Holger Darjus

(Jg. 1962) Entwickelt seit 1996 FileMaker Lösungen für Prozesssteuerung, Auftragsabwicklung und Informationsmanagement im Bereich Marketing, Handel und Industrie. Besondere Ausrichtung: benutzerorientierte Gestaltung nach EN ISO 9241.

holger@darjus.de

Nachdruck aus dem FileMaker Magazin
siehe auch: www.filemaker-magazin.de

Ist doch logisch, oder?

Worauf ein Entwickler bei der Planung oder Weiterentwicklung seiner Datenbanken achten sollte

Die Leser dieses Magazins werden eines gemeinsam haben: eine gewisse Verbundenheit zu den Produkten aus dem Hause FileMaker. Die Arbeit mit den Anwendungen macht einfach Spaß und bietet im Vergleich zu vielen „Wettbewerbs-Produkten" eine überzeugende Vielzahl an Vorzügen. Egal, ob FileMaker als Ersatz für eine Tabellenkalkulation oder als „echte" Datenbank mit vielen gleichzeitig zugreifenden Anwendern eingesetzt wird – die meisten Anforderungen lassen sich konkurrenzlos-schnell und elegant realisieren.

Ich kann mich zwar nicht mehr an den konkreten Anlass erinnern, aber als ich vor über 20 Jahren das erste Mal etwas mit FileMaker „programmiert" habe, war ich restlos begeistert. Zuvor hatte ich mit zahlreichen, heute längst vergessenen Datenbanken und einer Sybase-SQL-Datenbank entwickelt und auch Quälereien mit Excel-Tabellen gehörten zur Tagesordnung. Umso mehr war ich angetan von der Vielfältigkeit und der leichten Umsetzbarkeit der Funktionen innerhalb von FileMaker. Besonders die Möglichkeit, sich wiederkehrende Aufgaben von Scripts abnehmen zu lassen, machte den Charme dieser Software aus.

Bis zum heutigen Tage sind unter Zuhilfenahme von FileMaker mittlerweile ungezählte Lösungen entstanden – sowohl kleine „Helferleinchen" als auch große und sehr komplexe Lösungen, mit denen mehrere hundert Anwender arbeiten. Insbesondere die zuletzt genannten Lösungen bedürfen aber mittlerweile einer Planung und Struktur, die deutlich über die bloßen Überlegungen zur Lösung einer Aufgabe hinausgehen. Oft wird über Namenskonventionen und Ähnliches diskutiert, damit „fremde" Entwickler eine Richtlinie haben und die Datenbanken auch im Team bearbeitet werden können.

Aus meiner Sicht ist das bei Weitem nicht mehr ausreichend. Vor einiger Zeit habe ich einen externen FileMaker Entwickler zur Unterstützung bei einem großen Projekt engagiert, der sich in sehr kurzer Zeit in die komplexe, multilinguale Lösung eingearbeitet hat. Als ich aber einmal auf einer Geschäftsreise war, wurde er mit einer Problematik konfrontiert, bei der er meine Unterstützung benötigte. Wir telefonierten und ich versuchte ihm zu erklären, worauf er achten muss, um die Änderungswünsche des Kunden umzusetzen. Während dieser Telefonate ist mir bewusst geworden, an welchen Stellen überall „Programm-Logik" vorkommt. Obgleich FileMaker von mir sehr geschätzt wird, zeigt sich hier eine Schwachstelle, die es schwer macht, einheitliche Programmierstile zu entwickeln oder zu etablieren.

An welchen Stellen finden wir überall Logik?

Das erste Mal begegnet uns das Logik-Thema bei der Erstellung einer Datei, denn bereits der Dateiname kann Auswirkungen auf das Verhalten der Lösung haben. Weiterhin können in den „Dateioptionen..." Einstellungen vorgenommen werden, die ebenfalls Einfluss auf das Verhalten der Datenbank nehmen. Dort kann beispielsweise festgelegt werden, zu welchem Layout als Erstes gesprungen werden soll. Nachdem wir eine Datei angelegt haben, beginnen wir in der Regel mit der Definition von Feldern, wo wir auf eine der vermutlich am häufigsten verwendeten logischen Funktionen treffen. Felder können einen „Berechneten Wert", automatische Eingabeoptionen oder Referenzwerte enthalten. Wir können Formeln definieren, automatische Überprüfungen einrichten, dafür sorgen, dass in einem Feld nichts eingegeben werden kann sowie die Standardsprache und die Indizierung für einzelne Felder festlegen.

