

# FoxTalk 2.0

Solutions for Microsoft® Visual FoxPro® Developers



2004 年 7 月刊

## VFP9.0: 怀柔还是扩军? — David Stevenson 著 LQL.NET 译

Page.3

六月的第一个礼拜, DevEssentials 会议在堪萨斯城举行, 我刚从那儿回来。组织者安排了主会场和分会场所 VFP9 测试版进行了全面阐述、讨论、研究。我预言 VFP 的这个版本将会非常流行, FOXER 们肯定都会忍不住升级到这个版本。

## Treeview 之母, 第一部分 — Doug Hennig 著 Fbilo 译

Page.5

如果你曾经使用过 TreeView 控件, 你肯定会重视 Doug Hennig 这个月的文章: 它提供了一个封装了 TreeView 所有古怪特性的类, 你只要自己做几个方法就可以对这个控件实现完美的控制, 并向你的用户提供所需的功能。

## 拆分工具 (第一部分) — Lauren Clarke & Randy Pearson 著 CY 译

Page.15

这是 VFP 高级文本处理的系列文章中的第一篇。Lauren Clarke 和 Randy Pearson 将带着你一步一步地开发一套可重用的 VFP 高级文本处理工具。在这四个部分系列的结尾, 你将会有一个文本拆分框架, 可重用于任何类型的应用程序。

## 来自 VFP 开发组的 Tips — 微软 VFP 开发团队 著 LQL.NET 译

Page.27

现在 VFP9 公开测试版已经提供下载了 (<http://msdn.com/vfoxpro>)，这里是本月 VFP 开发团队成员直接向你提供的一些 TIPS。

## 使用 Inno Setup 安装程序 — Rick Borup 著 YASUR 译

Page.29

Inno Setup 是一个免费的但是功能强大而且非常适用于 vfp 程序的安装工具，它不仅为开发者提供了强劲的功能，而且为最终用户呈现出了非常专业的界面。Rick Borup 为你介绍了 Inno Setup 而且会告诉你在 vfp 中如何来使用它。这篇文章包含了一些基本概念并且带你建立一个简单的安装程序。

## 在 EditBox 中实现自动完成 — Andy Kramek & Macia Akins 著 Fbilo 译

Page.33

在 VFP 9.0 中引入的文本框自动完成非常的酷，可惜只适用于文本框控件。但是，还有很多情况下自动完成功能能够为一个应用程序增加极大的价值。在这个月的专栏中，Andy Kramek 和 Marcia Akins 向你展示了怎样为编辑框实现类似的功能，它适用于 VFP 9.0 以及之前的版本。

# VFP9.0：怀柔还是扩军？

原著：David Stevenson

翻译：LQL.NET

---

六月的第一个礼拜，DevEssentials 会议在堪萨斯城举行，我刚从那儿回来。组织者安排了主会场和分会场所对 VFP9 测试版进行了全面阐述、讨论、研究。一些与会者当即下载了 VFP9 测试版在那里埋头试用，另一些则跟着功能议题挨个尝试，试图快速掌握这些 VFP 开发团队带来的强大功能。当问到他们“VFP9 怎么样？”时，许多 FOXER 回答“了不起”、“哇塞”或其他夸张的词汇，但惊呼的同时带着一丝紧张的笑容，紧张，是因为有这么多的新东东要学；笑，是因为 VFP9 是如此的有趣和激动人心。我预言 VFP 的这个版本将会非常流行，FOXER 们肯定都会忍不住升级到这个版本。

## DevEssentials 会议的一些亮点

我参与的几个议题都分享了“可扩展”这一 VFP 主旋律。Ken Levy 的议题“VFP9 的可扩展性”指出了 VFP 语言和开发环境中新增的一些有趣的东西。他指出在 VFP9，我们可以改变属性页(Property Sheet)中的默认行为，甚至可以拦截系统菜单和快捷菜单的选择，用自己设计的对话框来代替她，以此来扩展 VFP 的内置行为（Doug Hennig 在 FOXTALK 六月刊已经写了关于属性页的文章，过些时候他还将会发布关于如何拦截菜单的文章）。

Doug Hennig 的两个议题集中在如何在设计时和运行时对报表系统进行扩展。我个人认为报表系统是 VFP9 中最激动人心的一部分。这些年 VFP 开发团队对报表设计器的改动实在太少了，而这一次 VFP 开发团队不仅增加了几个重要的特性（比如多细节带区），更为重要的是通过“HOOK”和“Subclassing”尽可能地开放了整个报表核心引擎。

Toni Feltman 的议题是“为应用程序加上智能感知”，他不仅展示了文本框的 Auto-Complete 功能，还演示了如何定制智能感知在编辑 MEMO 窗口时用——你可以让你的应用程序在用户面前牛逼一下。实际上整个会议最滑稽的那一刻就是在谈及智能感知兼容性的时候，请往下看。

晚上的小组讨论很有意思，参与的有 FOXer 也有.NETer，大家都有自己所钟爱的开发工具，于是一场善意的明争暗斗便开始了。有个.NET 家伙吹嘘说他有个第三方工具，可以让他只要输入“fi”就可以在编辑窗口插入带整形参数的函数云云，说一通，这时一个 FOXER 笑着说了一句话：“这就是我们的 IntelliSense 呵”，对方当即吐血数升……

各位，我们拥有着一个令人难以置信的可扩展的开发工具，我们要如何挖掘她的潜能呢？

## VFP9.0：我们要怀柔还是扩军？

VFP9 的投放在各个在线论坛一石激起千层浪，好评如潮，世界各地的 FOXER 们充分肯定了 VFP 开发团队的努力成果。不久后测试版提供下载，扩展 VFP 的代码例程也开始涌现，FOX 社区一贯的“互助”精神再度掀起一个高潮。

举个例子，David Frankenbach 在周末花了点时间为 VFP9 新的锚（Anchor）开发了一个“生成器”，用来正确快捷地设置 Anchor 属性值。然后放到网上供大家下载使用（[www.geocities.com/df\\_foxpro/buildanchor.htm](http://www.geocities.com/df_foxpro/buildanchor.htm)）。很多人都这样做了，于是，VFP9 就这样被自由地快速地扩展了。DevEssentials 与会人员到会中就收到了大量象这样的“外挂”（噢，这是今年参加这个会议的巨好的理由）。

“怀柔然后扩军”这个短语通常是带有贬义的，但我这里的意思是世界各地的 FOX 社区将通过 VFP 开发团队提供的这些“HOOK”，来尽可能地、无止境地扩展 VFP 的新功能。让我们赶赶时髦吧，弟兄们，记得把你的成功分享给其他人！

在 VFP9 的杠杠作用下，请准备好“拿灭火水龙带喝水吧”，你定会为 VFP 的扩展程度瞠目结舌的！如果你还没预定所有的 VFP 相关的出版物和在线杂志，那么现在该订了；如果你还没登录到 VFP 相关的论坛、邮件列表和新闻组，那么现在该上了；如果你还没加入到 VFP 开发行列，那么你现在该来了；当然，如果你现在还没从 <http://msdn.com/vfoxpro> 下载 VFP9 BETA，那么现在赶快下过来爽一下吧！

**注：**明年，VFP9 在报表系统及其他方面的扩展将更加深入、具体。VFP9.0 就要在 2004 年第 4 季度如期发布了，不过我们仍然不会放弃 VFP 以前的版本，只要是有趣的文章，我们都将奉献给我们可爱的 VFP 开发者！

# Treeview 之母，第一部分

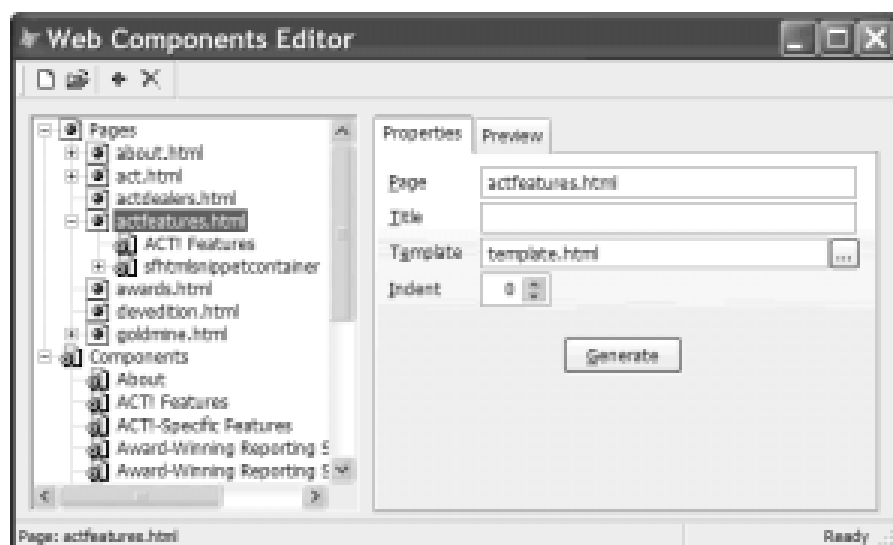
原著: Doug Hennig

翻译: Fbilo

如果你曾经使用过 TreeView 控件，你肯定会重视 Doug Hennig 这个月的文章：它提供了一个封装了 TreeView 所有古怪特性的类，你只要自己做几个方法就可以对这个控件实现完美的控制，并向你的用户提供所需的功能。

**在** Foxtalk 2004 年 5 月刊中，我们看了一套“让你可以生成一个由一些可重用的 HTML 部件的集合组成的 Web 页”的类。不过，我很快发现，在 Browse 窗口中生成驱动这些类的表的活可没那么有趣。本月和下个月，我将为这些表建立一个前台的界面以简化这件工作，在这个过程中，最后你会获得一个非常有用的 TreeView 封装类以用于你自己的项目。

当我开始想象这个编辑器工具应该是什么样的时候，我想到的是一个这样的表单：左边是一个 Treeview 控件（因为 HTML 的内容是可以分层的），右边是一个 PageFrame，其中的一页用于显示选中节点的属性、另一页则用嵌入的 Web 浏览器来显示 HTML 的预览。图 1 展示的就是完工的产品，WebEditor.SCX。



