PDF-Power to the People

Lisa Slater Nicholls

Die Ausgabe in PDF-Dokumente ist eine der am häufigsten nachgefragten Erweiterungen in Visual FoxPro, steht aber von Haus aus nicht zur Verfügung. Lisa zeigt Ihnen in diesem Artikel, wie Sie mit allen Versionen von FoxPro PDFs generieren und demonstriert einige Techniken von VFP 9.0, sowie einige Druckfeatures, die das Erstellen von PDFs einfacher gestalten als in allen früheren Versionen. Außerdem behandelt Sie die Notwendigkeit der „Anwendungstransparenz" und stellte einen Code zur Verfügung, der „on the Fly" GhostScript verlässlich installiert, eine Komponente der Free Software Foundation.

Wenn zu den Anforderungen an die Ausgabe auch exakte elektronische Versionen gedruckter Dokuments gehören, ist heute PDF (Portable Document Format) der beste Standard. Das Format Microsoft Document Imaging (MDI), das auf TIFF (Tagged Image File Format) basiert, ist nur dann eine Alternative, wenn der Empfänger über den Microsoft Office Document Imaging Reader oder über ein anderes Programm verfügt, das TIFF- und MDI-Dateien lesen kann. Der Adobe® Reader® für PDF ist kostenlos, sehr weit verbreitet und für jede Plattform verfügbar.

- Die ReportListener von VFP 9.0 können mehrseitige TIFF-Dokumente sowie andere Grafikdateiformate erzeugen, aber Ihre Anwender könnten immer noch fragen, weshalb Ihre FoxPro-Anwendungen nicht wie andere Software-Werkzeuge PDF-Dokumente erstellen können. Tatsächlich können FoxPro-Anwendungen PDFs erstellen – und dabei auf die gleichen Ressourcen zurückgreifen wie andere Werkzeuge. Dies werde ich Ihnen in diesem Artikel zeigen.
- Diese Ressourcen werden nicht zusammen mit Visual FoxPro oder anderen Microsoft-Produkten ausgeliefert. Das Vertrauen auf „Free Software Movement" und Open Source-Komponenten sind eine mögliche Erklärung für das Fehlen der PDF-Ausgabe. Dass es sich bei PDF um einen de Facto-Standard handelt, der von einem anderen großen Hersteller entwickelt wurde, ist eine weitere Möglichkeit.
- Ihre Anwender interessiert diese Strategie nicht, so dass unsere Spekulationen über die Gründe eigentlich nutzlos ist. Aus der Sicht des Anwenders sind *Sie* der Hersteller. Worauf warten Sie also?

Was Sie wollen

Die Anwender wollen einfach PDFs haben. Als professioneller Entwickler haben Sie noch weitere Anforderungen:

- Durch das Einfügen dieses Features in Ihre Anwendungen sollen keine Lizenzzahlungen anfallen.
- Das Feature soll mit allen FoxPro-Berichten funktionieren.
- Der Anwender soll nicht merken, dass an dem Vorgang externe Programme beteiligt sind, und es soll kein Eingriff des Anwenders in den Prozess erforderlich sein, auch nicht während der Installation. Dieses Charakteristikum nenne ich *Anwendungstransparenz*.

Kein Witz: die Anwendungstransparenz ist ein besonderer Reiz. Sie könnte als ein Satz Charakteristika definiert werden, durch die Ihre komponentenbasierte Softwarelösung dem Anwender so erscheint, als wäre sie von Ihnen von Grund auf nur für ihn erstellt worden. Wenn Sie eine FoxPro-Anwendung erstellen, können Sie die Komponenten auf

unzählige Arten aufwerten, aber manchmal kann die Anwendungstransparenz den Unterschied machen zwischen der Akzeptanz der Anwender für Ihre Anwendung auf der einen Seite und der Ablehnung der Anwendung auf der anderen Seite.

Ich lebe in Las Vegas. Ich gehe daher eine Wette an, dass Sie alle Anforderungen erfüllen können und ich erhöhe den Einsatz noch. Mit kleinen Anpassungen sollten die gleichen Techniken für jede Ausgabe funktionieren, die mit den Anweisungen im Befehl SET PRINTER TO funktionieren, nicht nur für Ergebnisse in den Formaten FRX oder LBX. Die Techniken sollten außerdem mit jeder Version von FoxPro funktionieren, auch mit zeichenbasierten Berichten von FoxPro für DOS und UNIX sowie mit anderen Druckausgaben.

Kommerzielle Alternativen

Ich empfehle sicher nicht, dass Sie die Methode für das Generieren von PDFs ändern, wenn Sie bereits mit einer Methode arbeiten, die Sie mögen. Es gibt verschiedene Add-Ons für Visual FoxPro, die diese Aufgabe sehr gut erledigen.

Während der Betaphase von VFP 9.0 habe ich mit vielen Herstellern von Werkzeugen zusammengearbeitet, während diese ihre Produkte neu erstellten, um mit Hilfe der neuen Architektur des Berichtssystems die PDFs in VFP 9.0 schneller und besser zu erstellen. Die Ergebnisse waren großartig – diese Super-Anwender bestätigten ums, dass die Lösung, an der wir arbeiteten, richtig war. VFP vereinfacht es und macht es interessanter, neue Berichtserweiterungen zu erfinden und diese nahtlos in Anwendungen einzufügen.

Jetzt fragen Sie sich eventuell, weshalb Sie nicht einfach ein Produkt eines Drittherstellers nehmen sollen, um mit VFP PDFs zu generieren.

Ich weiß es nicht. Vielleicht haben Sie verschiedene Antworten auf diese Frage, von den Kosten bis hin zu Einschränkungen bei der Lizenzierung und der Weitergabe. Sollte dies auch bei Ihnen der Fall sein, stellt Ihnen dieser Artikel eine weitere Möglichkeit vor, die Sie in Ihre Überlegungen mit einbeziehen sollten.

Betrachten Sie Ihre Möglichkeiten

Sie können PDFs auf die gleiche Weise generieren, auf die Sie die meisten Programmierprobleme lösen, also wahlweise mit Low Level- oder mit High Level-Techniken.

Für die Arbeit mit Low Level-Techniken halte ich es für sinnvoll, Ihnen das PDF-Format sowie die Ausgabe des Berichts in diesem Format zu beschreiben, bei der immer ein Objekt ausgegeben wird. Wenn Sie es wünschen, können Sie in Visual FoxPro 9.0 diese Lösung einsetzen. Verwenden Sie einfach die Vorbereitung der Objekte in einer vom ReportListener abgeleiteten Klasse und generieren Sie während der Laufzeit des Berichts den entsprechenden PDF-Code.

Wenn Sie bereits mit dem PDF-Format vertraut sind, können Sie mit dieser Methode eventuell bereits ausreichend gute Ergebnisse erzielen, allerdings müssen Sie für die unterschiedlichen vorbereiteten Berichtsobjekte noch verschiedene Probleme lösen. *Wenn Sie ein Produkt oder Utility entdecken, das PDF generiert, aber Einschränkungen mit verschiedenen Ausgabetypen aufweist, oder das eine lediglich eine abgespeckte Version seiner Quelle erstellt, ist dies vermutlich der verwendete Lösungsweg.*

Mit High Level meine ich, dass Sie PDF-Dokumente aus bereits fertig gestellten Komponenten zusammensetzen: eine Post Script-Datei sowie ein PostScript-Interpreter. PostScript ist die Seitenbeschreibungssprache, auf dem das PDF-Format aufbaut. Ein PostScript-Interpreter kann das Dokument untersuchen, alle erforderlichen Referenzen in einer einzelnen komprimierten PDF-Datei bündeln und dabei optional Fonts einbetten.

