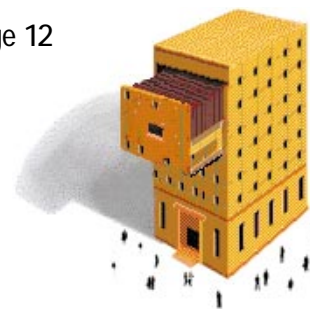


WORKSHOP



FileMaker 3.0

Folge 12



AppleScript II

Während FileMaker in unserem letzten Praxisbeispiel durch ein Applet „fremdbestimmt“ wurde, geht es in dieser Folge um die „versteckte“ Integration von AppleScript. Dabei wird die Menüleiste von FileMaker schwarz (aktiviert) bleiben und in ihr auch kein zusätzliches Applet-Icon erscheinen. Trotzdem wird FileMaker Dinge tun, die wir aus gängigen Anwendungen nicht gewohnt sind.

Die Undercover-Strategie. FileMaker kann AppleScripts, die im internen Skripteditor ScriptMaker oder in Variablenfeldern an- oder abgelegt wurden, selbständig ausführen. Über einige Bugs sollte man dabei ruhig großzügig hinwegsehen, ebenso über die Tatsache, daß FileMaker die Geschwindigkeit, mit der

beispielsweise XPress die Skripts kompiliert und ausführt, bei weitem nicht erreicht. Immerhin liegt FileMaker im Tempo gleichauf mit den externen Applets. Zudem sprechen für die versteckte Integration von AppleScript die Benutzerfreundlichkeit und der Einsatz des (schnelleren) FileMaker-Skripteditors ScriptMaker für die Abarbeitung FileMaker-interner Prozeduren.

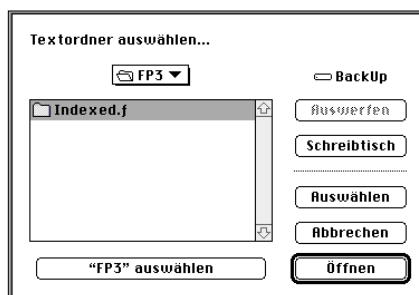
Das folgende Beispiel entstand unter der Prämisse: Nur was FileMaker definitiv nicht kann, soll AppleScript erledigen. FileMaker kann in dieser Konstellation die AppleScript-Ergebnisse nur über die Zwischenablage empfangen. Es ist also wichtig, die Werte in AppleScript so aufzubereiten, daß sie in einem Container übergeben werden, den FileMaker lesen und dekodieren kann.

Das Szenario. Folgendes Szenario sei gegeben: Im Laufe der Zeit haben sich in mehreren Ordnern Hunderte von News, Technical Notes, ReadMe-Dateien und anderen Textdokumenten angesammelt. Vor der Entsorgung in den Papierkorb wollen wir die wichtigsten Dokumente mit einem kleinen Stichwort versehen und in eine FileMaker-Datenbank übernehmen.

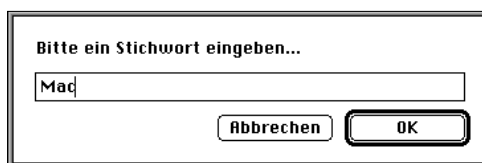
Die Prozedur soll in etwa folgendermaßen ablaufen: FileMaker bittet uns nach der Auswahl eines Dokumentenordners um ein Stichwort, präsentiert in einer Auswahlliste die gefundenen Textdokumente und importiert die selektierten Texte in die Datenbank. Da das Feldlimit von FileMaker bei 64 Kilobyte liegt, dürfen die Textdokumente selbstverständlich nicht größer sein.

FileMaker-Workshop

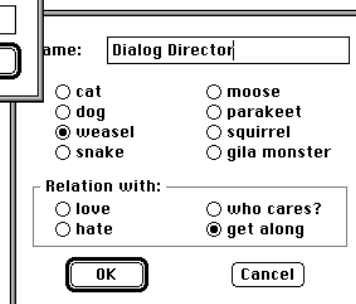
Mit Version 3 hat die im Mac-Bereich weitverbreitete Datenbank FileMaker den Schritt zur Relationalität gemacht. Was das konkret bedeutet und wie Sie die neuen Funktionen sinnvoll einsetzen können, zeigt Ihnen unsere Serie.



1 Den ersten Dialog der Importprozedur liefert Apples Scripting-Erweiterung „Choose file“. Dem Anwender bleibt AppleScript bei allen Aktionen verborgen. Nur der geübte FileMaker-Anwender wird sich über die folgenden Dialogfenster wundern.



2 Der Suchdialog wird von Apples Osax „Display dialog“ generiert, einer der meistgebrauchten Erweiterungen. Wer aufwendigere Rückfragen an den Anwender hat, kann mit der Osax „Dialog Director“ die Dialogfähigkeit erheblich steigern.