（图一）

想象完毕，我就开始呻吟起来——我曾经做过类似的表单，它需要成吨的工作。TreeView 既是一个

非常丰富、又是一个非常“恶劣”的控件，包含着大量需要操心的属性、事件和方法（PEMs）。下面只是你在使用 Treeview 时会碰到的麻烦中的一部分：

- 加载节点非常的慢，所以如果有大量的节点的时候，最好一开始只加载顶层的节点，只有当一个顶层节点第一次被展开的时候才去加载该节点下的子节点。实现这个功能需要大量的工作：你需要在顶层节点下面加上一个“假的”子节点，否则这个顶层节点前面就不会显示出 + 号。然后，当这个顶层节点被展开的时候，你需要去检查它的第一个子节点是否是一个假节点，如果是，则需要删除它并加载全部子节点。
- 删除“假”节点并加载子节点会导致大量的屏幕刷新活动。对于 VFP 表单来说，我们可以用 `LockScreen = .T.` 来阻止屏幕的更新，但是 Treeview 可不受 LockScreen 的限制，并且自己还没有相应的属性。所以，我们只好自己调用一个 Windows API 函数来锁定和解锁 TreeView。
- TreeView 采用的坐标系统是 Twips (1/1440 英寸) 而不是 Pixels，所以在调用需要坐标参数的 TreeView 方法前你必须自己将 Pixels 转换成 Twips。这又涉及到 Windows API 函数。
- 用右键单击、或者拖放一个节点并不会使该节点变成当前选中的节点，所以为当前节点显示快捷菜单、或者支持拖放的代码必须弄清楚哪个节点是用户单击的那个。
- 你也许想让用户能够双击某个节点来执行某些操作，比如打开一个对话框以编辑该节点的隐藏数据。不幸的是，如果该节点有子节点，那么双击该节点会它被展开或缩起。
- 如果你支持 OLE 拖放，当你拖放到 TreeView 的顶端或者底部的时候，TreeView 不会自动的向上或者向下滚动，此外，在拖放到某个节点上的时候，该节点也不会被高亮显示。你必须自己实现这些功能。

基于以往“处理所有这些麻烦、以及协调 TreeView 节点的单击和表单的其它部分行为”的经验，我决定是时候一劳永逸的建立一个封装有所有我需要特性的部件了。所以，建立 Web 部件编辑器的第一步，就是建立这个可重用的 TreeView 类。

定义在 SFTreeView.VCX 中的 SFTreeViewContainer 类是 SFContainer（位于 SFCtrls.VCX 中的我的容器基类）的一个子类。它包含一个 TreeView 和一个 ImageList 控件，后者将是 TreeView 的图片来源。

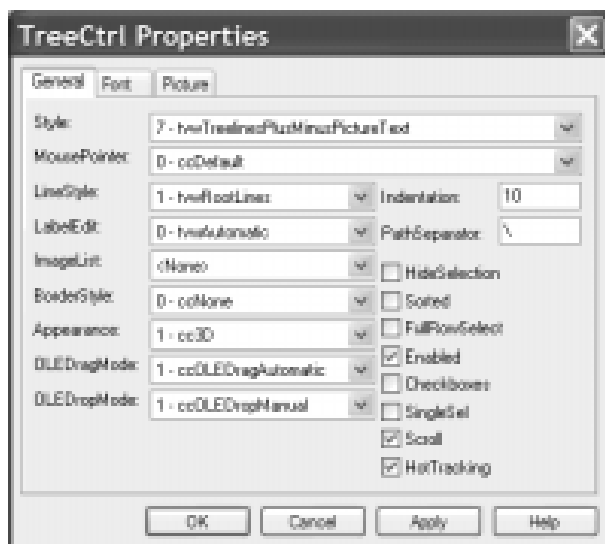
大多数 TreeView 的事件都是调用这个容器的方法，这样做有几个理由。由事件调用方法，是好的

设计。此外，在类设计器里面，获得容器的方法要比 TreeView 容易些。

我考虑使用 VFP 8 新的 BindEvents() 函数来避免把代码放到 TreeView 的事件里面去。但是某些 TreeView 的事件，比如 BeforeLabelEdit，是通过引用接收它们的参数的，这种功能是 BindEvents() 不支持的。

## TreeView 的表现

我用如图 2 所示的 TreeView 窗口属性来可视化的设置 TreeView 控件的属性。如果你想要在一个子类中使用不同的设置，例如将 Appearance 属性设置为 0 以让 TreeView 显示为平面而不是 3D 样式，你需要在代码中改变它。由于某些不明的原因，在一个子类中改动 TreeView 属性是无效的。所以，请在一个 SFTreeViewContainer 的子类的 INIT 方法中，使用 DODEFAULT() 再加上改变想要改动的 TreeView 属性的代码。



(图二)

用于 TreeView 的图片来自于容器中的 ImageList 控件。象 TreeView 一样，再一个子类中可视化的向 ImageList 中添加图片的办法无效，所以我将 SFTreeViewContainer 的 Init 方法设计为调用封装了的 LoadImages 方法。在你的子类的这个方法中，你可以调用 ImageList 对象的 ListImages 集合的 Add 方法来添加图片。注意，你必须使用 VFP 的 LoadPicture() 函数以向 Add 方法传递一个要被加载的图片对象的引用。

这里是 WebEditor.SCX 表单上 SFTreeViewContainer 对象的 LoadImages 方法里面的代码。它将 ImageList 的 ImageHeight 和 ImageWidth 属性设置为将要被加载的图片的值，然后，加载 Page.Gif 作为图片 1，key 值为“Page”；加载 Component.Gif 为图片 2，key 值为“Component”。

```

With This.oImageList
    .ImageHeight = 16
    .ImageWidth = 16
    .ListImages.Add(1, 'Page',;
        LoadPicture('Page.Gif')
    .ListImages.Add(2, 'Component',;
        LoadPicture('Component.Gif'))
endwith

```

## 加载 TreeView

这里是 SFTreeViewContainer 的四个与将正确的节点加载入 TreeView 相关的自定义属性：

- **lLoadTreeViewAtStartup**——如果这个属性为 .T. (默认值)，该容器的 Init 方法将调用 LoadTree 方法来加载 TreeView。如果它是 .F.，你就要在需要加载 TreeView 的时候手动调用 LoadTree 方法。如果需要在加载 TreeView 之前做某些工作的话，这个属性就重要了。
- **lAutoLoadChildren**——如果这个属性为 .T.，那么在整个 TreeView 中所有的节点都将被加载。这个值适用于节点不多的情况，否则的话将导致严重的性能问题，加载 TreeView 可能会花费太长的时间以至于让用户以为程序崩溃了。这种情况下就需要把 lAutoLoadChildren 属性设置为 .F. (默认值)，这样，容器会只加载顶层节点（或者再多加载一些，根据后面会讲到的属性而定）。任何加载了的顶层节点都会有一个标题为“Loading...”的假节点，这样一个 + 号就会出现在该顶层节点前面，表示该节点可以被展开。当一个节点第一次被展开的时候，容器的 TreeExpand 方法（由 TreeView 的 Expand 方法所调用）将删除这个假节点，并为它添加真正的子节点。
- **nAutoExpand**——这个属性指定当 TreeView 被加载的时候，哪一层的节点应该被自动展开。默认值是 0，意思是没有一个节点应该被展开（虽然，你后面会看到，将 TreeView 恢复到它最近的查看状态会因为某些节点而覆盖这个值）；将 nAutoExpand 设置为 1 表示顶层节点应该被展开，这意味着即使 lAutoLoadChildren 属性为 .F.，容器还是将加载顶层节点的直接子节点（但不包括这些子节点的子节点）；设置为 2 表示顶层节点的直接子节点应该被展开（由此导致它们的直接子节点将被加载），等等。
- **lUsePathAsKey**——如果这个属性为 .F.，你将需要给 TreeView 中的每个节点分配唯一的 key



值。如果它为 `.T.` (默认), 你就不需要关心 `key` 值的问题了; 容器将使用节点的 `path` (类似于一个文件的路径) 作为它的 `key`。

`LoadTree` 方法执行主要的加载 `TreeView` 的工作。基于专栏文章空间限制的原因, 我就不在这里展示它的代码了, 简单的说一下它的工作原理。它首先通过调用 `LockTreeView` 方法来锁定 `TreeView` 控件 (使用一个 Windows API 函数), 这样就不会频繁的刷新。

下一步, 任何当前已经存在、并且已经被展开的节点都被保存到 `aExpandedNodes` 数组属性中, 而当前选中节点的 `key` 值则被保存在 `cLastNode` 属性里。这让你可以再次调用 `LoadTree` (例如, 如果 `TreeView` 显示的记录是来自于一个网络上的其它用户也正在编辑中的表的情况)、并恢复每个节点的扩展状态和当前选中节点。

然后, `LoadTree` 通过调用 `GetRootNodes` 方法加载顶层节点, 将这些顶层节点作为一个集合传递。`GetRootNodes` 是一个封装的方法, 你必须在一个子类中自己实现它, 以用一系列包含那些顶层节点的信息的对象填充这个集合。稍后我将讨论关于这个问题更多的内容。

接着, `LoadTree` 为集合中的每个对象调用 `LoadNode` 方法以将该对象加载到 `TreeView` 中去。任何原来已展开的节点都将被重新展开、原来已经被选中节点也将被重新选中 (如果原来存在这样的节点的话)。最后, `LoadTree` 再次调用 `LockTreeView` 来解除对 `TreeView` 的锁定。

你必须在一个子类、或者 `SFTreeViewContainer` 的实例的 `GetRootNodes` 方法中写入代码, 以用包含着所有顶层节点信息的对象来填充那个集合。为了建立这么一个对象, 可以调用 `CreateNodeObject` 方法, 它简单的建立一个 `Empty` 对象, 并添加如下属性:

- `key`——节点在 `TreeView` 中的 `key` 值。这个属性自动被设置为 `SYS(2015)`。但是如果 `lUsePathAsKey` 属性被设置为 `.F.` 的话, 则你需要将它改动为一个唯一值。
- `Text`——节点显示的文本。
- `Image`——该节点的图片在 `ImageList` 中的名称或者索引号。
- `SelectedImage`——该节点被选中时显示的图片在 `ImageList` 中的名称或者索引号。你可以让这个属性保留为空以使用与 `Image` 属性设置的同一个图片。
- `Sorted`——如果节点应该排序则为 `.T.`。
- `HasChildren`——如果节点有任何子节点, 则为 `.T.`。

如果你还需要增加一些其它的属性, 你可以覆盖 `CreateNodeObject` 方法。在这样的情况下, 你还

需要覆盖 `GetNodeItemFromNode` 方法,该方法将建立一个来自于当前 Treeview 节点的节点 Item 对象,并填充该对象的属性。

`GetRootNodes` 应该为每个顶层节点建立一个对象,为对象填充好属性,然后把它添加到被传入的集合中去。这里是在 `WebEditor.SCX` 表单上的 `SFTreeViewContainer` 对象中的代码,它会建立两个顶层节点:Pages 节点,用于那些定义在 PAGES 表中的页(我在 2004 年 5 月刊中讨论过这个表);Components 节点,用于那些定义在 CONTENT 表中的 Web 部件。`ccPAGE_HEADER_KEY` 和 `ccCOMPONENT_HEADER_KEY` 是定义在这个表单的包含文件 `WebEditor.H` 中的常量。

```
lparameters toCollection
local loNodeItem
loNodeItem = This.CreateNodeObject()
go top in PAGES
with loNodeItem
    .Key = ccPAGE_HEADER_KEY
    .Text = 'Pages'
    .Image = 'Page'
    .Sorted = .T.
    .HasChildren = not eof('PAGES')
endwith
toCollection.Add(loNodeItem)

loNodeItem = This.CreateNodeObject()
go top in CONTENT
with loNodeItem
    .Key = ccCOMPONENT_HEADER_KEY
    .Text = 'Components'
    .Image = 'Component'
    .Sorted = .T.
    .HasChildren = not eof('CONTENT')
endwith
toCollection.Add(loNodeItem)
```