Leistungsfähige PostScript-Druckertreiber generieren auf einfache Weise PostScript-Dokumentendateien. Diese Treiber werden für jede Version von Windows ausgeliefert. Ich verwende die freie Software-Komponente von GhostScript als Interpreter und führe anschließend die Konvertierung in PDF durch.

GhostScript ist ein stabiles, gut supportetes und ausgereiftes Produkt. Es ist ausreichend skalierbar und wird deshalb auf vielen Websites eingesetzt, die eine PDF-Generierung anbieten und „on the Fly" PDF-Ausgaben generieren. *Wenn Sie ein kommerzielles Utility finden, das eine leistungsfähige und vollständige PDF-Erstellung anbietet, ist GhostScript sehr häufig die darunter liegende Technologie.*

Ich glaube, Sie können meinen Standpunkt erkennen. Unabhängig davon, wie sehr Sie es genießen, sich die Hände mit den Interna des Codes zu beschmutzen, empfehle ich den Einsatz von High Level-Produkten: verwenden Sie PostScript-Treiber, um eine qualitativ hochwertige PDF-Ausgabe zu erhalten.

Weshalb nicht XSL:FO verwenden?

Zusätzlich zu den allgemein verfügbaren Techniken für die Generierung von PDF haben Sie in VFP 9.0 noch eine weitere Möglichkeit für Berichte und Etiketten: nehmen Sie die XML-Ausgabe von VFP, in der die Ausgabe vollständig beschrieben wird, und setzen Sie eine XSLT-Transformation ein, um das XML in XSL:FO zu mappen, weitgehend wie der HTML Listener eine XSLT-Transformation verwendet, um das XML in HTML zu mappen. XSL:FO ist ein XML-Dialekt, der eine sehr detaillierte Seitenbeschreibungssprache darstellt. Die Transformation von VFP Report XML nach XSL:FO könnte wie folgt aussehen:

```
<xsl:template match="/Reports/VFP-Report">
<fo:root>
<fo:page-sequence
  master-reference="default-page" format="1">
 <xsl:attribute name="fo:initial-page-number">
  <xsl:value-of select="./Data/PH[1]/@id"/>
 </xsl:attribute>
 <xsl:for-each select="./Data/PH">
 <fo:flow flow-name="xsl-region-body">
  <fo:block break-after="page" top="0in"
   width="100%">
   <!-- contents of a page described here -->
  </fo:block>
 </fo:flow>
 </xsl:for-each>
</fo:page-sequence>
</fo:root>
</xsl:template>
```

- Wenn Sie die Ergebnisse dieser Umwandlung durch einen Prozessor laufen lassen, der XSL:FO versteht, wird als Ausgabetyp PDF produziert.
- XSL:FO wäre eine ideale Methode, wenn die Microsoft XML-Komponenten einen Prozessor enthalten würden, der mit XSL:FO umgehen könnte. Da dies nicht der Fall ist, benötigen Sie weiterhin eine externe Komponente, beispielsweise das Apache XML FOP Project (http://xml.apache.org/fop/).
- Jeder zur Verfügung stehende XSL:FO Prozessor bringt seine eigenen Einschränkungen, seine eigenen Installationeroutinen und seine eigenen Probleme bei der Ausführung mit sich, sowie eine eigene Lernkurve für den VFP-Entwickler. Ich habe bei meinen Versuchen keinen XSL:FO Prozessor gefunden, der weniger Probleme verursachte als die Minimalanforderungen von GhostScript.

Jeder zur Verfügung stehende XSL:FO Prozessor bringt seine eigenen Einschränkungen, seine eigenen Installationeroutinen und seine eigenen Probleme bei der Ausführung mit sich, sowie eine eigene Lernkurve für den VFP-Entwickler. Ich habe bei meinen Versuchen keinen XSL:FO Prozessor gefunden, der weniger Probleme verursachte als die Minimalanforderungen von Ghost Script.

Wenn Sie diese Technik bereits getestet haben und mit diesen Versuchen gescheitert sind, könnte dies daran liegen, dass Sie einige Probleme der Anwendungstransparenz nicht lösen konnten. Dieser Artikel zeigt Ihnen, wie es geht und stellt Ihnen als einfache Referenz den dafür erforderlichen Code vor, der in einer von ReportListener abgeleiteten Klasse gekapselt ist. Sollten bei Ihnen weitere FoxPro-spezifische Probleme auftreten, sollte zusätzlicher Code in der Klasse PDFListener auch diese lösen. Falls Sie sich unsicher sind, OpenSource oder freie Softwarekomponenten in eine Anwendung aufzunehmen, denken Sie daran, dass unterschiedliche Komponentenhersteller unterschiedliche Lizenzvereinbarungen treffen. Sehen Sie sich einmal das folgende funktionierende Beispiel an.

Wie Sie erhalten, was Sie wünschen

Ich liste Ihnen hier die Schritte auf, die bei der High Level-Lösung für die PDF-Generierung erforderlich sind. Diese Schritte sind alle in der von ReportListener abgeleiteten Klasse enthalten, die Sie in PDFLISTENER.PRG finden. Aus Bequemlichkeitsgründen erfordert keiner der Schritte für seine Aufgabe einen ReportListener:

1. Stellen Sie fest, ob GhostScript in der Umgebung zur Verfügung steht. Falls erforderlich, installieren Sie es im Hintergrund.
2. Stellen Sie fest, ob Ihre Druckereinrichtung zur Verfügung steht. Falls erforderlich, installieren Sie den Treiber automatisch, in der Regel an Ihre Anwendung gebunden.
3. Weisen Sie VFP an, den Treiber zu verwenden.
4. Generieren Sie Ihren Bericht oder in eine PostScript-Datei und speichern Sie den Dateinamen. Diesen Schritt können Sie mehrfach ausführen, wenn Sie mehr als einen Bericht erstellen.
5. Um die gute Form zu wahren, stellen Sie die Druckerumgebung von VFP wieder her.
6. Erstellen Sie eine Befehls-Textdatei, in der die PS-Dateien aufgelistet werden, die konvertiert werden sollen.
7. Rufen Sie GhostScript mit der Befehlsdatei, dem Namen der Ausgabedatei sowie den anderen zur Verfügung stehenden Befehlszeilenparametern auf.

Die Schritte 1 und 2 können Sie jederzeit vornehmen. In der Regel sollten diese Teil der normalen Setupprozeduren Ihrer Anwendung sein. PDFListener enthält sie in der folgenden Methode als Teil der Startprozedur für eine Berichtsausführung:

```
PROCEDURE LoadPrinterInfo()
   IF NOT THIS.VerifyGSLibrary()
      RETURN .F.
   ENDIF
   IF NOT THIS.VerifyPrinterSetup()
      RETURN .F.
   ENDIF
   IF NOT
THIS.AdjustVFPPrinterSetups()
      RETURN .F.
   ENDIF
ENDPROC
```

Das Einrichten

Die ersten beiden Aufrufe in dieser Methode sind globale Methoden, die Sie in jeder Setup-Prozedur separat aufrufen können, um sicherzustellen, dass der GhostScript- und der PostScript-Treiber zur Verfügung stehen. Vermutlich verfügt Ihre Setup-Anwendung über die erforderlichen Rechte, um ein Verzeichnis zu erstellen und eine Druckereinrichtung zu installieren.