Wie bei der Datei selbst kann auch bei den Feldern bereits die Benennung Einfluss auf die Logik und Struktur der Datenbank nehmen. Sind die Felder angelegt, widmen sich die meisten Programmierer dem Beziehungsdiagramm, wo ein weiterer, sehr wesentlicher Teil der Logik festgelegt wird, wie die folgenden Fragen verdeutlichen sollen: Wie stehen die einzelnen Tabellen zueinander? Über welche Felder werden sie verknüpft? Dürfen über die Beziehung Bezugsdatensätze angelegt werden und werden diese Bezugsdatensätze ebenfalls gelöscht, wenn der „Anker" gelöscht wird? Sind die Bezugsdatensätze in irgendeiner Weise sortiert? Ist die „Beziehungsarbeit" abgeschlossen, beginnen wir mit der Erstellung der Layouts – ein weiterer, bunter Strauß an Möglichkeiten, logische Funktionen zu verbauen: „Bedingte Formatierung", „Objekt ausblenden, wenn", Farben, Linien, Verankerungen, Stile, Größen, um nur einige zu nennen. Nicht zu vergessen die Layouts selbst und ihre Namen. All dies hat nicht nur Auswirkungen auf das Aussehen unserer Datenbanken, sondern auch auf ihr Verhalten und sind somit integrale Bestandteile der Logik.

Aber hiermit nicht genug. Im Zusammenhang mit der Erstellung der Layouts entstehen auch die ersten Scripts. Hier liegt vermutlich der Schwerpunkt der Logik, denn mit Scripts hauchen wir unseren Datenbanken erst richtig Leben ein: Benutzer werden mit Navigations-Scripts durch die Lösung geführt, Fenster und Popover werden geöffnet und geschlossen, wiederkehrende Aufgaben automatisiert, Daten im- und exportiert, Berechnungen ausgeführt, Daten und Datensätze manipuliert – kurzum: Mit Scripts machen wir unsere Datenbanken zu „Programmen". Einen ebenfalls nicht unerheblichen Einfluss auf das Verhalten unserer „Programme" nimmt die Verwaltung von Benutzerkonten, Berechtigungen, Erweiterten Zugriffsrechten und Dateizugriffen. Mit ihnen legen wir fest, ob ein Anwender Datensätze anlegen und/oder löschen darf, welche Layouts, Felder und Inhalte er zu Gesicht bekommt, ob er das Recht hat, zu drucken und/oder exportieren, welche Scripts er ausführen und welche Wertelisten er sehen oder gar bearbeiten darf usw.

Sind die Einstellungen zur „Sicherheit" getätigt, stürzen sich viele von uns auf die „Angepassten Menüs...", wo wir ebenfalls erheblichen Einfluss auf die Logik nehmen können, indem wir Standardfunktionen „verbiegen" und die Funktionen in der Menüleiste nach Belieben erweitern oder reduzieren.

Zu guter Letzt können wir mit „Eigenen Funktionen" und Plugins die Programmlogik derart beeinflussen, dass das Produkt FileMaker kaum mehr wiederzuerkennen ist.

Haben wir unsere Entwicklung abgeschlossen, stellt sich die Frage, was wir mit der mit Logik vollgestopften Lösung überhaupt machen. Wird daraus eine Runtime-Anwendung, publizieren wir sie per WebDirect oder als Datenquelle für Webseiten unter „Custom Web Publishing", arbeiten Menschen damit und wenn ja, auf welchen Geräten (Desktop Win/Mac, FileMaker Go für iOS), läuft sie auf einem Server oder als Einzelplatzlösung oder dient sie lediglich zur Datenaufbereitung?