Die Werkzeugkiste. Wie bereits erwähnt, läßt sich der Sprachschatz von AppleScript durch Scripting Additions erheblich erweitern. Für unsere Aufgabe benötigen wir einige dieser Helfer, nämlich „Tanaka's osax“, „Jon's Commands“, „GTQ Choose from list“ und die mit AppleScript gelieferten Additions. Zu beziehen sind diese unter anderem über die Suchmaschine www.shareware.com oder Apples neue AppleScript-Site (<http://applescript.fovista.com/as/sw.html#wheredownload>).

Der Ablauf. Die FileMaker-Datei bietet folgende Grundausstattung:

■ **Textfelder:** Titel, Text, Stichworte, Datei

■ **Textvariablen:** var_List, var_Key, var_Path, var_AS

■ **Layouts:** Eingabe, Liste, Intern
Das Layout „Intern“ verfügt über alle Felder und bleibt für den Benutzer unsichtbar. Es dient lediglich zum Einsetzen von Feldwerten aus der Zwischenablage. Die Variablenfelder sind in den Layouts „Eingabe“ und „Liste“ nicht vorhanden.

Das untenstehende FileMaker-Skript „Dokumente importieren“ besteht aus einer Master-Prozedur mit AppleScript-Part und der Subprozedur „Textdatei einsetzen“, die für jedes Dokument aus der Schleife aufgerufen wird. In der Subprozedur wird ein AppleScript „on the fly“ generiert und in einem zweiten Script-Maker-Schritt mit einem Wert des aktuellen Datensatzes bestückt.

FileMaker-Skript: Dokumente importieren

Fenster fixieren
Blättern aktivieren
Gehe zu Layout ['Intern']
Kommentar ['Layout mit Variablenfeldern']
Feld angeben ['var_List', '']
Kopieren [Auswählen, 'var_List']
Kommentar ['Zwischenablage löschen!']
Fehlerrückmeldung setzen [Ein]
Kommentar ['Modale Fehlermeldungen unterdrücken']
AppleScript ausführen [--Osaxen,...']
Kommentar ['Siehe 1. AppleScript']
Einsetzen [var_List]
Feld angeben ['var_Key', 'Links(var_List; Position(var_List; "¶"; 1; 1) -1)']
Kommentar ['Stichwort extrahieren und kopieren']

Feld angeben ['var_Path', 'Mitte(var_List; Position(var_List; "¶"; 1; 1) +1; ~ (Position(var_List; "¶"; 1; 2) -1) ~ Position(var_List; "¶"; 1; 1)']
Kommentar ['Ordnerpfad extrahieren und kopieren']
Feld angeben ['var_List', ~ 'Austauschen(var_List; var_Key & "¶" & var_Path & "¶"; "")']
Kommentar ['Variable auf Dateinamen einschränken']
Schleife
Skript ausführen ['Textdatei einsetzen']
Kommentar ['Subprozedur aufrufen']
Schleife-Verlassen-Wenn ~ ['MusterAnzahl(var_List; "¶") < 1']
Kommentar ['Container mit Dateinamen ist leerräumt']
Ende-Schleife
Fehlerrückmeldung setzen [Aus]
Gehe zu Layout ['Originallayout']

1. AppleScript. Im ersten Part soll AppleScript einen Container mit dem gesuchten Stichwort, dem ausgewählten Ordner und den gefundenen Dateinamen liefern. Folgende Dialoge und Optionen begleiten die Prozedur:

■ **a*:** Ordnerauswahl – wird kein Ordner ausgewählt, bricht die Prozedur ab.

■ **b*:** Stichworteingabe – erfolgt keine Stichworteingabe, werden alle Textdateien aufgelistet.

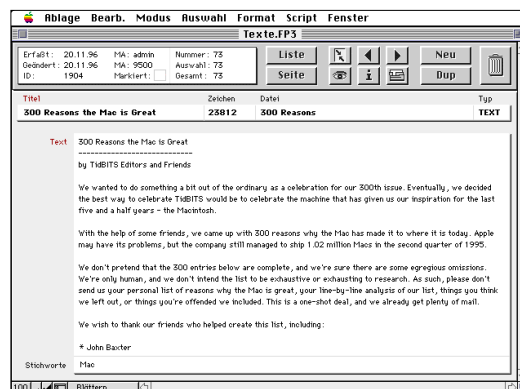
■ **c*:** Auswahlliste – aus den gefundenen Dateien läßt sich eine Unterauswahl zusammenstellen. Der „OK“-Button wird erst nach Auswahl mindestens einer Datei aktiviert. Andernfalls muß der Benutzer abbrechen.

