Link Shop

Burkhard Quenzler
(Jg. 1957) Fernstudium, 1990-2014 Administrator IT, seit 2014 Datenbankprogrammierer. Ab 1993 nebenbei für DB wie dBase, MS SQL und Access, seit FileMaker Version 7 auch als FileMaker Entwickler tätig, hat viele kleinere und größere Projekte mit FileMaker realisiert.
info@fmqu.de

Teil 6

Erstellen einer einfachen Anwendung mit FileMaker

Ein Volltextsucharchiv für das FileMaker Magazin und Zeitschriften im PDF-Format

Herzlichen Dank an all die Leser, die mir und meinen Beiträgen seit mittlerweile einem Jahr die Treue gehalten haben! In dieser Zeit haben wir eine Anwendung programmiert, in der wir unsere PDFs mit den dazugehörigen ZIP-Archiven speichern und nach Wörtern, Begriffen und Autoren durchsuchen können. Alles, was uns jetzt noch fehlt, ist ein bisschen mehr Komfort.

Um zur perfekten Funktionalität noch ein bisschen Komfort hinzuzufügen, müssen am Programm nur ein paar Kleinigkeiten verändert werden. Gerne hätte ich z. B. einen Ordner, in den ich meine zu archivierenden PDF-Dateien einfach ablege. Beim nächsten Start werden diese dann vom Programm automatisch eingelesen und können anschließend von mir katalogisiert werden. Dazu erstellen wir verschiedene Scripts, die uns die eintönige Arbeit des Einlesens abnehmen werden.

Es sollten folgende Schritte erledigt werden:

1. Beim Start des Programms soll das Script nachschauen, ob im Ordner ***EinlesenPDF*** Dateien vorhanden sind.
2. Sind Dateien vorhanden, prüft das Script, ob es sich um PDFs handelt, und erstellt anschließend eine Liste mit den im Ordner befindlichen Dateien.
3. Alle Dateien werden nun nacheinander eingelesen, gefiltert nach PDF-Dateien. Im letzten Schritt wird der Inhalt der Dokumente ausgelesen.

Ganz sicher gibt es noch sehr viel mehr Möglichkeiten, aber diese gehören jedoch nicht zur Aufgabenstellung des Projekts und würden den Rahmen dieser Veröffentlichung sprengen.

Wichtig ist, dass wir das verwendete Plugin etwas mehr ausreizen, als das bisher der Fall war. Es wird uns sowohl beim Auslesen der PDF-Dateien als auch bei den Dateioperationen helfen.

Zunächst gilt es aber, das **MBS**-Plugin zu registrieren, damit wir seine vielen Möglichkeiten auch einsetzen können. Die Registrierung ist mit einer Zeile erledigt, die Variable ist frei wählbar:

```
◆ Variable setzen
  $Registrierung ; Wert: MBS ("Register"; "Vorname Name";
  "Complete"; "Developer"; 9999999; 9999999999999)
```

Nun müssen wir in das bereits existierende Script „System“ den folgenden Teil einfügen, um die notwendigen Pfade zu erhalten. Das muss nicht unbedingt im Script „System“ geschehen, Sie müssen nur dafür sorgen, dass dieser Teil beim Start des Programmes ausgeführt wird.

```
◆ # --------- Ergänzung Folge 6 ---------
◆ Variable setzen
  $$databasePath ; Wert: Hole ( DateiPfad )
◆ Variable setzen
  $$databasePath ; Wert: Austauschen ( $$databasePath ;
  ".fp7" ; ".fmp12")
◆ Variable setzen
  $$databaseNatPath ; Wert: MBS ( "Path.FilemakerPathToNative
  Path"; $$databasePath )
◆ Variable setzen
  $$databaseName ; Wert: Hole ( DateiName ) & ".fmp12"
```

```
◆ Variable setzen
  $$databaseFolder ; Wert: Austauschen ( $$databasePath ;
  $$databaseName ; "" )
◆ Variable setzen
  $$databaseNatFolder ; Wert: Austauschen ( $$databaseNatPath ;
  $$databaseName ; "" )
◆ Variable setzen
  $$databaseNatPDFPath ; Wert: $$databaseNatFolder &
  "EinlesenPDF/"
◆ Variable setzen
  $$databasePDFPath ; Wert: $$databaseFolder & "EinlesenPDF/"
◆ Variable setzen
  $$databaseNatRootDataPath ; Wert: $$databaseNatFolder &
  "Dateiablage/"
◆ Variable setzen
  $$databaseRootDataPath ; Wert: $$databaseFolder &
  "Dateiablage/"
◆ # --------- Ergänzung Folge 6 ---------
```

