qToolBox – DoShellCommand

… und die Macht ist mit dir!

Holger Darjus

(Jg. 1962) ist Technischer Leiter des Medienvorstufen-Unternehmens EINSATZ Creative Production GmbH & Co. KG in Hamburg mit rund 45 Mitarbeitern und entwickelt seit 1996 FileMaker-Datenbanken speziell für die Medienvorstufe.

h.darjus@einsatz.de

Früher gehörte ich zur Spezies der Plugin-Verweigerer. Der Gedanke, dass eine Lösung nicht funktionieren könnte, weil ein Plugin fehlt, war mir so unsympathisch, dass ich auf den Einsatz gern verzichtete. Als aber irgendwann die Entwicklung einer FileMaker basierten Bilddatenbank auf dem Plan stand, kam ich an dem „Troi File-Plugin" nicht vorbei. Damit war das Eis gebrochen und meine Hemmschwelle, FileMaker Erweiterungen einzusetzen, deutlich gesenkt. Mittlerweile mag ich mir ein Arbeiten ohne Plugins gar nicht mehr vorstellen. Besonders ist mir die „ToolBox" von *Quart-EDV* ans Herz gewachsen. Diese Toolbox enthält eine auf den ersten Blick kleine, unscheinbare Funktion namens „DoShellCommand", die aber so mächtig ist, dass sich damit völlig neue Horizonte eröffnen.

Wie der Name schon vermuten lässt, ist dieses Plugin in der Lage, Shell-Kommandos auszuführen. Das bedeutet aber auch, dass der Leistungsumfang des Plugins im Wesentlichen von dem Wissen des Entwicklers abhängt. Wer es gewohnt ist, mit dem Terminal zu arbeiten, gewisse Unix-Grundlagen beherrscht und ein wenig Phantasie mitbringt, dem bietet dieses Werkzeug (auf einem Apple Macintosh) ungeahnte Möglichkeiten. Im Prinzip stehen einem das gesamte Leistungsspektrum vom Mac OS sowie alle Programme mit einem Command-Line-Interface (CLI) in FileMaker zur Verfügung. Was das in der Praxis bedeutet, möchte ich hier an einigen Beispielen aufzeigen.

Ping – ist da jemand?)

Ein zentrales Instrument unserer IT-Abteilung ist eine Datenbank, in der die gesamte Hard- und Software verwaltet wird. Die Komponenten werden einem Host (Netzknoten; Computer, Netzdrucker, … mit einer IP-Adresse) zugeordnet. Um zu überprüfen, ob dieser Host im Netzwerk erreichbar ist, sendet man ein so genanntes „ping" [1].

Wenn ich beispielsweise überprüfen möchte, ob die IP-Adresse 192.168.1.1 erreichbar ist, öffne ich für gewöhnlich ein Terminal und gebe „ping 192.168.1.1" ein. Ist die IP-Adresse erreichbar, könnte die gelieferte Antwort so aussehen:

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=18.176 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=8.450 ms
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.450/18.176/ ms
```

Sollte der Host nicht antworten, würden diese Zeilen zurückkommen:

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Für die Umsetzung des „ping" in der FileMaker Datenbank möchte ich das eigentlich gar nicht alles wissen. Mich interessiert nur, ob eine Maschine da ist oder nicht und ich möchte als Ergebnis einen booleschen Wert bekommen. Das übergebene Kommando muss also noch etwas verfeinert und dann in einem Script ausgewertet werden.

[1] *Erläuterungen dazu unter http://de.wikipedia.org/wiki/Ping_%28Datenübertragung%29*

Bei genauer Betrachtung der Ergebnisse gibt es einen bestimmten Wert, der darauf schließen lässt, ob es eine Antwort gab oder nicht: „... % packet loss“. Nach diesem Wert bzw. nach dem %-Zeichen lässt sich schon beim Absenden des Ping-Kommandos suchen. Nimmt man dann noch eines der mächtigsten Unix-Werkzeuge, nämlich **awk** zur Hilfe, um den String zu zerlegen, kann man mit einer Kommandozeile überprüfen, wie viele Pakete verloren gegangen sind. In dem genannten Beispiel sähe der Befehl so aus:

```
ping -c 1 -t 1 192.168.1.1 | grep % | awk '{print $7}'
```

Bei Erreichbarkeit liefert der Befehl „0%“, bei Nichterreichbarkeit „100%“ zurück. Eingebunden in FileMaker müsste der Scriptschritt so aufgebaut werden.