Sie können diese Methoden am Beginn jedes Berichtslaufs aufrufen, so wie es der PDFListener tut. Dies ist in dem Fall sinnvoll, dass ein Anwender eine dieser externen Komponenten deinstalliert hat. Verfügt der aktuelle Anwender über die erforderlichen Rechte, wird eine fehlende Komponente neu installiert. Verfügt der Anwender nicht über diese Rechte und fehlt eine Komponente, liefert die Methode PDFListener.Run Re-

ports() .F. zurück. Ihre Anwendung sollte in diesem Fall dem Anwender eine sinnvolle Meldung anzeigen, genau als wenn eine andere erforderliche Datei der Anwendung fehlt, beispielsweise eine änderbare FRX oder eine Tabelle.

Einige Entwickler sind auf die Idee gekommen, Code zu erstellen, der einen anderen Anwender imitiert, um diese Situation ohne eine Fehlermeldung der Anwendung zu lösen. Ich halte dieses Vorgehen für übertrieben. Außerdem ist es ein potentielles Sicherheitsrisiko, für diesen Zweck ein Login mit Administratorrechten in der Anwendung zu speichern.

GhostScript „on the Fly" installieren

VerifyGSLibrary ist die Methode, die prüft, ob die GhostScript-Komponenten vorhanden sind. Sie verwendet dafür eine Eigenschaft der Klasse (GSLocation), um festzustellen, wo die Komponenten gesucht werden sollen. Wenn Sie die Eigenschaft nicht explizit einstellen, verwendet die Klasse eine Methode mit Namen GetDefaultGSLocation, um das Verzeichnis zu finden. Wie beschrieben versucht die Methode, ein entsprechend benanntes Unterverzeichnis unterhalb der ausgeführten Anwendung oder unterhalb des Verzeichnisses der Klassenbibliothek zu finden, wenn Sie es nicht in die Anwendung aufgenommen haben. Ziel ist, ein potentielles Verzeichnis zu finden, der garantiert vorhanden ist, unabhängig davon, ob es aktuell die GhostScript-Dateien enthält oder nicht.

Nachdem das Verzeichnis festgestellt wurde und wenn die Dateien nicht vorhanden sind, erstellt VerifyGSLibrary falls erforderlich ein Unterverzeichnis. Die Dateien werden auf die Festplatte geschrieben. Dafür wird die absurd einfache und schnelle FoxPro-Methode verwendet, die Dateien aus Memofeldern herausschreibt. In einem späteren Abschnitt werde ich erklären, wie die richtigen Dateien in die Tabelle INSTALL.DBF geschrieben werden.

Um GhostScript zu installieren, müssen Sie keine Registryaufrufe durchführen und Sie müssen keine externen Programme ausführen. Wenn Ihre Anwendung den Ort der Dateien kennt, haben Sie alles eingerichtet. Weder die Installation noch der Einsatz von GhostScropt sind von irgendwelchen Windows-Tricks abhängig.

Zum korrekten Einsatz von GhostScript gehört allerdings die Einhaltung der Lizenzbedingungen. Obwohl diese Lizenzbedingungen nicht einmalig sind, sollten Sie sich die Zeit nehmen und sich selbst mit diesen Bedingungen und Ihren unterschiedlichen Möglichkeiten vertraut machen.

Freie Software verantwortungsbewusst einsetzen

Ich verwende die GNU-Distribution von GhostScript, der unter der GPL-Lizenz steht. Es stehen auch eine alternative Aladdin GhostScript-Distribution sowie eine kommerzielle Distribution zur Verfügung. Auf der Website http://www.cs.wisc.edu/~ghost/ finden Sie Details zu den unterschiedlichen Distributionen sowie Hinweise, wie Sie diese erhalten.

Ich lege den GhostScript-Dateien immer eine vollständige Kopie der GPL-Lizenz bei, unter der artofcode LLC, der Inhaber des Copyright, GNU GostScript vertreibt. Da ich in meiner INSTALL.DBF nicht den vollständigen Quellcode beilege, liefere ich das schriftliche Angebot mit, gegen Erstattung der Versandkosten auf Anfrage den Quellcode nachzuliefern. Diese Themen finden Sie in einer Textdatei mit Namen LICENSE.TXT, die zusammen mit den GhostScript-Dateien aus der INSTALL.DBF auf die Festplatte kopiert wird.

Beachten Sie, dass diese Lizenzbedingungen, mich nicht zwingen, auf Anfrage den Quellcode meiner gesamten Anwendung herauszugeben. Aus Sicht der GPL, die von der Free Software Foundation (http://www.gnu.org/) definiert wurde, *verknüpfe* ich meine Anwendung mit GhostScript, leite aber meine Anwendung nicht von GhostScript ab. Ich verbinde meinen Code nicht mit dem Quellcode von GhostScript und ich verändere ihn auch nicht. Die Kriterien:

- Ein Endanwender könnte die ausführbare Datei von GhostScript von außerhalb meiner Anwendung aufrufen und
- wenn meine Anwendung gelöscht wird, die GhostScript-Dateien aber bestehen bleiben, funktioniert GhostScript weiterhin.

Ihren Treiber festlegen

PDFListener.VerifyPrinterSetup() ist die Methode, die Ihre Druckereinrichtung prüft und sie gegebenenfalls installiert. Die Klasse verwendet die globale Eigenschaft PSDriverSetupName, die den Namen enthält, den der Anwender für den installierten Drucker sieht. Wenn Sie diesen Wert angeben, prüft die Methode Ihren Wert mit dem Druckertreiber, von dem sie annimmt, dass er verwendet werden soll, und nutzt dafür ein neues Argument der Funktion APRINTERS(), um den Namen des Druckers zu erhalten. Sie können einen Namen wie „ABC Company SuperDriver“ verwenden, so wie es auch viele PDF-Utilities tun, den Sie dem Setup Ihrer Installation hinzufügen, ohne dass die Zuordnung zu PostScript offensichtlich wird:

```
PROCEDURE PSDriverSetupName_Assign(tcVal)
IF VARTYPE(tcVal) = "C" AND NOT EMPTY(tcVal)
   LOCAL laSetups[1], liIndex, ;
         llFoundAndNotAppropriate
   FOR liIndex = 1 TO APRINTERS(laSetups,1)
     IF ( UPPER(laSetups[liIndex,1]) == ;
          UPPER(ALLTR(THIS.PSDriverSetupName)) );
         AND ;
        ( NOT (UPPER(laSetups[liIndex,3]) == ;
           UPPER(ALLTRIM(DRIVER_TO_USE)) ) )
           llFoundAndNotAppropriate = .T.
           EXIT
     ENDIF
   ENDFOR
   IF NOT llFoundAndNotAppropriate
     THIS.PSDriverSetupName = ALLTRIM(tcVal)
   ENDIF
ENDIF
ENDPROC
```

Wie Sie sehen, ermöglicht es Ihnen die Assign-Methode, jeden von Ihnen gewünschten Druckernamen zuzuordnen, auch wenn die Druckereinrichtung nicht existiert, nicht aber einen Namen, der bereits für ein Druckersetup verwendet wird und der einen anderen als den von Ihnen benötigten Druckertreiber verwendet. Diese Bedingung behandelt den unwahrscheinlichen Fall, dass jemand den exakt gleichen Namen für einen Nicht-PostScript-Treiber verwendet hat.

Mit Hilfe des internen Standardwerts oder Ihrem angegebenen Druckersetupnamen führt PDFListener.VerifyPrinterSetup() ein einzigen Aufruf von PRINTUI.DLL aus, einer Windows-Komponente im Verzeichnis System32. Mit einer Codezeile erledigt dieser Aufruf verschiedene Aufgaben:

- Installieren der erforderlichen Basisdateien,
- Zuordnen dieser Dateien zu Ihrem benannten Drucker,
- Dieses Setup für Ihre Anwendung zur Verfügung stellen,
- Markieren des Druckersetups als mehrfach benutzbar, für den Fall, dass Sie diese Anwendung auf einem Druckserver einsetzen.