你还必须实现 `GetChildNodes` 方法。象 `GetRootNodes` 一样,这个方法用节点 Item 对象填充一个集合。在这种情况下,这些对象表示一个节点的子节点。你应该给 `GetChildNodes` 传递三个参数:需要

取得子节点的节点的 Type 和 ID、以及要被填充的集合。

这里是在 WebEditor.SCX 表单上的 SFTreeViewContainer 对象的 GetChildNodes 方法中的代码。在“Pages”根节点的情况下，在 PAGES 表中的记录被加载到集合里，ccPAGE\_KEY (另一个常量)和 ID 作为 key 值。在“Component”根节点的情况下，CONTENT 表中的记录被加载，使用 ccCOMPONENT\_KEY 和 ID 作为 key 值。

对于某个特定的 page 节点，在 PAGECONTENT 表中与其相对应的、parent 字段中又没有内容的记录被加载（译者注：在 Foxtalk 2004 年 5 月刊中提到，PageContent 表是 Page 表与 Content 表之间多对多关系中的那个中间的关联表，表示在一个页上使用了哪些 Web 页部件。在 PageContent 表中的 PageId 字段与 Page 表相关联、ContentID 字段与 Content 表相关联，而如果当前部件在本页中是另一个部件的子部件，则 Parent 字段记录其父部件在本表中的 ID）；Parent 字段中有内容的记录被忽略，因为它们将作为它们父节点的子节点被加载。

```
lparameters tcType, ;
tcKey, ;
toCollection
local loNodeItem, ;
lcWhere

do case
    * 如果这是 “Page” 节点，则用所有定义了的页填充集合
    case tcType = ccPAGE_HEADER_KEY
        select PAGES
        scan
            loNodeItem = This.CreateNodeObject()
            with loNodeItem
                .Key = ccPAGE_KEY + transform(ID)
                .Text = trim(PAGE)
                .Image = 'Page'
                .HasChildren = seek(PAGES.ID, 'PAGECONTENT',
                    'PAGEID')
            endwith
            toCollection.Add(loNodeItem)
        endscan
```

\* 如果这是“Components”节点，则用所有定义了的部件填充集合

```
case tcType = ccCOMPONENT_HEADER_KEY
  select CONTENT
  scan
    loNodeItem = This.CreateNodeObject()
    with loNodeItem
      .Key = ccCOMPONENT_KEY + transform(ID)
      .Text = trim(NAME)
      .Image = 'Component'
    endwith
    toCollection.Add(loNodeItem)
  endscan
```

\* 如果这是一个带有子节点的 Page 节点或者 content 节点，则加载其内容子节点

```
case tcType = ccPAGE_KEY or ;
  seek(tcKey, 'PAGECONTENT', 'PARENT')
  if tcType = ccPAGE_KEY
    lcWhere = 'PAGEID = tcKey and empty(PARENT)'
  else
    lcWhere = 'PARENT = tcKey'
  endif tcType = ccPAGE_KEY

  select PAGECONTENT.ID, ;
    PAGECONTENT.CLASS, ;
    CONTENT.NAME ;
  from PAGECONTENT ;
  left outer join CONTENT ;
  on PAGECONTENT.CONTENTID = CONTENT.ID ;
  into cursor _TEMP ;
  where &lcWhere ;
  order by PAGECONTENT.ORDER
  scan
    loNodeItem = This.CreateNodeObject()
    with loNodeItem
```

```

        .Key = ccPAGE_COMPONENT_KEY + ;
            transform(ID)
        .Text = trim(iif(empty(nvl(NAME, '')), ;
            CLASS, NAME))
        .Image = 'Component'
        .HasChildren = indexseek(ID, .F., ;
            'PAGECONTENT', 'PARENT')
    endwhile
    toCollection.Add(loNodeItem)
endscan
use

```

- \* 如果节点的数据错误，那么就什么也不错
- \* （其实 OTHERWISE 语句并非是必须的，
- \* 不过这里还是加上一句，以免 Andy Kramek 来吹毛求疵）

```
otherwise
```

```
endcase
```

## 恢复 TreeView 的状态

当你再次运行使用了 SFTreeViewContainer 的一个表单时，你也许会希望它显示得跟它上一次运行时一摸一样，原来选中的节点现在也选中，原来展开的节点现在同样展开。SaveSelectedNode 和 RestoreSelectedNode 这两个方法就可以那两个问题，它们分别由 Destroy 和 Init 来调用。这两个方法封装在 SFTreeViewContainer 里，所以要由你自己来决定是在一个子类还是在一个实例中去实现它们。

SaveSelectedNode 应该保存下选中节点和每个展开了的节点的 key 值（你可以把它们保存在你喜欢的任何地方：Windows 注册表、一个 INI 文件、一个表、一个 XML 文件——换句话说，可以是你喜欢的任何存储机制）。这里是在 WebEditor.SCX 表单中的 SaveSelectedNode 方法的代码，它使用 WriteINI.PRG 来将信息保存到一个 INI 文件里，这个 PRG 在附带的下载文档中。cWebSitePath 是当前打开的 Web 站点文件的位置。

```

local lcFile, ;
    lnNode, ;
    lnHandle, ;

```

```

loNode, ;
lcContent, ;
laLines[1], ;
lnLines, ;
lnI, ;
lcLine

```

```

with This.oTree

```

```

    if vartype(.SelectedItem) = '0'
        lcFile = Thisform.cWebSitePath + 'settings.ini'
        lnNode = 0
        if not file(lcFile)
            lnHandle = fcreate(lcFile)
            fclose(lnHandle)
        endif not file(lcFile)

```

```

        WriteINI(lcFile, 'Nodes', 'SelectedNode', ;
            .SelectedItem.Key)

```

```

        for each loNode in .Nodes
            if loNode.Expanded
                lnNode = lnNode + 1
                WriteINI(lcFile, 'Nodes', 'Expanded' + ;
                    transform(lnNode), loNode.Key)
            endif loNode.Expanded
        next loNode

```

```

* 如果 INI 文件中有一些以前保存下来的、但现在不再展开的节点,
* 则删除它们

```

```

lcContent = ''
lnLines = alines(laLines, filetostr(lcFile))

```

```

for lnI = 1 to lnLines
    lcLine = laLines[lnI]
    if lcLine <> 'Expanded' or ;

```

```

        val(streextract(lcLine,'Expanded','=')) <= ;
        lnNode
        lcContent = lcContent + lcLine + ccCRLF
    endif laLines[lnI] = 'Expanded' ...
next lnI
    strtofile(lcContent, lcFile)
endif vartype(.SelectedItem) = '0'
endwith

```

RestoreSelectedNode 恢复这些数据项的方法与此类似。这里我们就不展示在 WebEditor.SCX 表单中的代码了，它只是简单的使用 ReadINI.PRG 去恢复来自于 INI 文件中的 cLastNode 和 aExpandedNodes。

## 总结

这是这个月的全部内容了。下个月，我将讨论处理节点的选择、OLE 拖放、以及一个最后将 SFTreeViewContainer 和一个用于选中节点属性的 PageFrame 绑定到一起的表单类的内容。

涉及源码：407HENNIG.ZIP

# 拆分工具（第一部分）

原著: Lauren Clarke 和 Randy Pearson

翻译: CY

---

这是VFP高级文本处理的系列文章中的第一篇。Lauren Clarke 和 Randy Pearson将带着你一步一步地开发一套可重用的VFP高级文本处理工具。在这四个部分系列的结尾，你将会有一个文本拆分框架，可重用于任何类型的应用程序。

**我**们把我们的工作定位于VFP的数据驱动、动态Web应用驱动。由于我们的许多工作包括科学研究协作和文档管理，我们需要处理大量的零散表格，无分隔的文本。在这些应用里，我们对所有的各式各样的任务使用我们的拆分工具—转换原始数据为数据库格式，提供知识管理工具（比如，自动连接到大型术语和字典表），生成PDF和DOC文件，数据挖掘，和有效密码检查。因此你将会知道在这些文章的开头，有我们的拆分框架能为我们所做的一些事：

- 允许拆分作为应用范围的服务
- 为更好的性能作缓冲和重用拆分对象
- 对元数据配置更复杂的拆分任务
- 分解复杂拆分任务为可重用的、基本步骤
- 提供对原始文本基于模式的HTML连接（wiki类型拆分）
- 提供内置的动态文本替换，替换是基于完整内容匹配
- 避免连续拆分处理的冲突
- 转译纯文本为HTML标记，允许作者没有任何HTML技巧可以产生格式化文本（比如，转换“\*bold\*”为“<strong>bold</strong>”）
- 从简单的分栏原始文本创建复杂HTML表

在我们为你演示如何编写一个拆分框架前，我们来看看VFP给我们带来了什么。VFP提供了一系列强大的纯文本处理和转换的函数。多年以来，VFP开发者已经对各式各样的任务使用这些函数，包含数据提取，数据清除，文档生成，还有近来，生成标记XML，HTML，和其他SGML语言。

VFP有函数可用于多种类型变量和字符型的相互转换。我们可以在一个大的文本变量里查找和替换任意的字符串。我们可以分解字符串为更小的元素，并可以重新快速组合。所有这些都可以只用VFP函数来完成。这第一篇文章将研究公共的VFP字符处理函数，着重于那些近来新增和增强的函数。



注意：不久将要发布的VFP9.0包含有几个增强的字符处理函数，我们将来会在这个系列的适当时候说到。

### 升级VFP，然后升级你的应用程序

最新的VFP版本有多个非常有用的字符处理和生成函数，还有增强了现有的函数。让我们现在就来亲身熟悉这些工具，因此我们不会错过任何机会来编写清晰、快捷的文本处理问题的解决方案。

比如，考虑从分隔的IP地址列表重述的任务。多年前，这个分类任务将完成类似于这个的某些事情（参见在下载文件里的sample.prg的这个和其他示例）：

```
lcList = [205.180.85.40;205.140.111.203;]+;
[206.63.99.226;206.63.99.227;206.63.99.228]
lnEndPos = 0
lnStartPos = 0
FOR lnK = 1 TO OCCURS([;], lcList) + 1
  lnEndPos = AT( [;], lcList, lnK )
  lnEndPos = IIF( lnEndPos = 0, ;
LEN( lcList ) + 1, lnEndPos )
  lcIP = SUBSTR( lcList, lnStartPos+1, ;
lnEndPos-lnStartPos-1 )
  ? lcIP && <= print the IP
  lnStartPos = AT( [;], lcList, lnK )
ENDFOR
```