Fassen wir noch einmal kurz zusammen, an welchen Stellen wir in FileMaker Logik einbauen können:

- Dateioptionen
- Felder (und deren Optionen)
- Beziehungen
- Layouts
- Zugriffsberechtigungen
- Menüleiste
- Scripts
- Eigene Funktionen
- Plugins
- Stile (CSS)
- Art der Bereitstellung

... und bestimmt noch das eine oder andere, an das ich nicht gedacht habe.

Einige Beispiele, wie diese atemberaubende Vielfalt zu einem Problem werden kann

Zu Beginn dieses Artikels habe ich die Vielzahl an Möglichkeiten als „Schwachstelle" bezeichnet und manch einer wird sich vielleicht nach dem Warum fragen. Bieten uns doch all die aufgezählten Möglichkeiten die bunte Welt der Programmierung und sind zum täglichen Brot für uns Entwickler geworden. Ja, wir genießen sie geradezu, diese Vielfältigkeit, die uns immer wieder dazu veranlasst zu sagen „mit FileMaker geht das".

Doch spätestens wenn es um die Fehlersuche geht, kann diese Verteilung ihre Tücken haben. Auch dann, wenn eine Lösung auf unterschiedliche Weise genutzt werden soll, indem sie durch native FileMaker „Clients", per WebDirect und unter Verwendung von FileMaker Go geöffnet wird, kann es sinnvoll sein, auf einige der gebotenen Möglichkeiten zu verzichten und sie stattdessen in andere Bereiche zu verlagern.

Einige Beispiele:
Ein Kunde oder Nutzer kontaktiert Sie, weil auf einem Ausdruck eine Textzeile „merkwürdig" aussieht. Während der restliche Text ganz „normal" ist, wird diese eine Zeile kursiv und fett gedruckt. Sie als Entwickler haben vielleicht schnell eine Idee, wonach Sie suchen müssen. Falls Sie aber grade im Urlaub sind und einer Ihrer Kollegen sich um dieses Problem kümmern muss, wird es schon schwieriger. Mit etwas Glück hat er den betroffenen Datensatz schnell gefunden, um dann festzustellen, dass der Inhalt des Textfeldes keinerlei Auffälligkeiten aufweist. Im nächsten Schritt wirft er den Script-Debugger an und lässt das zum Druckvorgang gehörende Script durchlaufen, um zum entsprechenden Drucklayout zu gelangen. Dabei stellt er fest, dass er nicht über die gleichen Zugriffsrechte wie der Anwender verfügt und das Drucklayout gar nicht betreten darf.

Er meldet sich also mit einer anderen Benutzerkennung an. Allerdings wird das Layout nun nicht mehr angesprungen, weil der Anwender das Script zum Drucken bereits ausgeführt und damit eine neue Bedingung für das Verhalten des Scripts geschaffen hat. Ihr Kollege beginnt also, sich das Script durchzulesen, um herauszufinden, wohin es beim ersten Durchlauf führt. Geschafft! Das Layout ist gefunden und dem Kollegen ist es sogar gelungen, die passenden Datensätze aufzurufen. Aber was ist das? Bei ihm sieht alles ganz normal aus. Es gibt keine bedingte Formatierung für das Feld und er kann sich beim besten Willen nicht erklären, wieso der Text fett und kursiv sein soll. Also nimmt er sich das Script noch einmal vor und entdeckt einen „Feldwert setzen"-Aufruf. Als er sich diesen genauer anschaut, handelt es sich lediglich um das Einsetzen eines „nackten" Textes in ein Feld. Nun ist Ihr Kollege mit seinem Latein fast am Ende, aber dann fällt ihm noch etwas ein: die Feldoptionen. Und tatsächlich: In der Datenbankverwaltung sieht er, dass das Feld mit einer automatischen Berechnung versehen ist, die da lautet: „CF_NumberFormat (Number; DecimalPrecision; ThousandSeparator)". Seufzend ruft der Kollege nun die Verwaltung der „Eigenen Funktionen" auf und studiert die aufgerufene Funktion. Sofern er versteht, was dort berechnet und formatiert wird, muss er sich jetzt nur noch trauen, die Funktion so anzupassen, dass der Anwender seinen Ausdruck in gewünschter Form erhält, aber der Rest der Anwendung nicht beeinflusst wird.