Wundern Sie sich nicht über die mehrmalige Zuweisung der Pfade – einmal wird der Pfad nativ benötigt, dann wieder nicht. Im Grunde werden hier nur Variablen gesetzt, in denen die Namen der Ordner gespeichert werden. Dazu wird der aktuelle Dateipfad geholt und die gewünschten Ordnernamen werden angehängt.

Für das Zwischenspeichern der Ordnerinhalte benötigen wir eine neue Tabelle namens ***PDF_FileListing***.

Feldname	Typ	Option
HauptgruppeLID	Text	Autom. fortl. Nr.
Result	Text	Global
Name	Text	
Dateiname	Formel	= ZeichenRechts (Name ; Länge (Name) – Länge ($$databaseNatRootDataPath))
Ausgabe	Formel	= FilterZeichen (Dateiname ; "0123456789")
NDateiname	Formel	= Wenn (Endung = "PDF" ; ZeichenLinks (Dateiname ; Länge (Dateiname) –4) & "." & Endung ; "")
Endung	Formel (Indiziert)	= Großbuchstaben (ZeichenRechts (Dateiname ; 3))
NName	Formel	= Wenn (NDateiname ≠ "" ; Wenn (Großbuchstaben (Endung) = "PDF" ; $$databaseNatPDFPath & NDateiname ; $$databaseNatArchivPath & NDateiname) ; "")
Ergebnis	Text	

In dieser Tabelle werden alle Dateien aufgelistet, die sich im Ordner ***DateiAblage*** befinden. Achten Sie bitte darauf, dass Sie die oben angezeigten Feldoptionen genau übernehmen, damit die Scripts anschließend fehlerfrei funktionieren können. Ist die Tabelle bereit, wenden wir uns den Script-Dateien zu.

Zunächst prüft das Script, ob die Ordner vorhanden sind. Ist das der Fall, können wir weitermachen, sonst müssen die Ordner erstellt werden.

`EinleseVorgang` *(Teil 1)*

```
◆ # ____________________________________________________
◆ # © 2016 by Burkhard Quenzler, BeQuSoft
◆ # Script-Parameter: keine
◆ # Einlesevorgang
◆ # Version: 1.06
◆ # ____________________________________________________
◆ # Prüfung, ob Datenpfad vorhanden ist
◆
◆ Wenn
  MBS ( "Files.DirectoryExists" ; $$databaseNatRootDataPath )
  = 0
◆ Eigenes Dialogfeld anzeigen
  "Fehlermeldung" ; "Der erwartete Pfad " &  $$databaseNatRoot
  DataPath & " existiert nicht. Er wird jetzt angelegt,
  danach wird das Programm geschlossen. Bitte legen Sie im
  Ordner 'DateiAblage' Ihre PDF's und Archivdateien ab."
◆ Variable setzen
  $r ; Wert: MBS("files.createDirectory"; $$databaseNatRoot
  DataPath)
◆ Ende (wenn)
◆ Wenn
  MBS ( "Files.DirectoryExists";$$databaseNatPDFPath ) = 1
◆ # Für Erweiterungen vorgesehen
◆ Sonst
◆ Eigenes Dialogfeld anzeigen
  "Fehlermeldung" ; "Der erwartete Pfad " & $$databaseNat
  PDFPath & " existiert nicht. Er wird jetzt angelegt,
  danach wird das Programm geschlossen. Bitte legen Sie im
  Ordner 'DateiAblage' Ihre PDF's und Archivdateien ab."
◆ Variable setzen
  $r ; Wert: MBS("files.createDirectory"; $$databaseNat
  PDFPath)
◆ Ende (wenn)
◆ Wenn
  MBS ( "Files.DirectoryExists";$$databaseNatArchivPath ) = 1
◆ # Für Erweiterungen vorgesehen
◆ Sonst
◆ Eigenes Dialogfeld anzeigen
  "Fehlermeldung" ; "Der erwartete Pfad " &  $$databaseNat
  ArchivPath & " existiert nicht. Er wird jetzt angelegt,
  danach wird das Programm geschlossen. Bitte legen Sie im
  Ordner 'DateiAblage' Ihre PDF's und Archivdateien ab."
◆ Variable setzen
  $r ; Wert: MBS("files.createDirectory"; $$databaseNat
  ArchivPath)
◆ Ende (wenn)
```