Der Aufruf sieht folgendermaßen aus:

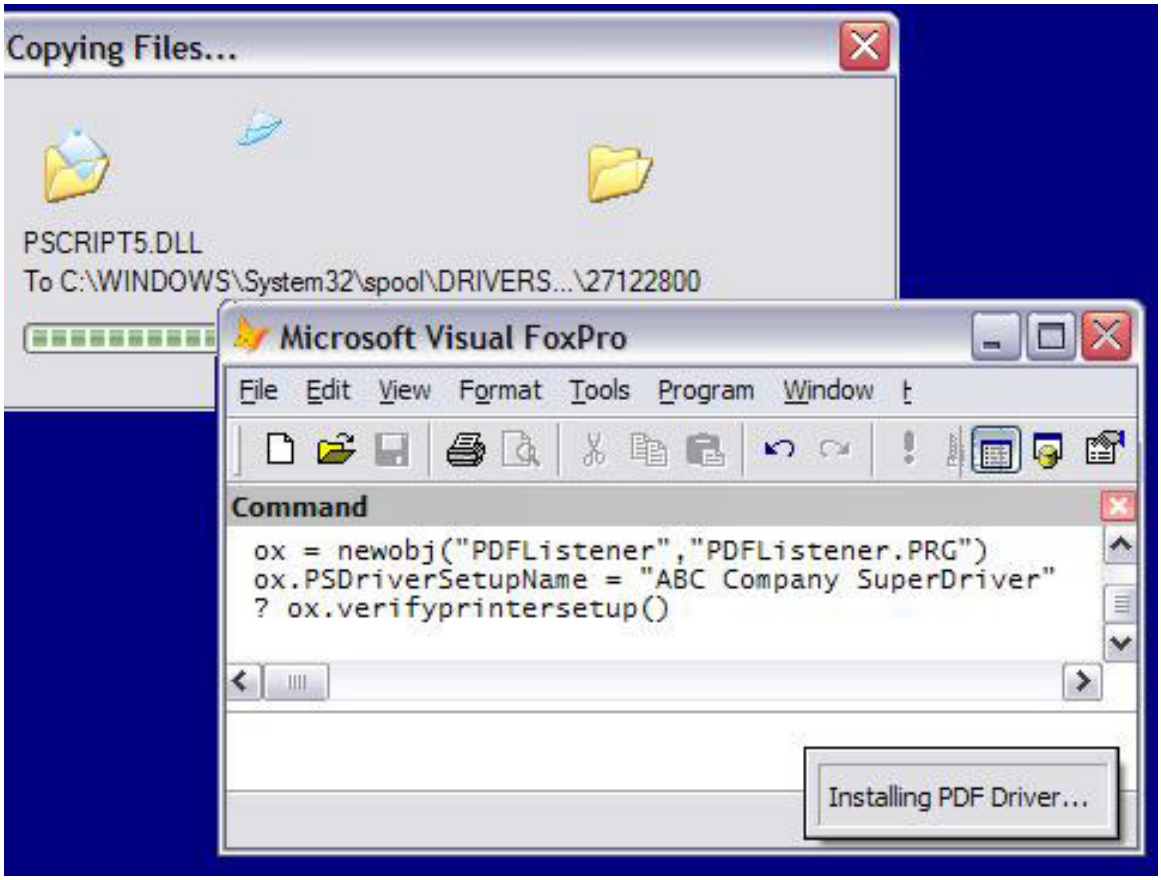
```
THIS.DoStatus(INSTALLING_DRIVER_LOC)
lcCmd =  ;
 [%windir%\\system32\\rundll32.exe ] + ;
 [printui.dll,PrintUIEntry /if /b ] + ;
 ["] + THIS.PSDriverSetupName + ["] + ;
 [ /f %windir%\\inf\\ntprint.inf /r ] + ;
 [ "lpt1:" /m "] + ;
 DRIVER_TO_USE + [" /Z]
llReturn = THIS.oWinAPI.ProgExecute(lcCmd)
THIS.ClearStatus()
```

Das sieht furchtbar aus, aber Sie können auf der Kommandozeile von DOS den folgenden Befehl ausführen, um festzustellen, was die ganzen Optionen bedeuten:

```
%windir%\\system32\\rundll32.exe printui.dll, _
   PrintUIEntry /?
```

Sie werden bemerken, dass ich den Port als LPT1 zuordnete. Die ursprüngliche Portzuordnung ist nicht relevant, da Sie VFP immer mitteilen, dass die PostScript-Ausgabe in eine Datei erfolgt.

Der oben stehende Aufruf von ReportListener.DoStatus() gibt Ihnen die Möglichkeit, dem Anwender eine Meldung anzuzeigen, die ihn über Ihre Aktivität informiert, während der Windows-Aufruf einen Moment benötigt, um die Druckertreiberdateien zu kopieren (siehe **Abbildung 1**). Wenn Sie dem Anwender kein Feedback geben möchten, verwenden Sie den Schalter ReportListener.QuietMode, um die Nachricht zu unterdrücken. Das Ergebnis ist ein Druckersetup für Ihre Anwendung (siehe **Abbildung 2**).



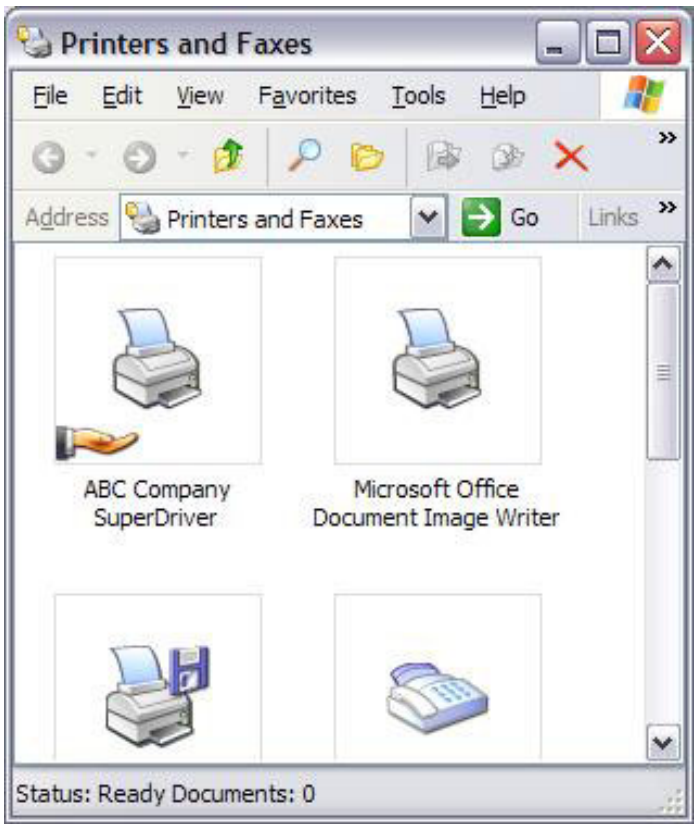


Abbildung 1 und 2. Die Standard-Systemdialoge von Windows sowie Ihre eigene Statusnachricht von ReportListener sind die maximale Oberfläche, die ein Anwender sieht, wenn Sie PDFListener einsetzen, um einen Drucker „on the Fly" zu installieren. Anschließend sieht er nur noch die Druckerauswahl Ihrer Anwendung.