Osaxen: Display dialog, Beep, Choose file, Tanaka's osax (1), GTQ Choose from list (2*), Jon's Commands (3*);
Ergebnistyp: Textcontainer in Zwischenablage,
Delimiter: Return; Ergebniswerte: Stichwort, Ordnerpfad, Dateinamen*

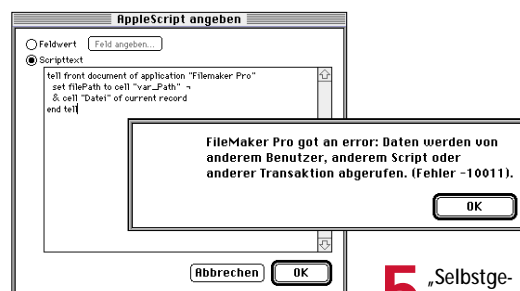
```
set mypath to (choose folder with prompt ~
"Textordner auswählen...") -- a*
set myDialog to display dialog ~
"Bitte ein Stichwort eingeben..." default
answer "" ~
buttons ("Abbrechen", "OK") default ~
button "OK" -- b*
if button returned of myDialog is "OK" then
set myKey to text returned of ~
myDialog
set myFileList to (dirLister mypath ~
type "TEXT" keyword myKey) -- 1*
set theCT to count (myFileList)
if theCT > 0 then
```



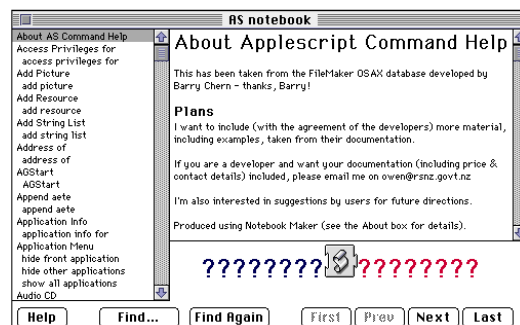
3 Nachdem die Erweiterung „Tanaka's osax“ alle Texte mit dem gesuchten Stichwort gefunden hat, übernimmt „GTQ Choose from list“ die Anzeige der eingeschränkten Dateiauswahl. Hier lassen sich auch diskontinuierliche Auswahlen treffen.



4 Ist der Import fehlerfrei verlaufen, sind alle Textdokumente in FileMaker verfügbar. Natürlich hätte man dem Finder noch mitteilen können, daß die importierten Dokumente gelöscht oder in einen anderen Ordner verschoben werden. Auch der Umgang mit dem 64-Kilobyte-Feldlimit in FileMaker wurde in dem Skript nicht berücksichtigt.



5 „Selbstgespräche“, wie sie ein Applet („tell me to...“) ermöglicht, quittiert FileMaker mit einer Fehlermeldung. Mit den erwähnten Techniken – über Zwischenablage und Austausch von Platzhaltern – kann die Kommunikation dennoch erfolgreich ablaufen.



6 Wer die AppleScript-Erweiterungen exzessiv nutzt, braucht ein schnelles „Lexikon“. Hier steht das gleiche, was der Skripteditor unter dem Befehl „Verzeichnis öffnen“ präsentiert. Allerdings kommt man über die Suchfunktion von „AppleScript Command Help“ wesentlich schneller zum Ziel.

```

set myFileList to choose from ~
list myFileList prompt ("Auswahl ~
enthält "" & myKey & ""; " & ~
theCT) cancel button ~
"Abbrechen" with multiple selec-
tions -- 2*, c*
else
beep
display dialog "Keine Datei ent-
spricht den Suchkriterien!" ~
buttons {"OK"} default button ~
"OK"
return
end if
try
set theCT to count (myFileList)
on error -- Abbruch im Auswahldialog!
return
end try
-- myFileList vor Konvertierung durch
Return trennen!
set oldDelimiter to AppleScript's ~
text item delimiters
set AppleScript's text item delimiters ~
to return
set the clipboard to (myKey & return ~
& myPath & return & (myFileList as ~
string) & return) -- 3*
set AppleScript's text item delimiters ~
to oldDelimiter
end if

```

Das AppleScript-Ergebnis wird über die Zwischenablage mit folgender ScriptMaker-Zeile in das Variablenfeld „var_List“ kopiert:

Einsetzen [Auswählen, 'var_List']

In folgender Form sollte das Ergebnis erscheinen:

Stichwort
Ordnerpfad
Datei 1
Datei 2
Datei n

Nun verteilen wir die Informationen des Containers an die richtigen Adressen. Zuerst wird das Stichwort mit folgender ScriptMaker-Zeile in die Variable „var_Key“ kopiert:

Feld angeben ['var_Key', 'Links(var_List; Position(var_List; "¶"; 1; 1)-1)']

Zeile 2 des Containers (Ordnerpfad) wird kopiert in die Variable „var_Path“:

Feld angeben ['var_Path', 'Mitte(var_List; ~ Position(var_List; "¶"; 1; 1) +1; ~ (Position(var_List; "¶"; 1; 2)-1) ~ Position(var_List; "¶"; 1; 1)']

Dann schränken wir den Inhalt von „var_List“ auf die Dateinamen ein:

Feld angeben ['var_List, ~ 'Austauschen(var_List; var_Key & "¶" & ~ var_Path & "¶"; "")']

Nun stehen in der FileMaker-Datei alle Informationen zur Verfügung, um in einer Schleife die Textdokumente einzulesen.

Subprozedur: Textdatei einsetzen. Dieses Skript erzeugt für jedes Dokument einen Datensatz, setzt über ein AppleScript den Text der Datei ein und füllt die Felder „Datei“, „Stichwort“ und „Titel“ automatisch aus.

Neuer Datensatz / Abfrage
Kopieren [Auswählen, 'Titel']
Kommentar ['Zwischenablage löschen!']
Feld angeben ['Stichworte', 'var_Key']
Feld angeben ['Datei', 'Links(var_List; Position(var_List; "¶"; 1; 1)-1)']
Kommentar ['Aktuellen Dateinamen aus ~ Liste kopieren.']
Feld angeben ['var_AS', ~ "try¶" & ~ "set the clipboard to (readFromFile ""File-Path""¶)¶" & ~ "on error¶" & ~ "beep¶" & ~ "end try"]
Kommentar ['Siehe 2. AppleScript']
Feld angeben ['var_AS', 'Austauschen(var_AS; "FilePath"; var_Path & Datei)']
Kommentar ['"FilePath" durch aktuelle Feldinhalte ersetzen!']
AppleScript ausführen ['var_AS']
Einsetzen [Auswählen, 'Text']
Kommentar ['Ergebnis des AppleScripts einsetzen.']
Feld angeben ['Titel', ~ 'Links(Text;Position(Text;"¶";1;1)-1)']
Kommentar ['1. Textzeile als Titel einsetzen.']
Feld angeben ['var_List', 'Rechts(var_List; Länge(var_List)-Position(var_List;"¶";1;1))']
Kommentar ['1. Zeile (Aktuellen Dateinamen) aus Container löschen.']

2. AppleScript. Da alle FileMaker-Objekte nur für Apple Events anderer Programme ansprechbar sind, muß man FileMaker überlisten. Wie also lassen sich aktuelle Feldwerte in ein von FileMaker ausgeführtes AppleScript einbinden? Natürlich könnte man wie im ersten AppleScript erneut die Zwischenablage bemühen. Bei unterschiedlichen Datentypen oder vielen Feldwerten gibt es aber Probleme. Die folgende Lösung ist nur dank der neuen Textfunktion „Austauschen“ möglich.

Die Originalfunktion in Apples Skripteditor lautet so:

Osaxen: Tanaka's Osax (1*), Jon's Commands (3*), Beep:
Parameter: Voller Pfadname der Textdatei;
Ergebnistyp: Textcontainer in Zwischenablage;
Ergebniswerte: Text einer Datei

```

on fileToClip(filePath)
try
set the clipboard to ~
(readFromFile filePath) -- 3*, 1*
on error
beep
end try
end fileToClip

```

Der Funktionsparameter („filePath“) ergibt sich aus den FileMaker-Feldern („var_Path & Datei“) und ließe sich aus jedem Applet über diesen Dreizeiler abfragen:

```

tell front document of application ~
„FileMaker Pro“
set filePath to cell „var_Path“ ~
& cell „Datei“ of current record
end tell

```

Soll das Skript von FileMaker ausgeführt werden, meldet das Programm lediglich eine Schutzverletzung. So bringen wir die Werte trotzdem erfolgreich unter:

```

"try¶" &
"set the clipboard to (readFromFile ""File-Path""¶)¶" &
"on error¶" &
"beep¶" &
"end try"

```

Dieser Text wird über die Funktion „Feld angeben“ in eine Variable („var_AS“) geschrieben. Über die Textfunktion „Austauschen“ wird dann der Platzhalter „FilePath“ durch die aktuellen Feldinhalte ersetzt. Der dritte Schritt „AppleScript ausführen, Feld angeben“ führt das Skript aus der Variablen „var_AS“ anstandslos aus. ■ *Martin Fuchs*



Vorschau

In der nächsten Folge beenden wir das Thema „AppleScript und FileMaker“ und geben abschließend Tips und Tricks zu den Themen Sicherheit und Tuning.