Der zweite Teil des Scripts wird nur dann ausgeführt, wenn der Ordner ***DateiAblage*** nicht leer ist. Das Script zeigt uns an, dass Dateien vorhanden sind und wir können wählen, ob eingelesen werden soll oder nicht.

`EinleseVorgang` *(Teil 2)*

```
◆ # --------- Teil 2 -----------------------------
◆ Wenn
  HoleWert ( MBS ( "Files.FolderSize" ;
  $$databaseNatRootDataPath ) ; 2 ) > 1
◆ Eigenes Dialogfeld anzeigen
  "Dateneinlesevorgang" ; "Es wurden Dateien im
  Dateiablegeordner gefunden. Jetzt werden alle Dateien,
  die sich im Ordner 'Dateiablage' befinden, in das
  Programm eingelesen."
◆ Wenn
  Hole ( LetzteMeldungswahl ) ≠ 1
◆ Alle Datensätze anzeigen
◆ Aktuelles Script verlassen
◆ Textergebnis:
```

```
◆ Ende (wenn)
◆ Wenn
  MBS ( "Files.DirectoryExists" ; $$databaseNatRootDataPath ) = 1
◆ Script ausführen
  "Dateien einlesen"
◆ Wenn
  $$NoData = 1
◆ Eigenes Dialogfeld anzeigen
  "Fehlermeldung" ; "Es wurde keine Datei in dem Ordner " & $$databaseNatRootDataPath & " gefunden. Das Einleseprogramm wird jetzt beendet."
◆ Variable setzen
  $$NoData ; Wert: 0
◆ Alle Scripts abbrechen
◆ Sonst
◆ Script ausführen
  "Dateien verschieben"
◆ Script ausführen
  "PDF-Dateien einlesen"
◆ Script ausführen
  "Leeren der Einleseordner"
◆ Gehe zu Layout
  "PDF" (PDF)
◆ Ende (wenn)
◆ Sonst
◆ Eigenes Dialogfeld anzeigen
  "Fehlermeldung" ; "Der erwartete Pfad " &  $$databaseNatRootDataPath & " existiert nicht.  Er wird jetzt angelegt, danach wird das Programm geschlossen. Bitte legen Sie Ihre PDFs und Archivdateien im Ordner 'DateiAblage' ab."
◆ Variable setzen
  $r ; Wert: MBS ( "files.createDirectory"; $$databaseNatRootDataPath )
◆ Alle Scripts abbrechen
◆ Ende (wenn)
◆ Ende (wenn)
```

Nach der Bestätigung des Anwenders beginnt der eigentliche Einlesevorgang, der sich in drei Schritte aufteilt:

1. Verschieben der Dateien (nur PDFs) in den Ordner ***EinlesenPDF***
2. Einlesen der PDF-Dateien in das Programm
3. Leeren der Einleseordner mitsamt der ***DateiAblage***

Für jeden dieser Schritte gibt es ein eigenes Script.