Welchen Druckertreiber sollten Sie wählen? In PDFListener ist mit #DEFINE der DRIVER_TO_USE als „Apple Color LaserWriter 12/600" angegeben, aber es funktioniert nahezu jeder PostScript-Drucker, vorausgesetzt, dass er von der Windows-Version unterstützt wird, auf der Sie Ihre Anwendung installieren. Die Systemdatei NTPRINT.INF, die in der oben stehenden Befehlszeile verwendet wurde, bietet Ihnen eine große Auswahl. Lesen Sie http://www.cs.wise.edu/~ghost/doc/printer.htm, um eine Liste potentieller Druckerprobleme zu finden, obwohl in den meisten Fällen jeder PostScript-Drucker funktioniert, der Ihre benötigte Auflösung und Seitengröße sowie falls erforderlich Farbe unterstützt.

Der Einsatz von PRINTUI.DLL für die programmgesteuerte Installation eines Druckertreibers funktioniert nicht mit allen Versionen von Windows; die Version funktioniert mit XP und mit einigen Änderungen auch unter Windows 2000. Details finden Sie im Microsoft KB-Artikel 189105. Dies ist aber nicht die einzige Möglichkeit, einen Druckertreiber programmgesteuert zu installieren, wenn Sie Windows Shell Scripting verwenden können. Prüfen Sie Ihr Windows-Verzeichnis, ob dort Dateien mit dem Format PRN*.VBS als Dateinamen vorhanden sind, und Sie werden einen Host mit Skript-Utilities für die Druckersteuerung finden.

Erinnern Sie sich noch, dass ich ausgeführt habe, dass diese Methode mit allen Versionen von FoxPro funktionieren würde? Bevor FoxPro-Programmierer die Druckertreiber von Windows verwendet haben, setzten wir _GENPD ein, um Druckeranweisungen für die zeichenbasierte Ausgabe von FoxPro zu verwenden. Die standardmäßige _GENPD-Anwendung wurde mit einem perfekten PostScript-Mechanismus ausgeliefert, einschließlich eines grundlegenden Sets mit PostScript-Fontbeschreibungen. Sie können die Befehle SET PDSETUP TO <your PostScript setup> und SET PRINTER TO <filename> ausführen, und schon schreiben Ihre DOS-Anwendungen jedes Bit genauso gut in PostScript-Dateien wie Sie sie von Windows erhalten.

Einige GhostScript-Features werden wir ignorieren

Wenn Sie mit GhostScript bereits vertraut sind, fragen Sie sich jetzt vielleicht, weshalb ich RedMon nicht behandle, ein Utility für die Portumleitung für den Einsatz mit GhostScript unter Windows. RedMon bietet theoretisch ein „transparentes Drucken von PostScript", jedenfalls laut der Homepage auf http://www.cs.wisc.edu/~ghost/redmon/. Es kann eingesetzt werden, um PostScript durch GhostScript zu leiten, um während des Berichtslaufs PDFs zu erstellen. Diese Prozedur würden den Schritt nach der Verarbeitung überflüssig machen, bei dem die ausführbare Datei von GhostScript aufgerufen wird.

Transparenz mag bei dieser Laufzeitverarbeitung sein, ein transparentes Anwendungs-Setup ist aber nicht die Frage. Wenn Sie auf der Website die Liste der bekannten Probleme lesen oder das Web durchsuchen, um Unterstützung bei der Installation des RedMon Utilities zu erhalten, werden Sie schnell feststellen, dass das Setup und die Konfiguration von RedMon extrem schwierige Aufgaben darstellen. Es mag sogar unmöglich sein, diese Aufgaben in einer unbekannten Umgebung unter der Steuerung durch eine Anwendung verlässlich durchzuführen.

Ich behandle RedMon hier nicht, genau wie ich hier die windowsspezifische SETUP.EXE von GhostScript auslasse. Windowsspezifische Prozeduren sind per Definition nicht erforderlich, um GhostScript erfolgreich einzusetzen. Die Autoren des Quellcodes von GhostScript haben Windows nicht im Hinterkopf, während sie ihren Code schreiben.

Konfiguration zur Laufzeit

Wenn wir davon ausgehen, Sie verwenden Visual FoxPro statt DOS oder UNIX, stehen Sie jetzt vor Schritt 3, in dem Sie FoxPro anweisen, Ihren angegebenen Treiber zu verwenden. Sie haben PDFListener bereits den Namen Ihrer Druckerauswahl mitgeteilt, so dass es einfach ist, FoxPro die Information mit einem Standardbefehl mitzuteilen:

```
SET PRINTER TO NAME;
   (THIS.PSDriverSetupName)
```

Da PDFListener mit VFP 9.0 arbeitet, kann es die Druckereinrichtung besser speichern und wiederherstellen als es in früheren Versionen möglich war. Vor SET PRINTER TO NAME verwendet es in der PROTECTED Methode PrepareFRXPrintInfo SYS(1037,2), um die aktuelle Druckerumgebung von VFP in einem Cursor der speziellen FRX-Datensitzung zu speichern. Nachdem die Druckverarbeitung abgeschlossen ist, werden eine andere PROTECTED Methode UnloadPrinterInfo sowie SYS(1073,3) eingesetzt, um dem Anwender seiner vollständige Druckerumgebung zurückzugeben. Durch den Einsatz dieser Technik werden alle Optionen, die nicht auf dem Standardwert stehen, beispielsweise die Seitengröße und die Seitenausrichtung wiederhergestellt, die der Anwender in seiner VFP-Umgebung eingestellt hat.

PDFListener erledigt diese Aufgaben im dritten Methodenaufruf in der Methode LoadPrinterInfo: AdjustVFPPrinterSetups. Ich habe diesen Namen gewählt, um klarzustellen, dass hier nicht die Windows-Umgebung geändert wird, sondern lediglich das Druckverhalten von VFP. Anders als die früheren Aufrufe der Setup-Methode in LoadPrinterInfo ist diese Methode als PROTECTED gekennzeichnet, da Sie sie ausschließlich als Teil einer definierten Ausgabesequenz aufrufen. Ihre Mechanismen sollten immer zusammen mit UnloadPrinterInfo() verwendet werden, um die Druckerumgebung am Ende der Berichtsverarbeitung zurückzusetzen. Wenn Sie sich entscheiden, Code zum Entfernen und Wiederherstellen für FRX- und LBX-Druckerumgebungen hinzuzufügen, so wie ich es im nächsten Abschnitt empfehle, können Sie ihn in diesen Methoden platzieren.

Verwenden Ihrer Druckanweisungen in Berichten und Etiketten

Wenn Sie Berichte und Etiketten für die Ausgabe verwenden, setzen Sie diese Technik nur mit FRX- und LBX-Dateien ein, die

keine Druckerumgebung gespeichert haben. Andernfalls beeinflussen die Anweisungen SET PRINTER TO NAME das Berichtssystem nicht (in VFP 9.0 speichern neue Berichte und Etiketten die Druckerumgebungsinformation nicht standardmäßig). Wie in Ihrem Quellcode ersichtlich, achtet PDFListener für Sie nicht auf diese Details. Achten Sie darauf, dass Sie das dazu gehörende TEST.PRG nur mit Berichten und Etiketten einsetzen, die nicht über eine eigene Druckerauswahl verfügen. Sie können PDFListener ändern, so dass diese Daten bis zu einer späteren Restauration entfernt werden. Verwenden Sie dafür einen privaten Cursor, sowie eine Technik, die der für die VFP-Druckerumgebung eingesetzten entspricht.