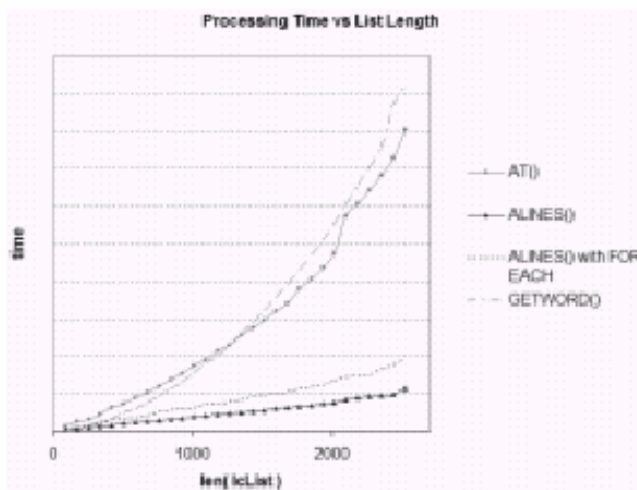
VFP很棒吧？我们可以很容易的拆分这个字符串或者任意的分隔字符串。也许并不容易，但是一旦我们找到那些在右边的+1和-1，并适当控制循环，它就可以正常运行。因为它是很快做出来的，这个代码象是无法理解的狗食早餐——在编写后五分钟就看不明白了，并且难于引导。仔细想来，我们现在做这样分类的事情可以更清楚。VFP7.0新增了ALINES()命令的完美扩展，允许我们在新的行加入数组时选择一个指定的分隔符。这可以让我们减少处理代码：

```
lcList = [205.180.85.40;205.140.111.203;]+;
[206.63.99.226;206.63.99.227;206.63.99.228]
ALINES(laIPs, lcList, .T., [;] )
FOR EACH lcIP IN laIPs
  ? m. lcIP
ENDFOR
```

是不是更好？更容易理解？更快？是的，所有的都是。这真是强大。满意吗？你可能不会满意。为什么我们要把字符串移入到数组结构，只是为了重述这些IP地址？考虑到这个替换操作可能用在VFP7里（也或者可能用于先前带foxtools.fll库）：

```
lcList = [205.180.85.40;205.140.111.203;]+;  
[206.63.99.226;206.63.99.227;206.63.99.228]  
FOR lnK = 1 TO GETWORDCOUNT( lcList, [;] )  
? GETWORDNUM( lcList, lnK, [;] )  
ENDFOR
```

这个方法是更简洁的，因为它不需要数组。并且，对于短字符串，它可以比先前的方法更快（大约三倍快于ALINES()例子）。然而，随着字符串长度的增加，GETWORDNUM()函数的性能显著的跌落（见图1）。因此，要注意只能在字符串长度比较有限的情况下使用它。这个语法也在我们可用的分隔符的数量上提供了至关重要的扩展。



**图1：**这个图演示了拆分IP地址的多种方法的相对速度。注意GETWORD方法是如何随着列表的增加而变得不理想的。然而，对于短列表，GETWORD多半能在所有方法提供最大的简洁和灵活。

假想分隔符规则改变了，并且你将不得不接受以分号或逗号作为分隔符的列表。在第一个示例里这需要分隔符规范化的步骤，将会导致变慢。在上一个情形里，我们只需要增加分隔符（实际上，更适宜的是使用一个可配置的lcDelimiters变量）。

```
FOR lnK = 1 TO GETWORDCOUNT( lcList, [;,] )  
? GETWORDNUM( lcList, lnK, [;,] )
```

ENDFOR

这个ALINES()方法也提供了多分隔符选项,但是每一个分隔符必须被指定为不同的参数(最大允许为23),它可以使得数据驱动的分隔符列表更困难。这个演示将使得你确信你要升级你的开发环境以保持VFP的最新增强。现在让我们来看看那些我们所喜欢VFP字符处理函数,着重于那些近来版本新增和增强的函数。

### \$操作符

如果一个字符表达式包含另一个字符表达式,\$将返回为真(.T.)。比如:

```
? [brush] $ [hairbrush] && prints .T.
```

这个操作符用于检测一个包含有某些分类选项的文本变量是否包含于一个选项列表里是非常便利的:

```
IF [,]+LOWER(ALLTRIM(lcColor))+[,] $ ;  
[,]+LOWER(ALLTRIM(lcColorList))+[,]  
**-- lcColor is in lcColorList  
**-- lcColor在lcColorList内  
ENDIF
```

注意我们是该如何在整理和转换为小写后通过填充[,]字符来规范比较的。为避免重复整理和填充代码,我们把\$包装在一个名为inListString()的自定义函数里。

```
FUNCTION inListString( tcItem, tcList, tcDelimiter, ;  
tltrim )
```

```
LOCAL lcTrimmedList  
tcDelimiter = IIF( VARTYPE( m.tcDelimiter ) # [C], ;  
[,], m.tcDelimiter )
```

```
IF m.tltrim  
tcItem = ALLTRIM( m.tcItem )
```

```
*-- strip all leading and trailing spaces from list  
*-- 从列表中剥去所有前导和后置空格  
lcTrimmedList = []
```

```

FOR lnK = 1 TO GETWORDCOUNT(m.tcList, m.tcDelimiter)
lcTrimmedList = m.lcTrimmedList + ;
GETWORDNUM(m.tcList, m.lnK, m.tcDelimiter + [ ])+;
m.tcDelimiter
ENDFOR
m.tcList = m.lcTrimmedList
ENDIF

tcItem = m.tcDelimiter + m.tcItem + m.tcDelimiter
tcList = m.tcDelimiter + m.tcList + m.tcDelimiter

RETURN m.tcItem $ m.tcList

```

通过利用这个函数，先前的代码变成：

```

if inListString( LOWER(lcColor), LOWER(lcColorList) )
**-- lcColor is in lcColorList
**-- lcColor在lcColorList内
endif

```

这个inListString()函数也允许我们来指定一个不是“,”的分隔符，并且我们可忽略列表项的空格，即使分隔符是空格。

### GETWORDNUM() 和 GETWORDCOUNT()

正如前面所演示，GETWORDNUM() 和 GETWORDCOUNT()对拆分分隔字符串是非常有用的。当你需要从多维数据挤出单维数据时，这类拆分是很方便的。比如，假想你运行一个带INI文件的应用时，而你想要压缩INI里的多个色彩配置属性集合为单行以减少混乱。你可以使用双分隔符的方法来实现，如下所示。

注意：下列代码只能在VFP8.0或以上工作，因为它使用了Empty类和ADDPROPERTY()函数。

```

*-- normally, we'd extract the color scheme string from
* the ini with an API call or another parsing routine
*-- 通常，我们用API调用或其他拆分过程来从INI里抽取配色字符
lcColorscheme= ;
[background, red|foreground, blue|fontcolor, green]

```

```

*-- extract the color scheme data from the .ini
* into an object
*-- 从INI里抽取配色数据放入到一个对象里
loCScheme = CREATEOBJECT("empty")
FOR lnK = 1 TO GETWORDCOUNT( lcColorscheme, [|] )
lcPropValue = GETWORDNUM( lcColorscheme, lnK , [|] )
lcProperty = GETWORDNUM( lcPropValue, 1, [, ] )
lcValue = GETWORDNUM( lcPropValue, 2, [, ] )
ADDPROPERTY( loCScheme, lcProperty, lcValue )
ENDFOR

WITH loCScheme
? .background && prints "red"
? .foreground && prints "blue"
? .fontcolor && prints "green"
ENDWITH

```

## STRTRAN()

这个是VFP的“替换”函数。它可以在另一个字符串中查找任意的子字符串，并以指定的字符串进行替换。如果你一直使用VFP，它应该是你所熟知的函数。VFP7.0扩展了这个函数的一个nFlags参数，这个参数改变着它状态，如表1所示。

**表1：**STRTRAN()函数的nFlags参数控制多种可能状态。

nFlags value	Description
0 (default)	查找大小写敏感；以完整的cReplacement文本替换。这是VFP7.0以前版本的状态。
1	查找大小写不敏感；以完整的cReplacement文本替换。
2	查找大小写敏感；以更改的cReplacement以匹配找到的字符串来替换。仅当找到的字符串都是大写或小时这个cReplacement才会被为更改。
3	查找大小写不敏感；以更改的cReplacement以匹配找到的字符串来替换。仅当找到的字符串都是大写或小时这个cReplacement才会被为更改。

STRTRAN()函数也可以用于从字符串里“剔除”指定的字符。注意cReplacement参数是可选的，所以这个代码可以从lcDirtyString变量里移去所有的%字符。

```
lcCleanString = STRTRAN( lcDirtyString, [%])
```

## CHRTRAN()

CHRTRAN()函数最初是混乱的，但是它最终是非常有用的函数。认为CHRTRAN()是字符对字符的STRTRAN()是有帮助的。比如，思考这行代码：

```
? CHRTRAN( "ABC123", "13", "*?") && prints ABC*2?
```

这个字符“1”和“3”分别被替换为“\*”和“?”。查找和替换参数中每一个字符位置决定着该如何进行替换映射。你可以让替换表达式为空，这将导致在cSearchExpression参数中的每一个字符，只要它出现就会被移出。

```
? CHRTRAN( "ABC123", "13", "") && prints ABC2
```

注意这类的操作与等效的STRTRAN()操作相比将是相当的慢，并且这个差距将会是与字符串长度很线性的。比如，看下列CHRTRAN()代码：

```
lcX = CHRTRAN( lcStr, [12], [])
```

对于25K的字符串，CHRTRAN()方法要多花费10倍于等效的STRTRAN()操作：

```
lcX = STRTRAN( lcStr, [1] )
```

```
lcX = STRTRAN( lcX, [2] )
```

这是因为CHRTRAN()是为位置而优化的，在被替换和替换字符间是一一对应的。注意不要小看CHRTRAN()作为替换的强大工具。因为CHRTRAN()是按位置优化的，当第二和第三参数是一样长时，它就会成为两步法的两个函数的重要支点：。

```
* replace 1s and 2s with *
```

```
* 替换1和2为*
```

```
lcX = CHRTRAN( m.lcStr, [12], [**])
```

```
* remove all *s
```

```
* 移去所有的*
```

```
lcX = STRTRAN( m.lcStr, [*] )
```

这两行代码比仅使用STRTRAN()方法快两倍。请记住象这样的基准测试是取决于处理的字符串的长度和字

符分布，因而它可以用于测试多种实际正文。也要记住CHRTRAN()是仅对单字符操作有用的，因此即使一个简单的任务，如替换单个空格(“ ”)为HTML的不分断空格(“nbsp;”),就需要用STRTRAN()。CHRTRAN()更灵巧的使用是两次使用这个函数，用以从一个字符串中除指定字符以外都剔除。比如，这个代码将从字符串中除数字以外都剔除。

```
lcStr=[13/3k48q5j39;3k3aa]
? CHRTRAN( m.lcStr, ;
CHRTRAN( m.lcStr, "1234567890", "" ), "" )
* prints: 1334853933
```