`Dateien einlesen`

```
◆ # _________
◆ # © 2017 by Burkhard Quenzler, BeQuSoft
◆ # Script-Parameter: keine
◆ # Einlesevorgang
◆ # Version: 1.0
◆ # _________
◆ Variable setzen
  $r ; Wert: MBS ( "Trace" )
◆ Gehe zu Layout
  “PDF_FileListing” (PDF_FileListing)
◆ Alle Datensätze löschen
◆ Mit Dialog: Aus
◆ Variable setzen
  $r ; Wert: MBS ( "Files.ListRecursive"; $$databaseNatRootDataPath; 39 )
◆ Feldwert setzen
  PDF_FileListing::Result ; $r
◆ Variable setzen
  $count ; Wert: ElementeAnzahl ( $r )
◆ Variable setzen
  $index ; Wert: 0
◆ Wenn
  $count > 0
◆ Schleife (Anfang)
◆ Neuer Datensatz/Abfrage
◆ Feldwert setzen
  PDF_FileListing::Name ; HoleWert ( $r ; $index + 1 )
◆ Schreibe Änderung Datens./Abfrage
  Mit Dialog: Aus
◆ Variable setzen
  $index ; Wert: $index + 1
◆ Verlasse Schleife wenn
  $index = $count
◆ Schleife (Ende)
◆ Sonst
◆ Variable setzen
  $$NoData ; Wert: 1
◆ Ende (wenn)
◆ Gehe zu Layout
  "PDF" (PDF)
```

Was hier passiert, ist schnell erklärt – und dank des Plugins von Christian Schmitz ein Kinderspiel. Diese „eierlegende Wollmilchsau“ unter den Plugins sollte sich eigentlich jeder FileMaker Programmierer gönnen: Das Plugin listet alle Dateien auf, die sich in dem Ablage-Ordner befinden und schreibt jede Datei in eine Tabelle. Im nächsten Schritt verschieben wir die PDF-Dateien – und eben nur die PDF-Dateien – in den Ordner ***Ablage PDF***. Dazu wird der Dateiname in Name und Endung zerlegt. Ist die Endung „PDF“, erhält die Datei einen neuen Namen und der geplante Ablageort wird mit angefügt. Somit können wir die Datei mithilfe des **MBS**-Plugins in den richtigen Ordner verschieben.

`Dateien verschieben`

```
◆ # _____________________________________________
◆ # ©  2017 by Burkhard Quenzler, BeQuSoft
◆ # Script-Parameter: keine
◆ # Dateien verschieben
◆ # Version: 1.0
◆ # _____________________________________________
◆ Gehe zu Layout
  "PDF_FileListing" (PDF_FileListing)
◆ Gehe zu Datens./Abfrage/Seite
  Erste(r)
◆ Variable setzen
  $Count ; Wert: 1
◆ Variable setzen
  $MaxCount ; Wert: PDF_FileListing::Current_total
◆ Suchenmodus aktivieren
  Pause: Aus
```

```
◆ Feldwert setzen
  PDF_FileListing::Endung ; "PDF"
◆ Ergebnismenge suchen []
◆ Variable setzen
  $$PDF ; Wert: PDF_FileListing::Current_Found_Count
◆ Alle Datensätze anzeigen
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetBottomText" ;
  "Insgesamt sind "& $Count&" von "& $MaxCount&"
  Datensätzen eingelesen") ]
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetTitle";
  "Einlesevorgang der PDF-Dateien")
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetTopText";"Achtung!
  Das Einlesen der PDF-Dateien kann durchaus einige Minuten
  andauern!")
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetButtonCaption";
  "Abbrechen")
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetImage";PDF_
  System::Bild)
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetProgress";"-1")
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.Show")
◆ Schleife (Anfang)
◆ Wenn
  PDF_FileListing::NDateiname ≠ ""
◆ Feldwert setzen
  PDF_FileListing::Ergebnis ; MBS ( "Files.MoveFile"; PDF_
  FileListing::Name;PDF_FileListing::NName ; 1 ; 1 )
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetProgress";
  $count*100/$$PDF)
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetBottomText" ;
  "Insgesamt sind "& $Count &" von "& $MaxCount&"
  Datensätzen eingelesen")
◆ Variable setzen
  $Count ; Wert: $Count + 1
◆ Ende (wenn)
◆ Verlasse Schleife wenn
  MBS ( "ProgressDialog.GetCancel" )
◆ Gehe zu Datens./Abfrage/Seite
  Nächste(r) ; Nach letztem beenden
◆ Schleife (Ende)
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.SetBottomText"; " ")
◆ Variable setzen
  $Nirwana ; Wert: MBS ( "ProgressDialog.Hide" )
```