Wenn Sie eine Druckerumgebung in einem Bericht speichern, sie später aber über VFPs native Menüoption löschen, achten Sie darauf, dass diese Option im Feld Expr des Headers den Schalter COLOR hinterlässt. Für PDF-Ausgaben sollten Sie festlegen, ob Sie den Bericht farbig oder in Graustufen ausgeben wollen. Wenn Sie sich für farbig entscheiden, achten Sie darauf, dass das Feld Expr den Wert COLOR=2 enthält. Wenn Sie mit VFP 9.0 arbeiten, müssen Sie das Feld Expr nicht ändern, sondern fügen Sie die Einstellung COLOR=2 im Feld Picture des Header-Datensatzes ein. Nähere Informationen finden Sie im Eintrag „Understanding and Extending Report Structure“ in der Hilfedatei und studieren die Spezifikationstabelle FILESPEC\60FRX.DBF.

Ergebnisse erhalten

Jetzt sind Sie bei Schritt 4 angekommen, dem Generieren der Dateien und dem Aufruf von GhostScript, um die Dateien zu konvertieren. Dies ist der einfachste Teil. PDFListener ist von _ReportListener in den FFC abgeleitet, und führt daher eine Collection mit Berichten mit. Sie erweitert die Methode _ReportListener.AddReport, so dass sie Namen für die temporären Postscript-Ausgabedateien für jeden Bericht in Ihrer Collection generiert und fügt ein intelligentes Parsen Ihrer Befehle REPORT FORM hinzu, um auf diese Weise sicherzustellen, dass der Befehl REPORT FORM die Informationen enthält, die für die Generierung der Dateiausgabe erforderlich sind. Sie erweitert auch die Methode _ReportListener. RunReports mit zwei Aktionen nach der Verarbeitung: wie oben beschrieben das Wiederherstellen der Druckerumgebung von VFP über UnloadPrinterInfo(), sowie das Ausführen seiner Methode ProcessPDF.

ProcessPDF erstellt zunächst eine GhostScript-Befehlsdatei, die alle temporären PostScript-Dateien in Ihrem Berichtslauf enthält

```
#DEFINE GS_COMMAND_STRING_1  " -q -dNOPAUSE " + ;
              " -I./lib;./fonts " + ;
              "-sFONTPATH=./fonts " + ;
              "-sFONTMAP=./lib/FONTMAP.GS " + ;
              "-sDEVICE=pdfwrite -sOUTPUTFILE="
#DEFINE GS_COMMAND_STRING_2 " -dBATCH  "

PROTECTED PROCEDURE MakeGSCommandFile()
LOCAL lcFile, lcContents, lcFileStr, liH
lcFile = FORCEPATH("C"+SYS(2015)+".TXT", ;
   THIS.GSLocation)
lcFileStr = THIS.GetQuotedFileString()
* GetQuotedFileString uses PDFListener's
* collection of temporary file names
* associated with each report in the run
lcContents = GS_COMMAND_STRING_1 + ;
             ["]+THIS.TargetFileName+["]+ ;
                GS_COMMAND_STRING_2 + ;
                + lcFileStr
   * use forward slashes or doublebackslashes:
   lcContents = STRTRAN(lcContents,"\","/")
   IF FILE(lcFile)
      ERASE (lcFile)
   ENDIF
```

```
    liH = FCREATE(lcFile)
    FPUTS(liH,lcContents)
    FFLUSH(liH,.T.)
    FCLOSE(liH)
RETURN lcFile
```

Dies ist noch so ein schreckliches Stück Code, da die Zeile, die GhostScript aufruft, viele Optionen enthält und aufgrund der Pfadangaben zu den Dateien mühsam zu schreiben ist. Die meisten der Optionen, die ich angegeben habe, verweisen GhostScript auf seine Bibliotheken. Wenn Sie die GhostScript-Bibliotheken an einer anderen Stelle ausliefern, als an der, die ich in INSTALL.DBF angegeben habe, ändern Sie diesen Teil des Befehls.

Wenn Sie im vollständigen Quellcode des PDFListener den Code lesen, der diesen Befehl ausführt, werden Sie bemerken, dass ich das aktuelle Verzeichnis temporär in das Verzeichnis geändert habe, in dem Sie die ausführbare Datei von GhostScript platziert haben. Mein Ziel war, die Probleme mit der Groß- und Kleinschreibung oder dem Finden des Verzeichnisses zu minimieren. Dies sind die größten Fallstricke bei der Arbeit mit Code, der ursprünglich nicht für Windows geschrieben wurde. FoxPro ist sehr bequem bei der genauen Groß- und Kleinschreibung der Dateinamen und tendiert daher dazu, diese Gefahr zu vergrößern. Unmittelbar nach der Ausführung von GhostScript stelle ich das Standardverzeichnis wieder her.

Bei der Überprüfung der Methode ProcessPDF werden Sie feststellen, dass, nachdem MakeGSCommandFile() den vollständigen Befehl zurückgegeben hat, eine andere Methode verwendet wird, um die Befehlszeile (THIS.oWinAPI.ProgExecuteX) auszuführen, als der PDFListener vorher verwendet hat, um den Druckertreiber zu installieren. Ich habe festgestellt, dass die Befehlsübergabe an Windows für unterschiedliche Programmtypen am Besten mit einer anderen Fehlerbehandlung und einer anderen Timeout-Strategie durchgeführt wird. Sie haben eventuell Ihre eigene Lösung. Vielleicht ziehen Sie es auch vor, die gesamte Problematik auszuklammern, indem Sie Aufrufe von DECLARE DLL durchführen, um die Funktionalität aufzurufen, die durch die DLL-Bibliotheken von GhostScript bereitgestellt werden. GhostScript verfügt über eine gut dokumentierte API, so dass es nicht zwingend erforderlich ist, die Datei GSWIN32C.EXE auszuführen. Ich bevorzuge allerdings die Befehlszeilenoberfläche, da die Schalter dort gut entwickelt sind und die Tatsache betonen, dass Sie diese Aufgaben mit jeder Anwendung ausführen können, die eine Stapeldatei ausführen oder einen Skript der Shell aufrufen kann.

Unabhängig davon, wie Sie die Aufgabe erledigen, bringt Sie der Aufruf von GhostScript zu Schritt 7. Anschließend verfügen Sie über eine PDF, die alle Berichte enthält, die PDFListener ausgeführt hat.

Die Features von VFP 9.0 prüfen

Jetzt wundern Sie sich vielleicht, weshalb ich mit VFP 8.0 oder 9.0 nicht die Befehle NOPAGEEJECT und REPORT FORM verwende, um alle Berichte in eine einzige PostScript-Datei einzubinden. Dadurch würde PDFListener vereinfacht, da ich in diesem Fall keine Collection mit mehreren temporären Dateien mitführen müsste. Ich könnte auch GhostScript ohne eine Textdatei mit Befehlen ausführen, wenn ich mich nicht um die Länge der Befehlszeile mit mehreren Quelldateien kümmern müsste.

Die Verwendung der Berichtscollection von _ReportListener oder eines ähnlichen Mechanismus verschafft Ihnen verschiedene Vorteile gegenüber NOPAGEEJECT. Der wichtigste Vorteil ist die Möglichkeit, mehrere Berichte mit unterschiedlichen Seitenausrichtungen und Seitengrößen aneinander zu hängen. Im Gegensatz dazu öffnet NOPAGEEJECT für mehrere Berichte eine einzige Verbindung zu einem Drucker, von denen der erste Bericht das Seitenlayout definiert.