这里奇妙的是嵌套调用CHRTRAN()。因为里面的CHRTRAN()调用将从lcStr里清除所有的数字字符并返回所有的非数字字符，你可以这样理解：

```
? CHRTRAN( lcStr, ;
<<all the non-digit chars in lcString>> , "" )
```

这剔除lcString中所有非数字字符，仅返回数字。

### PADR()

PADR(还有相关的PADL和PADC)是以指定的字符(或空格)来扩展一个字符串以达到指定的长度。这对填充字符字段非常有用，因此可以避免在SET EXACT OFF时使用==操作。

```
lnFsize = FSIZE("cID")
SELECT * FROM SomeTable ;
WHERE cID = PADR(lcID, lnFsize)
```

也要注意第一个参数可以是除逻辑值、货币表达式、备注和图形字段外的任意类型。这意味着你可以使用PADR()来强行转换数值为字符串，而不再需要给定小数位数，如STR()所要求的那样。其实这可以用来对于用以表示数值的字符串中确定小数位数是更好用的。

这在科学应用程序中是非常重要的，提交的数值的精度(它可能来自于多种来源)必须是由输入源自行确定的，而不是那些预定或假设的精度。

```
? DecimalPlaces(1234.5678) && prints 4
? DecimalPlaces($1234.5678) && prints 4
? DecimalPlaces("1234.5678") && prints 4
```

```

FUNCTION decimalPlaces( tnValue ) AS INTEGER

* returns the number of decimal places
* present in tnValue
* 返回 tnValue中小数位数
LOCAL lnSetPoint, lnRetVal, lcStringVal

* max 18 digits either side of decimal point
* 水位位数最大为18
lcStringVal = PADR( tnValue, 40 )
lnRetVal = 0
lnSetPoint = AT( SET("point"), m.lcStringVal )
IF lnSetPoint > 0
lnRetVal = LEN( RTRIM( m.lcStringVal ) ) ;
- m.lnSetPoint
ENDIF
RETURN m.lnRetVal
ENDFUNC

```

### TEXT...ENDTEXT and TEXTMERGE()

VFP已经增加创建文本文件的TEXT...ENDTEXT命令很久了。VFP7.0新增了TO选项，它允许我们来直接输出文本到内存变量而不再是磁盘上的某个文件。如果你创建Web应用，这个是值得从VFP6.0升级的特性。并且，VFP7.0也新增了TEXTMERGE函数，在文本中的VFP表达式可以快速扩充，而无视当前SET TEXTMERGE状态。比如：

```

TEXT to lcText textmerge noshow pretext 1
    This is a test.
    Note how the initial spaces are trimmed.
    This is due to the pretext 1 directive above.
    Today is: <<DATE()>>
ENDTEXT

```

这将显示如下：



This is a test.

Note how the initial spaces are trimmed.

This is due to the pretext 1 directive above.

Today is: 2003-05-12

TEXTMERGE() 是等效于带TEXTMERGE关键字的TEXT...ENDTEXT函数。它可以接受一个字符串（即使是嵌入的文本）并进行合并，扩充以<<、>>分隔（或者你所选择的分隔符）的VFP表达式。

```
lcText = [The date is <<DATE()>>]
```

```
? TEXTMERGE( lcText )
```

这将输出：

The date is 2004-04-05.

可以嵌套调用TEXTMERGE()（不象TEXT...ENDTEXT结构—对于框架创建者来说是非常重要的理由）。

```
lcText2 = ;
```

```
[The time is <<TIME()>>, and <<TEXTMERGE(lcText)>>]
```

```
? TEXTMERGE( lcText2 )
```

这将输出：

The time is 22:22:12, and The date is 2004-05-12

或者，使用经常被忽略的第二个参数，你可以命令TEXTMERGE()函数在内部进行重复。

```
lcText = [The date is <<DATE()>>]
```

```
lcText2 = [The time is <<TIME()>>, and <<lcText>>]
```

```
? TEXTMERGE( lcText2 , .t. )
```

这也将输出：

The time is 22:22:30, and The date is 2004-05-12

最后对TEXTMERGE()要注意的是：VFP7.0的bug将导致在某些情况下TEXTMERGE()无法正常终止字符串。解决的办法是加入CHR(0)并在TEXTMERGE()完成后移去：

```
CHRTRAN( TEXTMERGE( lcString + CHR(0) ), CHR(0), [])
```

这个bug已经在VFP8.0得到修复。

## 展望未来

这里所提及的字符处理函数并不是你在VFP里可用的所有方法。在VFP里有数打的字符处理主力。然而，即使这样强大，VFP在字符处理能力上有一个漏洞，它缺乏支持常规表达式。

对于这一点，你可能会说，“谁在乎？我怎么可能需要VFP提供的工具外的任何东西？常规表达式又是什么？”下月在这个系列的第二部分，我们将回答这些问题，并考虑通过COM使用常规表达式的方法。然后我将使用这个扩展来创建一个强大的文本拆分框架，它可以优化动态的，数据驱动的Web应用和其他以文本处理作为关键任务的应用。

# 来自 VFP 开发组的 Tips

原著:微软 VFP 开发团队

翻译: LQL.NET

---

现在VFP9公开测试版已经提供下载了 (<http://msdn.com/vfoxpro>), 下面是本月VFP开发团队成员直接向你提供的一些TIPS。

## VFP9.0 后台编译设置

VFP9.0 提供了一个后台编译设置, 可以让你对命令窗口或代码编辑窗口中输入的不完全的、不正确的语法提示进行设置, 后台编译设置默认是用带下划线的文本显示错误, 你可以把她改成用红字显示或干脆不显示, 更改后台编译设置在系统菜单 工具/选项/编辑/后台编译 中。

## 用 VFP9 的类浏览器来查看 PRG 里面的类

VFP9 的类浏览器现在允许你打开并且查看 PRG 里定义的类, 就象使用 VCX 那样。同样, 也是用 打开/添加 对话框来选择一个 PRG。

## 在 VFP9 中获取最后一个自增值

新的 GetAutoIncValue() 函数将返回一个在当前工作期或当前作用域 (函数、过程、方法) 内由自增字段生成的最后一个值。下面的例子为 Table1 建立一个 INSERT 触发器, 把插入的记录送到另一个表。用 GetAutoIncValue() 函数带参数 0, 我们就可以从 Table1 中获取最后一个自增值插入到 sys\_trans 表中。

```
* Setup database, tables and stored procedures
```

```
CLOSE DATABASES ALL
```

```
ERASE db1*.*
```

```
ERASE table1.*
```

```
ERASE sys_trans.*
```

```
CREATE DATABASE db1
```

```
CREATE TABLE sys_trans ;
```

```
    (Id I AUTOINC, cTable C(20), cValues M)
```

```
CREATE TABLE table1 (Id I AUTOINC, f2 C(10))
```

```
* Populate a few records before adding trigger
```

```

FOR i = 1 TO 5
    INSERT INTO table1 (f2) VALUES (SYS(3))
ENDFOR
CREATE TRIGGER ON 'TABLE1' FOR INSERT ;
    AS table1_insert()
TEXT TO cProcs NOSHOW
PROCEDURE Table1_Insert
    LOCAL lcResults
    CURSOR TO XML (ALIAS(), "lcResults", 1, 8+1, 1)
    LogTransaction(ALIAS(), m.lcResults)
ENDPROC
PROCEDURE LogTransaction(tcAlias, tcResults)
    INSERT INTO sys_trans (cTable, cValues) ;
        VALUES (m.tcAlias, m.tcResults)
ENDPROC
ENDTEXT
STR TO FILE(m.cProcs, "dblprocs.txt")
APPEND PROCEDURES FROM dblprocs.txt
CLOSE DATABASES ALL
* Test it
INSERT INTO Table1 (f2) VALUES (SYS(3))
CLEAR
? "GETAUTOINCVALUE()", GETAUTOINCVALUE()
? "RECCOUNT('sys_trans')", RECCOUNT('sys_trans')
? "GETAUTOINCVALUE(0)", GETAUTOINCVALUE(0)
? "RECCOUNT('table1')", RECCOUNT('table1')

```

### 在 VFP9 中清除错误

VFP9.0 提供了一个 CLEAR ERROR 命令来清除最后的错误。

```

CLEAR
ON ERROR *
ERROR 1104
ON ERROR
? [Before CLEAR ERROR:]

```

```

? [-----]
?AERROR(atest)
LIST MEMORY LIKE atest
RELEASE atest
* now clear the error
CLEAR ERROR
? [After CLEAR ERROR:]
? [-----]
?AERROR(atest)
LIST MEMORY LIKE atest
RETURN

```

### 用智能感知 (IntelliSense) 打开 WINDOWS 资源管理器

用下面的代码往你的 FOXCODE 表中加入一条记录，然后，在命令窗口中键入：ecd，再按下空格，你就可以打开 WINDOWS 资源管理器，并定位到当前目录。请把这个例程加到你的自定义 IntelliSense 脚本里。

```

LOCAL lcISENSE AS STRING
TEXT TO lcISense NOSHOW
LPARAMETERS oFoxcode
IF !INLIST(oFoxcode.Location,0)
    RETURN "ecd"
ENDIF
oFoxcode.valuetype = "V"
LOCAL lcFCCode AS STRING
lcFCCode = "RUN /n Explorer.exe " + SYS(5)+SYS(2003)
EXECSCRIPT(lcFCCode)
RETURN "~"
ENDTEXT
USE (_FOXCODE) AGAIN SHARED IN 0 ALIAS FC
INSERT INTO FC VALUES ([U], [ecd], [], [{}], ;
                        [], lcISENSE, [U], .T., DATETIME(), [FoxTips], [], [])
USE IN SELECT([FC])

```

涉及源码：407TEAMTIPS.ZIP


# 使用 Inno Setup 安装程序

原著: Rick Borup

翻译: YASUR

---

Inno Setup 是一个免费的但是功能强大而且非常适用于 vfp 程序的安装工具，它不仅为开发者提供了强劲的功能，而且为最终用户呈现出了非常专业的界面。Rick Borup 为你介绍了 Inno Setup 而且会告诉你在 vfp 中如何来使用它。这篇文章包含了一些基本概念并且带你建立一个简单的安装程序。

 们常说：天下没有免费的午餐。这可能是对的，不过却有一个免费的安装工具，它的名字叫 Inno Setup。它的免费就如蛋糕上冰块，因为 Inno Setup 是一个强大的工具，非常值得你去花点时间学习甚至会为它慷慨解囊（当然，必须是你想支持它的情况下）

Inno Setup 被看作是一个基于脚本的安装工具，因为你所写的产生安装包的命令是被存储在一个简单的文本或脚本文件中的。Inno Setup 的脚本很容易理解，因为它的语法是基于一些简单的不用加以说明的英语单词和缩写词的。你能立刻开始你的第一个脚本。

