] = .F.
 latest[3] = .F.
 latest[4] = .F.
 latest[5] = .F.

Compatible = OFF
 Resize existing array from 3 to 5 rows
 latest[1] = Apples
 latest[2] = Bananas
 latest[3] = Pears
 latest[4] = .F.
 latest[5] = .F.

```

resizearray.prg
*****
*** Name.....: RESIZEARRAY.PRG
*** Date.....: 02/01/2005
*** Compiler.: Visual FoxPro 09.00.0000.2412 for Windows
*** Function.: Impact of re-sizing arrays when COMPATIBLE = ON
*****

ACTIVATE SCREEN
CLEAR
CLEAR ALL
SET COMPATIBLE ON
LOCAL ARRAY laTest[3]
laTest[1] = "Apples"
laTest[2] = "Bananas"
laTest[3] = "Pears"
? "", "Compatible = ON"
? "", "Resize existing array from 3 to 5 rows"
DIMENSION laTest[5]
FOR lnCnt = 1 TO ALEN( laTest, 1 )
  ? "", "latest[" + TRANSFORM( lnCnt ) + "] = " + TRANSFORM(latest[lnCnt])
NEXT
?
SET COMPATIBLE OFF
LOCAL laTest[3]
laTest[1] = "Apples"
laTest[2] = "Bananas"
laTest[3] = "Pears"
? "", "Compatible = OFF"
? "", "Resize existing array from 3 to 5 rows"
DIMENSION laTest[5]
FOR lnCnt = 1 TO ALEN( laTest, 1 )
  ? "", "latest[" + TRANSFORM( lnCnt ) + "] = " + TRANSFORM(latest[lnCnt])
NEXT
  
```

图 4: SET COMPATIBLE 影响数组的缩放

Compatible = ON
 Instead of being initialized, array is replaced by a variable
 laTest = ON
 latest is no longer an array!

Compatible = OFF
 Array is initialized as expected
 latest[1] = OFF
 latest[2] = OFF
 latest[3] = OFF

```

storearray.prg *
*****
*** Name.....: STOREARRAY.PRG
*** Date.....: 02/01/2005
*** Compiler.: Visual FoxPro 09.00.0000.2412 for Windows
*** Function.: Impact of STORE on arrays when COMPATIBLE = ON
*****

ACTIVATE SCREEN
CLEAR
CLEAR ALL
SET COMPATIBLE ON
LOCAL ARRAY laTest[3]
STORE "ON" TO laTest
? "", "Compatible = ON"
? "", "Instead of being initialized, array is replaced by a variable"
IF TYPE( 'latest', 1 ) = "A"
  FOR lnCnt = 1 TO ALEN( laTest, 1 )
    ? "", "latest[" + TRANSFORM( lnCnt ) + "] = " + latest[lnCnt]
  NEXT
ELSE
  ? "", " laTest = " + laTest
  ? "", "latest is no longer an array!"
ENDIF
?
SET COMPATIBLE OFF
LOCAL ARRAY laTest[3]
STORE "OFF" TO laTest
? "", "Compatible = OFF"
? "", "Array is initialized as expected"
FOR lnCnt = 1 TO ALEN( laTest, 1 )
  ? "", "latest[" + TRANSFORM( lnCnt ) + "] = " + latest[lnCnt]
NEXT
  
```

图 5: SET COMPATIBLE 还会改变 STORE 怎么影响一个数组

Andy: 哦，那太糟糕了！重写这个程序的时候我广泛的使用了数组，而且坦白的说我还根本没有注意到任何旧的表现。可现在我不得不去好好检查一下这些东西了——这种表现简直太荒谬了！

Marcia: 可推测起来可能 dBase 就是这么设计的。毕竟，兼容模式就是干这个的——让 VFP 表现得象 dBase 一样。

Andy: 我当时正在看 FSIZE()。这又是一个恐怖事件。它不是去返回指定字段的长度，反而试图去返回一个指定名称的文件的大小。一般的结果是，你会碰到一个“找不到文件”的错误。

Marcia: 就是这样，而这里还有一个不小心的话会让你栽一道东西。我知道那些你正在处理的数据中使用了字符型的主键，在记录中主键的值是用空格来填充（pad）的。

Andy: 是的，不过不是我选的。我得声明。

Marcia: Fox 2.x 的代码是不是到处都会有 LIKE() 函数，或者偶然会有？

Andy: 是的，事实上的确是这样。我第一次看到这个函数的时候还不得不去查了下资料，因为我不知道它是干嘛用的。你问这干嘛？

Marcia: 好吧，现在我们知道为什么了、或者至少能给为什么猜出一个好点的答案了——是它（指 Like() 函数）在要用 COMPATIBLE = ON。在这个设置下，字符串是先被去掉空格然后再被 LIKE() 用来进行比较的，而不是象正常的 Visual FoxPro 使用的那种“把空格看作是被比较字符串本身的一部分”的方式。

Andy: 啊哈！这个解释还回答了别的一些当时我已经注意到了、但却没有真正理解的事情。原来的代码在一个 Scan 循环中使用了 LIKE() 来选择匹配的记录，当我重写这段代码的时候，我使用了一个 SQL 查询可却得不到应有的结果，直到我用代码明确的去掉我用来连接表的主键中的空格才搞定。我没有意识到正是 COMPATIBLE 的设置使得原来的代码能够有效工作。

Marcia: 阴险，是吧？这些正是在使用这些全局设置的时候你可能会碰到的问题。尽管解决了一个问题，可它们经常会导致在别的完全无关的地方产生意外的问题。

Andy: 让我惊讶的是这些表现上的区别的广泛。我过去总以为 FoxPro 和 dBase 是相当一致的，可显然它们有一些相当原则性的不同。

Marcia: 它还再次证明了我们的假设经常是有缺陷的。所以，当面对程序意外的表现的时候，应该冷静而理智的检查所有的事情而不是立即去哭喊 “**Bug!**”。当然，当所有办法都没用的时候，你还可以去看帮助。

Andy: 不能告诉别人！所有我们用来演示在表现中的区分的代码都包含在本月专栏的下载文件中。

下载文件：504KITBOX.ZIP