Der Einsatz von GhostScript, um die Dateien zu verbinden, ermöglicht es Ihnen auch, PostScript-Quelldateien zu mischen, die unterschiedlichen Ausgabetechniken erzeugt

wurden. Es muss sich nicht um Ergebnisse des Befehls REPORT FORM handeln. Es können auch PostScript-Dateien sein, die durch andere Anwendungen oder verschiedene Instanzen von FoxPro erstellt wurden. Nachdem Sie über den Code verfügen, der im PDFListener für das Erstellen der Befehlsdatei verwendet wird, können Sie ihn an diese Anforderungen anpassen.

Das Verbinden von Ausgaben verschiedener Anwendungen oder unterschiedlicher Instanzen kann besonders auf einem Webserver hilfreich sein. Wie Sie eventuell wissen, wurde VFP 9.0 angepasst, um das Erstellen von Berichten aus einer DLL zu erleichtern (multithreaded DLLs sind dazu immer noch nicht in der Lage). PDFListener hilft Ihnen hier mit einem OLEErrorhandling und _VFP.StartMode.

Wenn Sie diesen Code im Kontext einer Webanwendung ausführen, wollen Sie wahrscheinlich die Setupprozeduren manuell ausführen und sicherstellen, dass der entsprechende Anwender die Rechte an den Features hat, auch für die Ausgabeverzeichnisse und die Druckerauswahl. Im Kontext eines Servers brauchen Sie sich keine Gedanken zur Anwendungstransparenz zu machen.

Sie werden bemerken, dass PDFListener die Eigenschaft generateNewStyleOutput bereitstellt, die anzeigt, ob Sie wünschen, dass Ihren ausgeführten Berichten eine Objektreferenz auf den ReportListener zugeordnet wird. Obwohl _ReportListener den Befehl REPORT FORM ausführt, können die Befehle selbst sowohl im alten wie auch im neuen Ausgabestil sein. PDFListener stellt diese Eigenschaft als Vorgabewert auf .F.; die daraus entstehenden PostScript-Dateien werden deutlich kleiner und die Qualität ist trotzdem gut. Wenn Sie die alte Berichtsengine einsetzen, enthalten die Dateien Text, der mit PostScript-Anweisungen für Ihre Daten eingefasst ist. Beim Einsatz der neuen Engine bildet jede volle Seite eine Grafik.

Stellen Sie generateNewStyleOutput auf .T., wenn Sie einen ReportListener einsetzen, um dynamische Effekte zu verwenden, beispielsweise wenn Sie eine Geschäftsgrafik anzeigen, die Sie mit GDI+ direkt für den Bericht vorbereiten. Die Methode AddReport des _ReportListener gibt Ihnen die Möglichkeit, für jeden Bericht in der Collection die Objektreferenz auf den ReportListener separat anzugeben, wenn Sie dies wünschen.

Die Features von VFP 9.0 für spezielle Vorteile einsetzen

In VFP 9.0-Umgebungen könnten Sie darüber nachdenken, die Ereignisse LoadReport und UnloadReport für PDF-Szenarien zu verwenden.

Beispielsweise könnten Sie die Druckerumgebung für einen bestimmten Bericht prüfen, und anschließend in eine „saubere“ temporäre Version ohne Druckerumgebung wechseln, die ausschließlich für den PDF-Druck benötigt wird und wie weiter oben beschrieben den erforderlichen Schalter COLOR hinzufügen. Sie finden die Beispiel-ReportListener-Klasse RLSwap im Eintrag LoadReport Event der Hilfedatei von VFP.

Ich ziehe es vor, diese Technik nicht einzusetzen, da sie mich ohne echten Grund auf den neue Ausgabestil einschränken würde. Ich verwende bereits die Methode _ReportListener.Run, um mehrere Berichte in einen PDF-Generierungsprozess aufzunehmen. Jeder Bericht in der Sequenz könnte über seine eigene von ReportListener abgeleitete Objektreferenz verfügen, die jeweils auf PDFListener oder eine vollständig andere von ReportListener abgeleitete Klasse verweisen. Alle Berichte könnten auch im alten Ausgabestil ausgeführt werden. Die Ereignisse LoadReport und UnloadReport des PDFListeners könnten einmal, mehrfach oder nie auftreten. Es wäre praktischer, sich durch die Collection mit den Berichten zu arbeiten und ihre Druckerumgebungen anzupassen, oder am Beginn der Methode .Run auf temporäre Berichtskopien umzuschalten. Die Methode würde dann fortfahren, die Befehle REPORT FORM in ihrer angegebenen Reihenfolge auszuführen.

Wenn Sie entscheiden, Ihren Prozess der PDF-Generierung auf einen einzelnen Befehl REPORT

FORM einzuschränken und wenn Sie von Ihrem ReportListener bereits die neue Ausgabe benötigen, ist eine FRX-swapping-Technik mit LoadReport und UnloadReport definitiv durchführbar. Sie können die Einrichtung auch anders durchführen, beispielsweise indem Sie in diesen Ereignissen den GhostScript Druckertreiber einstellen und wiederherstellen. In diesem Szenario könnten Sie dieser von ReportListener abgeleiteten Klasse ihren eigenen Wert OutputType zuordnen und sie registrieren, um anschließend mit der Anwendung _REPORTOUTPUT auf einfache Weise auf sie zugreifen zu können.

Vielleicht gibt es noch viele alternative Strategien. Aber seien Sie gewarnt: auch in der schönen neuen Welt des VFP 9.0-Berichtswesens sind Ihre gültigen Auswahlmöglichkeiten nicht vollkommen grenzenlos. Beispielsweise könnte es natürlich sinnvoll sein, die Berichts-Collection mit dem Einsatz der Ereignisse LoadReport und UnloadReport zu kombinieren, beispielsweise in einer Sequenz wie dieser:

- Speichere alle Berichtsinformation, mit Ausnahme der für den ersten Bericht, in der Berichts-Collection des PDFListeners.
- Führe einen Berichts-Befehl für den ersten Bericht mit einer Objektreferenz auf den PDFListener aus.
- Führe im Ereignis LoadReport des PDFListener das Setup aus.
- Führe den ersten Bericht aus.
- Im Ereignis UnloadReport arbeite dich durch die Collection und führe die restlichen Berichte aus. Anschließend führe das Cleanup aus.

Diese Sequenz würde nicht funktionieren. Das hat einen einfachen Grund: Der Befehl REPORT FORM kann nicht eingebettet werden, auch nicht im Ereignis UnloadReport. Wie Sie im folgenden Test erkennen können, ist dies keine Frage der Ereignisrekursion, in dem die Befehle REPORT FORM für die zusätzlichen Berichte im alten Stil ausgeführt werden:

```
ox = CREATEOBJECT("x")
REPORT FORM ? OBJECT ox
DEFINE CLASS x AS ReportListener
   PROCEDURE UnloadReport
      REPORT FORM ?
   ENDPROC
ENDDEFINE
```

Ausliefern und Verteilen der Installationsdateien

Ihr Quellcode enthält die kurze Routine CREATEINSTALLDBF.PRG, die die Installationstabelle für jede Version von GhostScript erstellt, die ich für die Auslieferung vorbereite. Sie fragt nach den Quellverzeichnissen und erstellt Datensätze, in denen alle Dateien gespeichert werden, die ich ausliefern will – auch die Textdatei mit der Lizenz der Free Software Foundation.

Beachten Sie, dass es sich bei dem Satz mit den Dateien nicht um den vollständigen originalen GhostScript-Satz handelt, sondern um den Satz, den GhostScript zur Laufzeit benötigt. Dazu gehören die ausführbare Datei und die DLL-Bibliotheken, GhostScript-spezifische Konfigurations- und Befehlsdateien, sowie die Dateien mit den Schriften.