Zugegebenermaßen handelt es sich bei diesem um eine etwas überzogene Beschreibung einer Fehlersuche, aber Hand aufs Herz: So ähnlich ist es sicher schon vielen von uns gegangen. Weitaus weniger theoretisch ist die Bereitstellung einer Datenbank für unterschiedliche Zugriffsarten. Immer häufiger ist es gängige Praxis, dass eine Datenbank zeitgleich von FileMaker Clients, WebDirect-Anwendern und mit FileMaker Go geöffnet wird. Nach Möglichkeit vermeiden wir es natürlich, für jeden Anwendungsfall spezielle Scripts zu schreiben, sondern versuchen im Sinne einer Modularisierung die Scripts und Funktionen übergreifend einzusetzen. Nutzer von FileMaker Pro Advanced können sich die Kompatibilität der einzelnen Script-Schritte für das jeweilige Umfeld anzeigen lassen, was ein großer Vorteil ist. Um die Erstellung separater Layouts wird man in vielen Fällen jedoch nicht herumkommen. Hierfür gibt es leider auch bei FileMaker Pro Advanced keine Unterstützung. Denn nicht alle Attribute, die wir einem Layout-Objekt mit auf den Weg geben, funktionieren auch in allen Welten. Schmerzlich wird man beispielsweise feststellen müssen, dass sich die lange ersehnte und großartige Funktion „Objekt ausblenden, wenn" insbesondere bei Verwendung mit WebDirect nicht so verhält, wie wir es im nativen Umfeld von FileMaker gewohnt sind. Wendet man diese Funktion beispielsweise auf die Taste eines „Popovers" an, in dem sich ein Objekt befindet, das durch den Script-Schritt „Gehe zu Objekt [...]" angesteuert wird, öffnet sich das Popover unter WebDirect leider nicht.

Ein weiteres Beispiel sind die gern und häufig verwendeten Formelfelder. Schnell und einfach bereiten wir uns damit die tollsten Inhalte auf. Allerdings kann der Preis dafür recht hoch sein, insbesondere dann, wenn die Ergebnisse nicht gespeichert, sondern bei Bedarf neu berechnet werden. In diesem Fall kann die Geschwindigkeit der Datenbanken dramatisch einbrechen, was besonders tragisch ist, wenn die Lösung über „dünne" Internetverbindungen bereitgestellt wird. Also gilt es auch hier, genau zu planen, wo die Logik untergebracht wird.

Nun lässt sich leider nicht alles an eine Stelle verlegen. Jedoch verfolge ich bei den Überlegungen zu meinen Lösungen seit Langem einen strengen Vorsatz, den ich den Lesern dieses Artikels wärmstens ans Herz legen möchte: So viel wie möglich in Scripts packen!

- In meinen aktuellen Datenbanken sind so gut wie keine Formelfelder oder Felder mit berechneten Werten enthalten. Alle Berechnungen finden in den Scripts statt, die die Datensätze anlegen oder werden nach einer Anwenderaktion durch Script-Trigger durchgeführt. Das hat gleich mehrere Vorteile: Zum einen muss ein Supporter niemals in die Datenbankverwaltung bzw. in die Felddefinitionen gehen. Außerdem laufen die Lösungen auf iOS-Geräten selbst dann noch sehr performant, wenn sie mehrere Tausend oder gar Millionen Datensätze enthalten. Die möglichen Quellen bei einer Fehlersuche werden um einen Punkt verringert und die Im- und Exporte sind wesentlich schneller.

- In meinen Lösungen gibt es lediglich fünf Benutzerkonten, die immer gleich bleiben. Alle Zugriffsberechtigungen werden durch eine integrierte, in einer eigenen Tabelle hinterlegten Benutzerverwaltung gesteuert und innerhalb von Script-Aufrufen berücksichtigt.