```
◆ Variable setzen
   Name:    $Echo
   Formel:  LiesAlsZahl (qToolBox_DoShellCommand ("ping -c 1 -t 1 " &
            Hosts::IPAdresse & " | grep % | awk '{print $7}'"))
```

Alles Weitere ist dann wieder FileMaker typisch. Jetzt kann in Abhängigkeit der Variable **$Echo** weiter verfahren werden. So kann beispielsweise ein Statusfeld auf „OK“ gesetzt werden, wenn der Wert der Variable **$Echo** = „0“ ist.

Dieses kleine Beispiel ist zwar noch nicht von besonders großem praktischem Nutzen, soll aber zunächst einen Einblick in die Funktionsweise des Plugins geben. Das Prinzip ist einfach. Es kann ein Unix-Kommando unter Einbindung von Feldinhalten zusammengesetzt werden und die Ergebnisse können wiederum in Feldern oder Variablen übernommen werden.

Überschrift ???

Richtig spannend wird es, wenn die zahlreichen, mitunter sehr umfangreichen Funktionen in Mac OS angesprochen werden. „Unter der Haube“ verstecken sich nämlich derart viele Möglichkeiten, dass dem ambitionierten Entwickler Tür und Tor in alle nur erdenklichen Richtungen geöffnet sind. Eines dieser Werkzeuge ist das „scriptable image processing system“, kurz **sips**. Damit lassen sich Bilddateien manipulieren und auslesen. Auch hierzu ein Beispiel: Ist in einer FileMaker Datenbank beispielsweise ein Pfad zu einem Bild hinterlegt und man möchte weiterführende Informationen zu diesem erhalten, reicht eine einzige Code-Zeile, um diese zu bekommen. Im Terminal würde ich Folgendes eingeben:

```
sips --getProperty all /{Pfad_zum_Bild}/Bild.tif
```

Die Antwort könnte dann so lauten:

```
pixelWidth: 713
pixelHeight: 3013
typeIdentifier: public.tiff
format: tiff
formatOptions: lzw
dpiWidth: 304.800
dpiHeight: 304.800
samplesPerPixel: 5
bitsPerSample: 8
hasAlpha: yes
space: CMYK
profile: ISO Coated
creation: 2008:01:17 10:56:54
software: Adobe Photoshop CS3 Macintosh
```

Wiederum in FileMaker umgesetzt, könnte ein Formelfeld folgende Definition haben:

```
qToolBox_DoShellCommand ("sips --getProperty all " & Pfad_des_Bildes)
```

Dies würde das gleiche Ergebnis wie oben in einem Textfeld darstellen.

Mit **sips** lassen sich die Parameter aber nicht nur abfragen, sie lassen sich auch setzten. Das bedeutet, **sips** kann Bilder bearbeiten beziehungsweise verändern. Drehen, beschneiden und die Größe ändern sind dabei die einfachsten Verfahren.

So können durchaus auch komplexe Vorgänge wie ICC-Profil-basierende Farbraumveränderungen vorgenommen werden. Wer sich für das volle Spektrum dieses Programms interessiert, öffnet am Besten einfach mal das Terminal und tippt „`man sips`“ ein. Das öffnet die „online manual pages“ zu dem Programm und erläutert den Funktionsumfang sowie die Syntax.

Mit etwas Geschick kann man mit diesem Werkzeug FileMaker zu einem Bildkonvertierungs-Tool machen. Denn die Parameter der Kommandos, die an **DoShellCommand** übergeben werden, können selbstverständlich auch aus Formeln übernommen werden.

Überschrift ???

Im letzten Beispiel möchte ich auf die Möglichkeit hinweisen, wie mit dem Dateisystem interagiert werden kann.

Angenommen, es soll ein Bild in ein bestimmtes Verzeichnis bewegt werden. Dieses Verzeichnis existiert aber noch nicht und muss vorher angelegt werden.

Auch hier genügt wieder eine einzige Zeile, um einen beliebigen Pfad, beispielsweise errechnet aus mehreren Feldwerten, zu erzeugen. Das Kommando dazu heißt „mkdir“ (make directories). Der Clou dabei: dieses Kommando kennt

die Option „-p“, die bewirkt, dass sämtliche „Parent-Ordner“ – also alle übergeordneten Ordner – gleich mit erzeugt werden.

Die Umsetzung in FileMaker könnte dann so aussehen:

```
◆ Variable setzen
   Formel:  qToolBox_DoShellCommand ("mkdir -p /Users/Benutzer/Desktop/"
            & Feld1 & "/" & Feld2 {...})
```

Anschließend kann das Bild mit den Befehlen „cp“ oder „mv“ in das Zielverzeichnis bewegt werden. Natürlich stehen auch alle anderen Befehlssätze zum Löschen, Verschieben oder Duplizieren zur Verfügung.

Zu beachten ist, dass sich Pfade im Terminal von den FileMaker Pfaden unterscheiden:

Unix trennt diese mit „/“ (Slash) und nicht mit „:“ (Doppelpunkt). Auch unterscheiden sich die absoluten Pfadangaben. Dies ist für den geübten Entwickler aber sicher keine Schwierigkeit. Liefert uns FileMaker doch so hilfreiche Funktionen wie „Austauschen“ und „ZeichenMitte“.

Etwas unangenehmer kann es werden, wenn man es mit Umlauten und dergleichen zu tun bekommt. Aber auch in diesem Fall kann unter Zuhilfenahme des Plugins wieder das Betriebssystem angesprochen werden. Dieses liefert nämlich auch gleich den Dolmetscher **iconv** mit. Mit diesem Programm lassen sich die Encodings der Zeichensätze einzelner Werte sehr komfortabel konvertieren. Das Plugin selbst beherrscht das aber auch sehr gut und so muss man sich in der Regel darum nicht mehr bemühen. Unter dem neuen Mac OS 10.5 (Leopard) wurde dem **Terminal** in Sachen Unicode auch noch einmal deutlich auf die Sprünge geholfen.

Fazit

Mit diesem Plugin lassen sich sehr komplexe Workflows abbilden. Ein konkretes Anwendungsbeispiel könnte das Publizieren von in einer FileMaker Datenbank verwalteten Bildern auf einem Web-Server sein. (Dass dieses auch mit Instant- oder Custom-WebPublishing möglich ist, lasse ich hier mal außer Acht). So können Bilder mit einem Klick aus FileMaker heraus in den RGB-Farbraum konvertiert, dann mit mit dem CLI von **StuffIt** zu einer ***.sit-*** oder ***.zip***-Datei komprimiert und anschließend mit **ftp** an einen FTP-Server übertragen werden. Bei erfolgreicher Veröffentlichung bekommt der entsprechende Datensatz den Status „online“. Dieses Verfahren eignet sich übrigens auch hervorragend für Backups auf einem Remote-Backup-Server.

Mit € 49 für den Einzelplatz ist dieses Plugin ein wahres Schnäppchen. Zumal **DoShellCommand** nur eine von insgesamt 16 Funktionen ist. Nach meinem Wissensstand gibt es noch zwei weitere Plugins, eines davon sogar kostenfrei, die eine ähnliche Funktion wie **DoShellCommand** bieten. Das aus dem Hause ***Quart-EDV*** ist jedoch das einzige Plugin, das auch auf Intel-Macs laufende und Unicode beherrschende Plugin. Eine Demo-Version kann beim Hersteller[2] oder beim K&K Verlag[3] geladen werden.

Wer Lust bekommen hat, seine FileMaker Lösungen mit Shell-Programmierungen aufzubohren, findet unter anderem in den nachfolgend genannten Büchern:

von bzw. Dave Taylor zahlreiche Anregungen und Tipps.

Titel	**Shell-Skript Programmierung**
Autor	**Patrik Ditchen**
Broschiert	**832 Seiten**
Verlag	**Mitp-Verlag**
Internet	**www.mitp.de**
Auflage	**3. A. (November 2007)**
ISBN-10	**3826617991**
ISBN-13	**978-3826617997**
Preis	**€ 44,95**

Titel	**Raffinierte Shell Scripts**
Autor	**Dave Taylor**
Taschenbuch	**374 Seiten**
Verlag	**Mitp-Verlag**
Internet	**www.mitp.de**
Auflage:	**1 (Oktober 2004)**
ISBN-10:	**3826615107**
ISBN-13:	**978-3826615108**

FMM_200801

[1] *http://www.quart-edv.de/Plugins/toolbox.html*

[2] *http://filemaker-magazin.de/*