Die Verzeichnisstruktur, die PDFListener aus der INSTALL.DBF erstellt, ist nicht die originale Verzeichnisstruktur, die durch die GhostScript-Distribution verwendet wird. Ich verwende ein Arrangement (siehe **Abbildung 3**), das meiner Planung entspricht, wo GhostScript seine Dateien zur Laufzeit finden soll. All dies scheint zusätzliche Prob-

leme zu verursachen, garantiert aber, dass Ihr Programm diese Version von GhostScript mit den richtigen Komponenten verwendet und keine andere Version, die auf der Maschine registriert sein könnte.

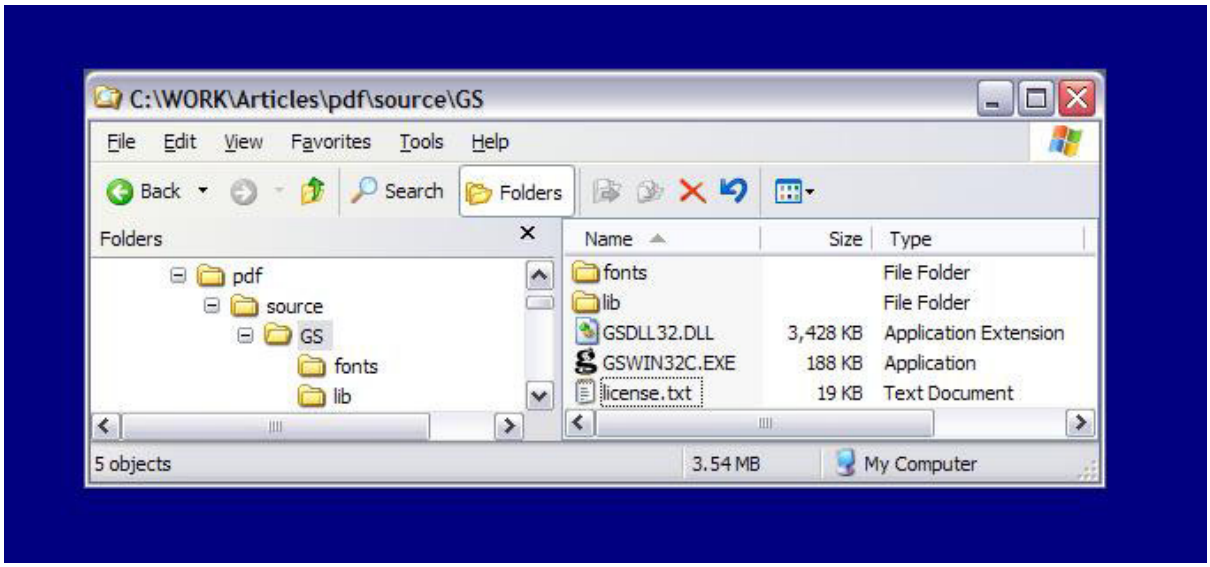


Abbildung 3. Die vereinfachte Verzeichnis- und Dateistruktur für GhostScript zur Laufzeit.

Auch wenn es sich nicht um die vollständige GhostScript-Distribution handelt, ist INSTALL.DBF eine große Datei (etwa 8 Megabyte). Ein Großteil dieser Menge sind Fontdefinitionen. Sie können einige dieser Definitionen weglassen. Wenn Sie sich die GhostScript-Site ansehen, werden Sie feststellen, dass unterschiedliche Distributionen mit unterschiedlichen Schriften ausgeliefert werden. Sie können, falls erforderlich, dem Satz weitere PostScript-Fonts hinzufügen oder andere löschen, aber die Grundmenge, die Sie hier finden und die als Standard-Fontmappings in GNU GhostScript vorhanden sind, bilden eine gute Entsprechung für den Großteil der PostScript-Ausgaben.

Sie könnten die Eigenschaft GSLocation verwenden, um auf den Speicherort zu verweisen, an dem sich sowohl INSTALL.DBF als auch die GhostScript-Dateien befinden. In diesem Szenarion würde PDFListener die Bibliotheken immer in einer speziellen Relation zu INSTALL.DBF installieren.

Vielleicht verwenden Sie lieber einen vollständig anderen Mechanismus. Beispielsweise könnten Sie die Inhalte von INSTALL.DBF als Unterstützungsverzeichnisse in einzelnen Dateien in Ihre SETUP.EXE mit aufnehmen. Sie könnten auch eine separate VFPSTARTUP.EXE erstellen, die diese Dateien auf der Festplatte neu erstellt oder Ihren Druckertreiber überprüfen und die aus der Prüfung resultierenden Namen und Verzeichnisse in Konfigurationsdateien speichern, die Ihre Anwendung verwendet, wenn die Berichte ausgeführt werden. Ich erstelle für diesen Zweck gerne eine CONFIG.XML. Diese Datei kann meiner EXE als einziges Argument übergeben werden. Intern weiß meine EXE, dass sie, wenn Sie mit einem Befehlszeilenparameter aufgerufen wird, diese als XML-Datei laden kann, die Konfigurationswerte aus dieser Datei laden und ihre Setup-Arbeit erledigen kann.

Das alles klingt nach mehr Arbeit als es ist

Ihr Quellcode enthält die Datei TEST.PRG. Dieses Programm enthält einige hilfreiche Kommentare mit Ratschlägen für den Einsatz von PDFListener und seiner verschiedenen nicht essentiellen Features. Sie können mit dem Programm den Prozess von Anfang bis Ende testen.

Die Testklasse fordert Sie auf, die Namen von FRX-Dateien anzugeben und fügt diese der Berichts-Collection des PDFListeners in

einer Schleife hinzu (vergessen Sie nicht, FRXe ohne gespeicherte Druckerumgebung auszuwählen). Anschließend wird die Methode RunReports des PDFListeners ausgeführt, in der der PDFListener prüft, ob die erforderlichen externen Komponenten vorhanden sind, diese falls erforderlich installiert und die Berichte ausführt, um die PostScript-Dateien zu generieren. Am Ende wird die PostScript-Ausgabe in eine PDF umgewandelt. Dann verwendet TEST.PRG einen Aufruf von ShellExecute, um das PDF-Dokument anzuzeigen.

Prüfen Sie vor und nach dem Prozess Ihre Druckereinrichtung und Sie sollten feststellen, dass PDFListener eine Einrichtung hinzugefügt hat. Ihr Standarddrucker und Ihre eigene VFP-Druckeinstellung sollten weiterhin vorhanden sein. Prüfen Sie das Verzeichnis unterhalb Ihrer Kopie von TEST.PRG und PDFLISTENER.PRG und Sie sollten die GhostScript-Dateien finden. Prüfen Sie Ihre Uhr und, vorausgesetzt, Sie haben in der Schleife nicht 100 FRXe mit 1000 Seiten gewählt, Sie sollten feststellen, dass keine große Zeitspanne vergangen ist. Prüfen Sie die Ausgabe und Sie finden, was Sie gesucht haben.

Jetzt geben Sie Ihren Anwendern, was sie wollen.

Lisa Slater Nicholls schreibt seit 1990 extensiv über die Datenbankentwicklung. Sie und Colin Nicholls arbeiteten im Projektmanagement, beim Entwurf der Architektur und der Xbase-Entwicklung der neuen Features des Visual FoxPro 9.0-Berichtssystems. Sie leben in Las Vegas, Nevada und beschäftigen sich mit der Integration von Unternehmensdaten in vielen unterschiedlichen Umgebungen. Sie erreichen Lisa unter lisa@spacefold.com.