Jetzt folgt nur noch das Einlesen der PDF-Daten und das Leeren der Ordner:

PDF-Dateien einlesen

```
◆ # ____________________________________________
◆ # © 2016 by Burkhard Quenzler, BeQuSoft
◆ # Script-Parameter: keine
◆ # PDF-Dateien einlesen
◆ # Version: 1.06
◆ # ____________________________________________
```

```
◆ Gehe zu Layout
  "PDF" (PDF)
◆ Datensätze importieren
  Mit Dialog: Aus ; $$databasePDFPath; Bild- und Filmdateien
  ; Hinzufügen; Unicode (Big Endian)
◆ Gehe zu Datens./Abfrage/Seite
  Erste(r)
◆ Variable setzen
  $Count ; Wert: 1
◆ Variable setzen
  $MaxCount ; Wert: PDF_FileListing::Current_total
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetBottomText";
  "Insgesamt sind "& $Count&" von "& $$PDF &" Datensätze
  übersetzt")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetTitle";"Umwandlung
  der PDF in TXT-Dateien")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetTopText";"Achtung!
  Das Umwandeln der PDF-Dateien kann durchaus einige
  Minuten andauern!") ]
◆ // Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetButtonCaption";
  "Abbrechen")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.ClearImage")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetProgress";"-1")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.Show")
◆ Schleife (Anfang)
◆ Script ausführen
  "PDF einlesen"
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetProgress";
  $count*100/$$PDF)
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetBottomText";
  "Insgesamt sind "& $Count&" von "& $$PDF &" Datensätze
  übersetzt")
◆ Variable setzen
  $Count ; Wert: $Count + 1
◆ // Verlasse Schleife wenn
  MBS("ProgressDialog.GetCancel")
◆ Gehe zu Datens./Abfrage/Seite
  Nächste(r) ; Nach letztem beenden
◆ Schleife (Ende)
◆ Sortieren
  Wiederherstellen ; Mit Dialog: Aus
◆ Alle Datensätze anzeigen
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.SetBottomText"; " ")
◆ Variable setzen
  $Nirwana ; Wert: MBS("ProgressDialog.Hide")
```

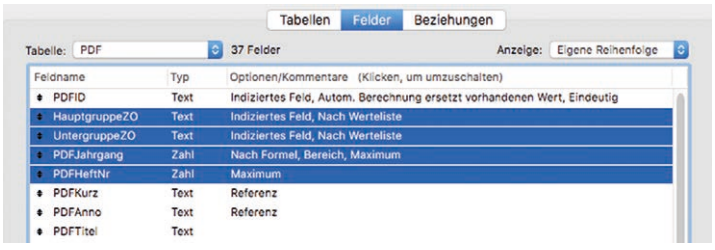
Hier wird es ein bisschen komplizierter, aber die Hauptarbeit leistet der Befehl „Datensätze importieren“. Wir machen uns einfach zunutze, dass PDFs unter „Bild- und Filmdateien“ fallen und somit direkt eingelesen werden können. Das ist zwar etwas von „hinten durch die Brust“, aber wie heißt es so schön: Der Zweck heiligt die Mittel. Damit sind alle PDF-Dateien eingelesen und wir können die Ordner wieder leeren. Dafür gibt es keinen Befehl, der mir spontan einfällt, sodass