去 <http://www.jrsoftware.org/isdl.php> 下载一个软件然后安装到你的电脑上，你就可以开始 Inno Setup 之旅了（译者注：国内用户可以到一些下载网站去下载这个工具，汉化版的也有，提供一个地址：

[http://www.sron.net/SoftView/SoftView\\_1824.html](http://www.sron.net/SoftView/SoftView_1824.html)）。这篇文章中我使用的是 4.1.3 版的，但是 Inno Setup 的更新很频繁，也可能你读到这篇文章的时候已经有了新的版本了（译者注：看来作者还是比较有预见性的嘛，我翻译的时候官方网站已经到了 4.2.7 了。可能你读的时候又不一样了，呵呵）。去下载安装最新的可用版本吧。Inno Setup 的官方网站有一个完整的更新历史。在这里你可以看到每个版本的新功能。

地址：[www.jrsoftware.org/files/is4.2-whatnews.htm](http://www.jrsoftware.org/files/is4.2-whatnews.htm)

尽管都被简化为 Inno Setup，但是这个工具的正式名字是 Inno Setup Compiler。编译器，当然，的确确实非常重要的部分，因为它是创建安装包的东东。安装了 Inno Setup 以后，你的开始菜单的 Inno Setup 中有四项子菜单，从其中选择 Inno Setup Compiler 便可以打开 Inno Setup 了，除了这个编译器，Inno Setup 有自己的用来编辑和创建脚本文件的编辑器。

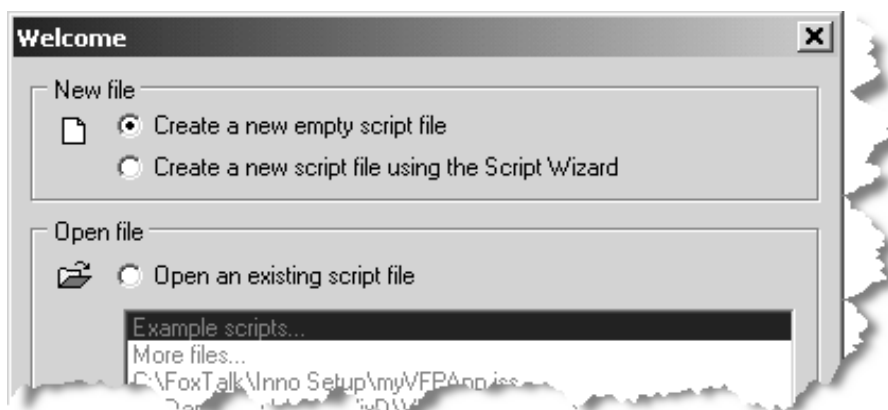
Inno Setup 附带了一个不错的帮助文件，不爽的是，这个帮助文件是一个较老的 WinHelp 格式的，而不是 HTML 格式的，不过帮助内容的质量可以弥补这点小小的不便。安装了 Inno Setup 以后，花点时间去熟悉一下这个帮助文件。作为启动，我推荐读一下 "What is Inno Setup?" 和 "How to Use" 部分的前四页。然后定位到 "Setup Script Sections"，而且在使用和学习 Inno Setup 的时候应该把这个放到手边，这一部分是你学习 Inno Setup 脚本语法的主要参考，而且你可能在阅读本文之后的内容的时候还需要使用它。

## 创建你的第一个 Inno Setup 脚本

本文将通过一个名为 **myVFPApp** 的假设的 **vfp8** 应用程序来贯穿每一个必要的步骤去完成一个非常小但功能很全的 **Inno Setup** 脚本。**myVFPApp** 包含一个名为 **myVFPApp.exe** 的可执行文件和一个名为 **readme.txt** 的自述文件。此外，安装程序还需要 **vfp** 的运行库和其它所需文件。**Inno Setup** 脚本文件是以 **iss** 为扩展名的文本文件，因为它们是简单的文本文件，所以既可以用任何文本编辑器来创建和编辑 **Inno Setup** 脚本。使用 **Inno Setup** 内建的编辑器有不少的优势，如语法着色、一键编译。

当你打开 **Inno Setup** 的时候，就会出现一个欢迎界面，它可以让你选择创建一个新的脚本文件还是打开一个已经存在的文件。如图一。选择“创建一个空的脚本文件”就会创建一个全新的文件，同事编辑器中就会出现一个空白页。

另外一个选项为使用脚本向导，这是一个内建的可以带你通过一系列的步骤创建一个新的安装脚本的工具。你应该了解一下这个脚本向导，因为当你选择[文件]-[新文件]菜单的时候，它会自动打开。在这里，我就不讲述这个向导了，如果你想知道更多，你可以自己去探索一下；在 **Hentzenwerke** 发布的 **Visual FoxPro Solutions** 的附录 d 中，你也可以找到关于脚本向导的一些讨论。



除了常见的如[新建][打开][保存][帮助]等按钮之外，在 **Inno Setup IDE** 环境中的工具栏中还有用来开始、停止编译和运行编译后文件的这些按钮。**Inno Setup** 是用节来组织脚本文件的，类似于 **INI** 文件，节的头被加上了方括号。每个节的条目是紧随着节头，每行一个条目。这种简单的结构对于读和写脚本文件都是非常容易的。

## SETUP 节区

一个 **Inno Setup** 脚本中的第一个节区是[SETUP]节，这个节区的条目适用于整个安装程序并且包含了产品名称、版本号等等。下面列出的是 **myVFPApp** 的[SETUP]节区的条目，要注意的是你可以通过在行首加一个分号(;)来在任何地方放置注释。

```
-- myVFPApp.iss --  
; 这是 myVFPApp 的 SETUP 节区，版本:1.0.0.
```

**[Setup]**

**AppName=MyVFApp**

**AppVerName=MyVFApp version 1.0.0**

**DefaultDirName={pf}\MyVFApp**

**DefaultGroupName=MyVFApp**

setup 节区条目中，等号左边的部分我们称之为键，如你所看到的，键是非常易读的英文单词或者缩写词，尽管在 setup 节区一个典型的脚本有许多键，但是只有 AppName, AppVerName 和 DefaultDir 是必需的。AppName 键是在安装过程中显示的应用程序名称。AppVerName 键类似于 AppName，但是包含了版本号。DefaultDirName 键指定了产品的默认安装目录。另外一个键是 DefaultGroupName，它指定了在开始菜单中的快捷方式的目录名。注意，在 DefaultDirName 键中目录名的前边使用了常量{pf}，Inno Setup 使用了一些类似于此的常量来标定那些在运行时被指定的真实值，{pf}常量是指用户在用户电脑上的程序安装目录，这个常量和其它常量在 Inno Setup 的帮助文档中的[HOW TO USE]--[Constants]中有详细的说明。

## Files 节区

Files 节区列出了你想安装的文件以及文件位置。这个节区的条目结构是：参数："值",你也可以在同一行放置两个或更多的参数，不过要用分号(;)分开它们，下面是 myVFApp 的 Files 节区的条目：

**[Files]**

**Source: "MyVFApp.exe"; DestDir: "{app}"**

**Source: "Readme.txt"; DestDir: "{app}"; Flags: isreadme**

Files 节区需要至少两个参数：源参数和目标参数。源参数指定了你的电脑或网络上的源文件的位置。除非指定了完整的驱动器和路径，否则将是相对于脚本文件所在的目录的。目标参数则标定了安装到用户电脑上的文件目录，注意这里使用了{app}常量作为目标参数的值，{app}常量是指用户在安装过程中所选择的目标目录。

readme.txt 文件的条目举例了 flags 的使用，flags 允许你指定文件的其它选项。isreadme 标记告诉编译器这是"readme"文件并且会指示安装程序在安装完成后提示用户去查看这个文件。这个标记在一个脚本中只能使用一次。

## Icons 节区

除了 vfp 运行库之外，你实际上已经可以在这里停止了，并且已经有了一个完整的脚本。不过一般我们习惯于创建一个快捷方式，在 Inno Setup 中，通过增加一个 Icons 节区便可以完成这个。Icons 节区的条目类似 Files 节区，包含参数和值。以下的条目可以为 myVFApp 创建一个开始菜单图标：



## [Icons]

**Name:** "{group}\MyVFPApp"; **Filename:** "{app}\MyVFPApp.exe"

再次注意常量的使用。**Name** 参数使用{group}常量，它是指在用户电脑上开始菜单中的目录组。**FileName** 参数使用了{app}常量连同应用程序可执行文件名来指定快捷方式的目标。

## VFP 运行时刻文件

VFP 运行时刻文件的条目包含在 **Files** 节区，但是这些文件需要特殊处理：**vfp7** 和 **vfp8** 的运行库安装到 **Common Files** 目录中，而 **Visual C++** 运行时刻 **dll** 文件则被安装到系统目录中(译者注：可能这里需要商榷，因为 **vfp** 的运行库相对自由，也可以安装到 **system&ssystem32** 目录中，亦可以直接安装在可执行文件当前目录下)。因为运行时刻文件的条目对所有 **vfp** 程序都是相同的，你可以在一个单独的文件中保存它们，并且如果需要的话可以很容易的将它们拷贝到脚本目录中。如下是 **vfp** 运行时刻文件的 **Files** 节区：(由于排版的需要可能会显示好几行，实际上它们每一项只有一行)

## [Files]

**Source:** C:\Program Files\Common Files\Microsoft Shared\VFP\gdiplus.dll; **DestDir:** {cf}\Microsoft Shared\VFP; **Flags:** sharedfile uninsneveruninstall restartreplace

**Source:** C:\Program Files\Common Files\Microsoft Shared\VFP\vfp8r.dll; **DestDir:** {cf}\Microsoft Shared\VFP; **Flags:** regserver sharedfile uninsneveruninstall restartreplace

**Source:** C:\Program Files\Common Files\Microsoft Shared\VFP\vfp8t.dll; **DestDir:** {cf}\Microsoft Shared\VFP; **Flags:** sharedfile uninsneveruninstall restartreplace

**Source:** C:\Program Files\Common Files\Microsoft Shared\VFP\vfp8renu.dll; **DestDir:** {cf}\Microsoft Shared\VFP; **Flags:** sharedfile uninsneveruninstall restartreplace

**Source:** C:\VFP8Distrib\System32\msvcr70.dll; **DestDir:** {sys}; **Flags:** uninsneveruninstall onlyifdoesntexist

要注意的是这里源参数使用了完整的驱动器和磁盘路径。这是因为源文件都不是在和脚本文件相同的目录，VFP 运行时刻文件在 **Common Files** 目录中，这样就可以在脚本中直接引用了。**msvcr70.dll** 文件则在系统目录中(**system&ssystem32**)。但是基于安全方面的原因，**Inno Setup** 不再允许你去直接从你的系统目录中配置文件，这种情况的解决办法是拷贝 **msvcr70.dll** 文件到另外一个目录，然后在脚本中引用。