- Bei Tasten, die durch „Objekt ausblenden, wenn" versteckt werden, verlasse ich mich nicht darauf, dass ein auf der Taste liegendes Script nicht ausgeführt wird, weil die Taste nicht sichtbar ist. Die Bedingung, die die Taste ausblendet, wird im Script nochmals abgefragt und beendet das Script gegebenenfalls. Auch das verringert mögliche Fehlerquellen, wenn man das Script beispielsweise auf eine andere Taste legt und vergisst, die Ausblendbedingung zu setzen.

- Im Beziehungsdiagramm werden so wenig Tabellenauftreten wie möglich angelegt. Niemals wird die Option gesetzt, dass beim Löschen eines Datensatzes auch die Bezugsdatensätze gelöscht werden, und die Sortierung der Bezugsdatensätze geschieht nur in Ausnahmefällen.

Durch eine derartige Vorgehensweise lässt sich die Logik unserer Datenbanken leider noch nicht an einen zentralen Ort verlagern. Dennoch können wir die Zahl der möglichen Fehlerquellen deutlich reduzieren und schaffen – neben Konventionen – klare Strukturen, die wiederum den Weg für modulare Entwicklungen bereiten. Die fehlende Unterstützung bei der Modularisierung ist nämlich eines der Mankos von FileMaker. Wenn ich an die Zeiten meiner Sybase-Entwicklung zurückdenke, fällt mir auf, wie viel strukturierter die Entwicklungsumgebung dort war. Zwar gab es auch dort zwei Teile – das Interface und die Datenbank selbst –, aber

alles, was in der Datenbank passierte, wurde im sogenannten EO-Modular definiert. Das war der zentrale Punkt der Logik. Wenn es in einem Layout eine Taste gab, dann wurde die Ablaufroutine, das Aussehen der Taste und die Bedingung, ob das „Script" überhaupt ausgeführt werden darf, in einem „Objekt" definiert – klar und einfach.

Letztendlich schätzen wir an FileMaker die bunte Vielfalt, die immer wieder zum Experimentieren einlädt. Wollen wir aber große Projekte mit FileMaker stemmen, sollten wir beim Experimentieren gelegentlich den Blick über den Tellerrand wagen und uns mit „handelsüblichen" Datenbanken und ihren Prozeduren beschäftigen. Dabei geht es nicht um das Suchen einer Alternative, sondern vielmehr darum, sich einiges „abzugucken". Denn wonach wir schon seit vielen Jahren Ausschau halten, ist in anderen Programmierwelten längst realisiert: Standards zur Entwicklung – auch wenn diese oftmals entstanden sind, weil es gar nicht anders geht. ♦

FileMaker Magazin

Das FileMaker Magazin

- Einzige, deutschsprachige Fachzeitschrift zu FileMaker
- Wissen aus erster Hand von anerkannten FileMaker Fachautoren
- Große Themenvielfalt für Anwender, Entwickler und Fortgeschrittene

Exklusiv für Premium-Abonnenten

- Sechs FMM Ausgaben pro Jahr
- Kostenlose Nutzung des Abonnentenbereichs auf ***www.filemaker-magazin.de***
- PDF-Online-Archiv mit allen bisher erschienenen Ausgaben
- Jede Ausgabe mit kostenlosen Beispieldateien und Zusatzinfos zum Download

Unser Service

- Aktuelle Neuheiten, Tipps und Infos, Kleinanzeigen und vieles mehr jederzeit online auf unseren Webseiten
- Hilfe bei allen Fragen zu FileMaker im FMM Forum
- Kompetente Beratung zum Kauf von FileMaker Lizenzen: Einfach anrufen +49 (0)40 589 65 79 70.

Hier finden Sie Aktuelles zu FileMaker Produkten, egal ob Sie kaufen, mieten oder sich einfach informieren möchten.

Eine kostenlose Leseprobe des FileMaker Magazins erhalten Sie, wenn Sie hier klicken.

Wenn Sie sich für ein FileMaker Magazin Abo interessieren, klicken Sie bitte hier!

K&K Verlag GmbH · www.filemaker-magazin.de · Telefon +49 40 589657970