ich den Ordner einfach lösche und ihn dann wieder neu erstelle. Das geht recht schnell und ich bin sicher, dass auch alle NICHT-PDF-Dateien entfernt sind.

Leeren der Einleseordner

```
# _____________________________________________
# © 2016 by Burkhard Quenzler, BeQuSoft
# Script-Parameter: keine
# Leeren der Einleseordner
# Version: 1.06
# _____________________________________________
Eigenes Dialogfeld anzeigen
  "Information" ; "Der Einlesevorgang ist damit abgeschlossen.
  Jetzt werden nur noch die Einlesen-Ordner geleert.
  Benötigen Sie diese Dateien noch, können Sie das Script
  vor dem Löschen Abbrechen."
Wenn
Hole (LetzteMeldungswahl) = 1
Variable setzen
  $r ; Wert: MBS ( "Files.DeleteFolder" ;
  $$databaseNATPDFPath )
Variable setzen
  $r ; Wert: MBS ( "Files.DeleteFolder" ;
  $$databaseRootDataPath )
Variable setzen
  $r ; Wert: MBS ( "Files.CreateDirectory" ;
  $$databaseNatPDFPath )
Variable setzen
  $r ; Wert: MBS ( "Files.DeleteFolder" ;
  $$databaseRootDataPath )
Sonst
Alle Scripts abbrechen
Ende (wenn)
```

Damit ist alles erledigt, was wir uns vorgenommen haben. Lediglich die blau markierten Felder dürfen nicht mehr auf „Nicht leer“ stehen. Da das Einlesen automatisch erfolgt, müssen Sie diese Felder nach dem Einlesevorgang von Hand ausfüllen.



Als Letztes muss das Script „System“ erweitert werden, sodass das Script „EinleseVorgang“ bei jedem Start der Datenbank ausgeführt wird. Sollte der Ordner ***Dateiablage*** leer sein, wird das Script einfach ohne weitere Aktion beendet.

```
Script ausführen [ "EinleseVorgang" ]
```

Damit haben wir unser Projekt abgeschlossen. Wie immer steht Ihnen die Beispieldatei auf der Webseite des FileMaker Magazins zur Verfügung. Ich bedanke mich noch einmal bei allen, die mich begleitet haben. Ein ganz besonderer Dank geht an das Team vom FileMaker Magazin, das im Laufe des letzten Jahres alle sechs Beiträge von mir gegengelesen, geprüft, lektoriert und im Satz in Form gebracht hat.

Ausblick

Wenn Sie Anregungen oder Wünsche haben oder einfach ein Feedback geben wollen, schreiben Sie mir gerne eine E-Mail an ***info@fmqu.de***.

Und letztendlich würde es mich freuen, weitere Projekte mit Ihnen gemeinsam anzugehen. Da wäre zum Beispiel eine Versionierung innerhalb der jeweiligen FileMaker Datei, d. h. Sie hätten eine direkte Versionsverwaltung Ihres Programms immer mit an Bord. Oder die Realisierung einer eigenen Seriennummergenerierung und -verwaltung mit FileMaker. Schreiben Sie mir, welches dieser Projekte Sie als Erstes lesen möchten.

Zum Schluss noch ein Wort in eigener Sache: Es gibt das **FileMaker Magazin Archiv** (FMMA) auch als fertiges Programm zu kaufen. Für 25 € können Sie es über meine o. g. Adresse oder über den FMM Webshop beziehen! ■