在这个条目的 **DestDir** 中引入了另外一个新的常量，{cf}常量是指用户电脑上的 **Common Files** 目录。新的标记在这里也被引入了，**sharedfile** 标记告诉安装工具增加在你的电脑上的文件参考，**uninsneveruninstall** 告诉安装工具当卸载文件的时候不要卸掉这个文件，**restartreplace** 标记则是知会安装工具如果当被安装的文件正在使用的时候重新启动后则覆盖掉这个文件。**onlyifdoesntexist** 标记的意思是仅在用户电脑上不存在这个文件的时候才会拷贝。最后，**regserver** 标记(**vfp8r.dll** 使用)告诉安装工具要在用户电脑上注册这个 **dll**。

你可以通过简单的输入来在你的脚本中的 Files 节区的条目中加入它们，或者你可以将它们存储在一个单独的文件中，使用的使用将它们粘贴过来即可，第三个方法就是使用 Inno Setup 的 #include 指令将它们在编译的时候从一个单独的文件中引入。例如，如果你将这些条目存储到文件 VFP8Runtimes.txt 中，你就能够通过你在 Files 节区中增加一行将它们包含到你的脚本中：

```
#include "VFP8Runtimes.txt"
```

通过 #include 命令包含在你的脚本中的文件的扩展名不一定要是 iss, 尽管这样也可以。在脚本的多个节区中有相同的名称是可以，所以你不必担心而去从你所包含的文件中去掉节头。

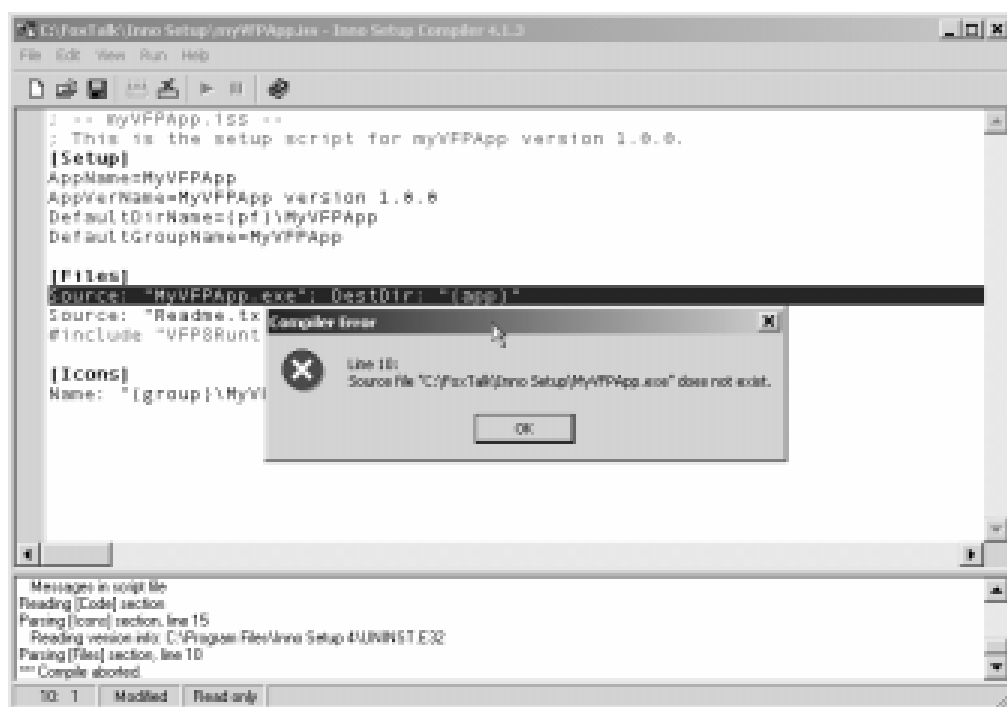
### 编译脚本

当你中被编译脚本的时候，简单的点击一下如图 2 所示的 Inno Setup 的工具栏中的编译按钮，在你第一次编译一个新的脚本之前，应该为它命名并保存到磁盘当中。



在附带的下载文件中有简单的脚本实例 myVFPApp.iss。下载的文件中还包括 VFP8Runtimes.txt 和 myVFPApp.exe, readme.txt。为了编译这个脚本，你必须确保你的电脑上安装了 vfp8, 只有这样编译器才能在你的电脑上找到这些运行时刻文件。

当你点击了编译按钮以后，Inno Setup 将会逐行编译脚本文件，并且会编辑器的底部打开一个编译器状态窗口用来记录编译的每一步和出现的错误。，如果发现一个错误，编译就会挂起，而且错误信息也会显示，同时出错行也会在编辑器中高亮标记(如图三)。



如果没有错误发生，编译器就会产生 **setup.exe** 文件。这个文件通常在一个 **Output** 的文件夹中，这个文件夹会被自动生成在脚本文件所在目录中。为了给你的用户你的程序，将这个 **setup.exe** 发给它就可以了。

Inno Setup 附带了一些实例脚本来帮助你开始使用，尽管没有关于 **vfp** 的，不过这些实例脚本是不错的学习所在，如果你将 Inno Setup 安装在了默认目录下，那么这些实例脚本就在 **C:\Program Files\Inno Setup4\Examples** 中。

## 结束语

在这篇文章中，我给你们讲述了如何为 Visual FoxPro 创建一个简单的 Inno Setup 脚本。在接下来的文章中，我将为你们展示一些用你们的脚本所做的一些很酷的事情：让你的工作更容易，提高你的用户的安装经验。

# 在 EditBox 中实现自动完成

原著: Andy Kramek & Macia Akins

翻译: Fbilo

---

在 VFP 9.0 中引入的文本框自动完成非常的酷，可惜只适用于文本框控件。但是，还有很多情况下自动完成功能能够为应用程序增加极大的价值。在这个月的专栏中，Andy Kramek 和 Marcia Akins 向你展示了怎样为编辑框实现类似的功能，它适用于 VFP 9.0 以及之前的版本。

**玛西亚：**你知道，我真的很喜欢 Visual FoxPro 9.0 中新的自动完成功能。可是他们没让编辑框也可以用这个功能令我非常遗憾。因为在我正在开发中的应用程序上刚好用得上它。

**安迪：**我觉得，我们应该能够建立一个复制了这个功能的编辑框类。如果我们了解清楚了你想要的东西的话，至少我们有这个能力。

**玛西亚：**在这个程序中的有几个表单要求用户在“备注”字段中输入自由格式的文本。当我查看数据的时候，我发现这些备注的要点都是一样的，但是由于它们是由不同的用户输入的，因此，在措辞上有着细微的区别。

**安迪：**那么你有什么想法？是要从自由格式的文本开始吗？如果只允许用户使用标准的短语，你应该使用另一个工具（比如某种多选列表）

**玛西亚：**但是这些字段必须允许自由格式的文本，因为，即使文本的某些部分包含标准的短语，可其它部分还是完全唯一的。举个例子，客服代表的备注字段经常包含这样的短语“客户对服务表示不满，因为...但是客户表示不满的就大不相同了。

**安迪：**我觉得，在这种情况下，你的字段应该是象“客户投诉”的东西，而不需要什么“标准文本”了。

**玛西亚：**象往常一样，你完全误会我的意思了（译者注：这两人是公婆俩，这句话嘛，嘿嘿，肯定是两口子斗嘴的“定式”了）。你难道没有碰到过客户因为多种原因而打电话来的情况吗？——询问关于一件没有完成的送货的问题、询问某个帐单上的某项费用是多少、什么只是表扬某个雇员的优良服务？

**安迪：**啊哈！是的——我明白了。还有什么要求？

**玛西亚：**是的。用户需要能够增加“标准片断”的功能。当他们输入的时候，他们也许会认识到他们以前已经输入过几次同样的短语了，所以希望能够把它记下来、以后就可以少敲几次键盘了。此外，可选项应该是“上下文敏感”的。

**安迪：**嗯？你说的“上下文敏感”是什么意思？

**玛西亚：**指使用的片断的类型，比如，一个客服记录字段显然与用在一个订单输入表单上的记录字段的内容不同。

**安迪：**啊哈！你的意思是只有那些与特定时间相关的片断才可用吧！

**玛西亚：**我想这就是我要说的。

**安迪：**好吧，那么我们需要的第一件东西就是一个用来储存那些“片断”的表。然后，我们需要用某种途径来显示可选项。你想储存多长的文本？

**玛西亚：**为什么要限制长度？只管用一個备注字段就是了。

**安迪：**就算可以的话，怎么显示它也会变成问题。我想我们最好使用一个弹出菜单或者列表来显示片断。很显然，这意味着一条记录只能显示在一行里。

**玛西亚：**我现在也想到这个问题了，看来这里的目的是能够让用户去建立可重用的文本“片断”。那么，实际的文本将会相当的短。现在先随便选一个字段长度值，就 120 吧！

**安迪：**看来差不多，你前面的例子“This customer was dissatisfied with the service because”（客户对服务表示不满，因为）也只有 55 个字符长。那么，我们需要一个 key 字段和一个字符型字段。

**玛西亚：**我们还需要一个字段以用于过滤可选项——记住，还需要实现上下文敏感功能。

**安迪：**啊，是的，我差点忘了这个事情。那么，需要的表结构大概就象表 1：

---

表 1、标准短语表 (stdtext.dbf)

字段	数据类型
itxtpk	I 4
ctxttext	C 120
ctxtcode	C 3

**玛西亚：**没错。下一步，我们需要决定怎样去触发列表、以及怎样显示这些片断。

**安迪：**我想用一个由编辑框的 RightClick 事件调用的弹出菜单来做。

**玛西亚：**你说你想在一个弹出菜单里面直接显示文本片断？

**安迪：**是呀，为什么不？

**玛西亚：**问题一，怎么把一个片断加入到表里面去？问题二，如果一个特定的编辑框有 30 个文本片断的话，在一个弹出菜单里面显示你不觉得太笨了吗？

**安迪：**第一个问题容易解决：给菜单增加一个“添加可选项”菜单项。第二个问题倒确实成问题。弹出菜单最后可能会超过屏幕范围。

**玛西亚：**还有，在一个太长的菜单里面找到并选择某个特定的菜单项实在是种痛苦（在任一个方法编辑窗口里面看看右键菜单，你就知道我什么意思了）。

**安迪：**那么，你的意思就是你不喜欢弹出菜单的主意了？

**玛西亚：**不是，我是说弹出菜单应该只有两个选项。第一个将打开一个表单，在这个表单上显示可用的文本片断，而另一个则将会把一个可选项添加到标准短语的列表中去。

**安迪：**那会导致产生另一个问题。用户要怎么添加一个片断？通过另一个表单吗？

**玛西亚：**那就太笨了。而且，当他们决定要添加一条文本片断的时候，他们已经输入一次了——为什么还要他们再输入一遍？就让他们高亮选中他们想要添加的文本，然后从快捷菜单中选择“添加可选项”就是了。

**安迪：**我喜欢这样。我们甚至能够让编辑框聪明到只当有高亮文本选中时才显示弹出菜单；否则，就只显示可选项列表。事实上，我们能够让它聪明到只有当编辑框有定义了的文本片断和用户选中了某些文本的时候才提供弹出菜单。在所有其它条件下，我们能起到合适的作用而不需要用户的干预。

**玛西亚：**我想我需要看看这种情况下的真值表（见表 2）。

表 2、处理右键菜单

	定义了的片断	没有定义了的片断
在编辑框中选中的文本	显示弹出菜单	将选定文本添加到片断表
在编辑框中没有选中的文本	显示可用的片断	显示“没什么可做的”消息

**安迪：**编辑框类还需要一些属性。我假定你将会考虑用不同的表来储存你的文本片断。

**玛西亚：**是的，既然自动完成文本框都允许你指定元数据表，我们不妨也复制出这种功能来。我们还将需要一些字段，以保存将被用于选择的字段的列表、以及要添加的过滤条件。需要添加的属性显示在表 3 中：

表 3、用于自动文本编辑框的自定义属性

属性名称	默认值	解释
cTextSource	stdtext	用作被显示文本来源的表别名
cFieldList	cTxtText, iTxtPK	将被光标选择的字段列表。第一个字段用于显示，第二个字段是文本表中的 ID 字段
cTextKey	cTxtCode=' ALL'	用于限制可插入文本片断列表的过滤键

**安迪：**自定义方法怎么说？很明显，我们将需要 RightClick() 事件来调用一个用于决定必须采取那种操作的方法——让我们把这个方法称作 OnRightClick() 吧。

**玛西亚：**我们还需要一个把文本添加到表里去的方法，和一个将从表中选中的文本插入到当前光标位置的方法。此外，我们还需要一个弹出表单以供用户从中选择可用的片断（见表 4）。

---

表 4、自动文本编辑框类的自定义方法

方法名称	解释
OnRightClick	当 RightClick 事件触发的时候，判定应该采取什么样的操作
AddTextToTable	将选中的文本添加到指定的表。cFieldList 中的第一个表达式应该是一个字段名，选中的文本将会被插入到该字段中去。
InsertText	将选中的文本插入到 Value 属性的当前光标位置

**安迪：**让我们先开始 OnRightClick() 方法的代码。第一个任务是判定当前文本框是否有已经为之定义了片断。基本上，我们所需要的就是从 cTextKey 这个过滤关键字属性中取出值，运行一次查询，然后根据结果得出一个逻辑型的标志：

```
IF NOT EMPTY( .cTextSource )
*** 当有自动文本数据时执行
    lcSql = "SELECT " + ALLTRIM( .cFieldList ) ;
           + " FROM " + .cTextSource ;
           + IIF( NOT EMPTY( .cTextKey ), ;
           + " WHERE " + ALLTRIM( .cTextKey ), "" );
           + " INTO CURSOR curText ORDER BY 1 NOFILTER"
    &lcSql
    llHasAutoText = (_TALLY > 0)
ENDIF
```

**玛西亚：**没错。下一步是检查编辑框中选中的文本并为此也设置一个标志。我们可以通过读取编辑框的 SelText 属性，并用 CHRTRAN() 来排除任何不需要的文本，最后设置 llHasSelected 标志：

```
*** 当有选中的文本时执行
IF .SelLength > 0
    *** 首先清除 CRLF 和 TAB 字符
    lcText = CHRTRAN( .SelText, CHR(13)+CHR(10)+CHR(9), '' )
ENDIF
llHasSelected = NOT EMPTY( lcText )
```

**安迪：**下一步，我们必须象我们的真值表（表 2）定义的那样判定要采取什么操作。这并不需要太多的



代码：

```
IF llHasSelected AND llHasAutoText
    *** 所有选项都可用，显示弹出菜单
    STORE 0 TO pnChoice
    DEFINE POPUP showtext SHORTCUT ;
        RELATIVE FROM MROW(),MCOL()
    DEFINE BAR 1 OF showtext PROMPT "Insert AutoText"
    ON SELECTION BAR 1 OF showtext pnChoice = 1
    DEFINE BAR 2 OF showtext PROMPT ["-"]
    DEFINE BAR 3 OF showtext PROMPT "Add Selected Text"
    ON SELECTION BAR 3 OF showtext pnChoice = 3
    ACTIVATE POPUP showtext
ELSE
    IF NOT llHasSelected AND NOT llHasAutoText
        *** 没有选中的文本，也没有已定义的文本片断
        lcMsg = "这个字段没有自动文本," + CRLF
        lcMsg = lcMsg + "并且什么也没选中"
        MESSAGEBOX( lcMsg, 64, "没有可做的事情" )
        *** 将选中值设置为空
        pnChoice = 0
    ELSE
        *** 只有一个选项可用，设置好 pnChoice
        pnChoice = IIF( llHasAutoText, 1, 3 )
    ENDIF
ENDIF
ENDIF
```

**玛西亚：**我注意到这个 pnChoice 是要由用户从菜单上选择什么来设定的，这就意味着它必须在调用例程中被声明为私有型（PRIVATE）变量。

**安迪：**是的，它是这样。这里我没有展示所有变量的声明，既然你提到了它，顺便指出一下，CRLF 是在代码中被显式的定义为 "CHR(13) + CHR(10)"。

**玛西亚：**最后一步是根据 pnChoice 的值执行操作，象这样：

```

DO CASE
    CASE pnChoice = 1
        *** 用户想要从自动文本中选取一条文本片断
        *** 将要使用的游标的名称传递给表单
        DO FORM frmautotext WITH 'curtext' TO lcText
        IF NOT EMPTY( lcText )
            *** 从表单返回了一些文本，因此将它插入到字段中去
            .InsertText( lcText )
        ENDIF
    CASE pnChoice = 3
        *** 将选中的文本添加到自动文本列表中去
        .AddTextToTable( lcText )
    OTHERWISE
        *** 什么都不用做
        lcText = ""
ENDCASE

```

**安迪：**另一个小麻烦是将选中文本插入到编辑框中光标位置的代码。这是由 `InsertText()` 方法处理的，该方法从弹出的表单接收到文本参数：

```

WITH This
    *** 忽略尾部的空格
    tcText = ALLTRIM( tcText )
    lcText = ALLTRIM( NVL(.Value, "") )
    lnLen = LEN( lcText )
    *** 取得插入点的位置
    lnPos = .SelStart

DO CASE
    CASE lnPos <= 1
        *** 当前出于起始位置
        *** 因此，在文本的后面加上一个空格
        .Value = tcText + " " + lcText
        lnNewPos = LEN( tcText ) + 1
    CASE lnPos < lnLen

```

```

    *** 在中间，在文本的两端都加上一个空格
    .Value = LEFT( lcText, lnPos ) + " " + tcText
    + " " + SUBSTR( lcText, lnPos + 1 )
    lnNewPos = lnPos + LEN( tcText ) + 2
  OTHERWISE
    *** 在最右边位置
    *** 在插入文本前加上一个空格
    .Value = lcText + " " + tcText
    lnNewPos = lnPos + LEN(tcText) + 1
  ENDCASE

  *** 重新定位插入点
  .Refresh()
  .SelStart = lnNewPos
ENDWITH

```

**玛西亚：**好。现在我们必须处理将一个文本片断添加到表中去的任务了。这并不太难，因为我们在表中使用了一个自动增长的整型，并且我们可以从 cFieldList 属性中取得文本和 ID 字段的名称、从 cTextKey 属性取得识别码字段的名称。这样，不管自动文本片断的来源是哪个表，代码都将正常运转。我喜欢数据驱动的事情，象这样：

```

*** 将文本插入到表中
lcTable = This.cTextSource

*** 目的字段是在 fieldlist 中的第一个词
lcField = GETWORDNUM( This.cFieldlist, 1, ',' )

IF NOT EMPTY( This.cTextKey )
  *** 识别码字段是在 cTextKey 属性中的第一个词
  lcField = lcField + "," + ALLTRIM( ;
    GETWORDNUM( This.cTextKey, 1, '=' ) )

  *** 处理嵌入的引号
  *** (首先将智能引号转换成标准的引号)
  *** 译者注：这里说的“智能引号”这个词是从词典中来的，

```

\*\*\* 从字符来看，Chr(147)、Chr(148)分别是中文引号“和”

```
tcText = CHRTRAN( tcText, CHR(147)+CHR(148), CHR(39) )
```

\*\*\* 删除尾部的空格，并加上封闭的定界符

\*\*\* 注意：用 '[]' 来避免嵌入引号的问题

```
tcText = "[" + RTRIM( tcText ) + "]"
```

\*\*\* 文本识别码的值是在 cTextKey 中的第二个词

```
tcText = tcText + "," + ALLTRIM( ;  
GETWORDNUM( This.cTextKey, 2, '=' ) )
```

ENDIF

\*\*\* 生成并执行插入语句

```
lcSql = "INSERT INTO " + lcTable + " (" + lcField + ") ;  
VALUES (" + tcText + ")"  
&lcSql
```

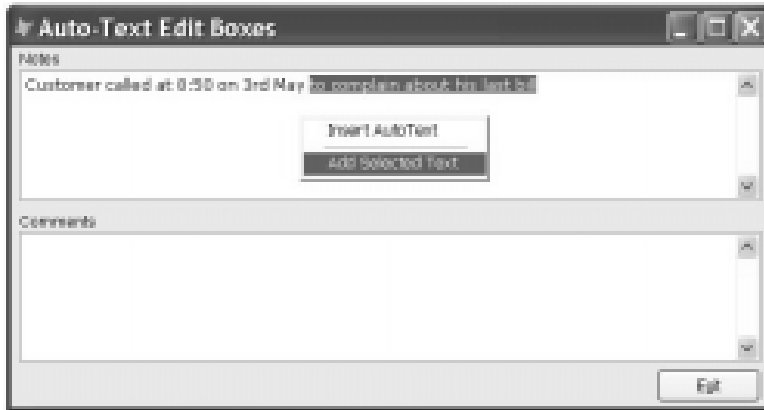
**安迪：**好，它的正确运行依赖于这些属性的设置正确与否，不过那是设计时的问题——我想做这个应该不难。

**玛西亚：**当然了！我们必须假定程序员会把这些事情做对，就像我们必须假定他们会测试他们的代码一样。

**安迪：**现在就看你的了！我们做完了吗？现在我们可以测试它吗？

**玛西亚：**是的。图 1 和图 2 显示了一个用上了这个自动文本编辑框类的小表单。

注意：尽管这个程序是在 VFP 8.0 中设计和测试的，但是只要在元数据表中把作为主键的自动增长字段去掉，这个类就可以在 7.0 中使用，再去掉类定义中的 GetWordNum() 函数而代之以 foxtools.fll 中的类似函数就可以在 6.0 中使用了。



(图一)



(图二)

涉及源码：407KITBOX.ZIP