mittel

Mit FileMaker shoppen

Wie aus einem Katalog ein Webshop wird

Maik Berchtold

(Jg. 1976) ist seit 1998 als FileMaker Entwickler tätig. Sein Schwerpunkt liegt in der Entwicklung branchenspezifischer Software für die Industrie und der Transportlogistik. So entwickelte sich ein Kundenstamm von Bremen bis nach Kapstadt/ Südafrika. Motto: Geht nicht, gibt's nicht!

MBerchtold@i-pro.de

Als FileMaker Entwickler kommt man irgendwann in die Situation, dass der Kunde einen Webshop benötigt. Wenn man HTML und PHP aus dem Effeff beherrscht, ist das sicherlich kein Problem. Doch wie sieht es aus, wenn man darin keine oder nur Grundkenntnisse hat? FileMaker bietet hierfür via IWP einige Möglichkeiten an und mit ein paar Tricks baut man Shops, die auch mit etablierten Anwendungen mithalten können.

Die Grundidee entstand – wie sollte es anders sein – aufgrund der Anfrage eines langjährigen Kunden. Dieser benutzt schon seit einigen Jahren eine FileMaker Katalogdatenbank zur Generierung von PDF-Katalogen und zur Verwaltung seines Sortiments, das noch in einer anderen Software separat verwaltet wird. Die Funktionen und Einschränkungen, die das FileMaker Instant Web Publishing mit sich bringt (z. B. können Feldeingaben erst mit einem Klick in das Feld aktiviert werden oder der Befehl **„Schreibe Änderung Datens./ Abfrage"**, der als Eingabebestätigung dient, wird so leider nirgends aufgeführt), waren jedoch glücklicherweise durch andere Projekte bekannt.

Der vorhandene FileMaker Katalog ***LM_Artikel.fp7*** aus FileMaker 6 Zeiten war natürlich nicht mehr up to date und musste ersetzt werden. Die Felddefinitionen und Wertelisten wurden modifiziert und importiert und die Datenbank so Schritt für Schritt auf FileMaker 9 gebracht. Parallel dazu wurde die neue Datei ***LM_Web.fp7*** geschaffen, die als Datei für das IWP fungierte.

Aufbau

Der bisher verwendete Webshop der ***Smart Store AG*** hatte seine Lebensdauer bereits überschritten. Der Wunsch nach einem Bereich für Endkunden ohne Preise sowie einem Bereich für Händler mit Login- und Bestellfunktion und die Anforderung, dass diese beiden Bereichen zweisprachig, also deutsch und englisch, sein sollten, überforderten das bisherige Konzept. Allein die Kosten für die monatliche Aktualisierung wuchsen stetig.

Wir entschieden uns dafür, dass die Artikeldatei beim Kunden verbleibt und via **Remote-Desktop** eine Verbindung zur IWP-Datei hergestellt wird. Darüber lassen sich die Artikel im bisherigen Prozess pflegen, ohne dass großartige Veränderungen auf den Mitarbeiter zukommen. Geplant wurde so, dass diese „Artikel-Katalog-Datei" später zu einem kleinen Warenwirtschaftssystem umgebaut werden kann. Doch dazu später mehr. Nun begann unsere Werbeagentur, den „neuen" Shop mit Seiten, Texten und Bildern zu bestücken, damit die Programmierung beginnen konnte.

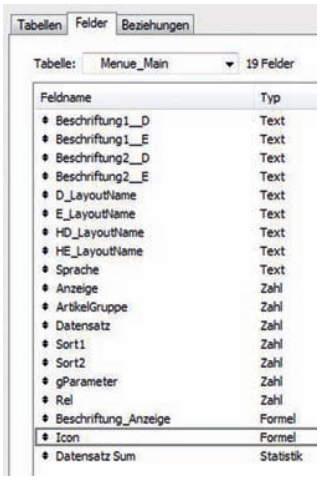
Look & Feel

Große Bedeutung für jedes Unternehmen hat ein einheitliches Erscheinungsbild (Corporate Identity) oder kurz CI. So auch in diesem Fall. Header und Footer sollen der bisherigen CI entsprechen. In der Funktionalität wollten wir das Menü über ein Portal realisieren. Der Vorteil liegt darin, dass die Webseiten später zwar teilweise statisch sind, jedoch über das Menü-Portal nur an einer Stelle gepflegt werden müssen. So spart man sich viel Zeit, wenn eine Seite im Menü hinzugefügt oder gelöscht werden soll.

[1] *FMM_200903, S. 5 ff. „Dialog mit der Datenbank"*

[2] *Othmar B. Strässle, Strässle Datenbanklösungen; Zuzwil (CH); straessle@datenbankloesungen.ch*

Für den Händlerbereich sollte ein separates Menü verwendet werden – schon allein wegen der Bestellmöglichkeit, die es im Endkundenbereich nicht gibt. Also erstellten wir in der ***IWP***-Datei die beiden Tabellen ***Menue_I*** und ***Menue_II***. Zwecks Mehrsprachigkeit haben wir die Felder jeweils in Deutsch und in Englisch erstellt.



Wenige Felder genügen, um das Menü flexibel zu halten.

Diese werden mit einem Formelfeld verknüpft, das über das Feld **Sprache** – dessen Wert zu Beginn einer Sitzung oder beim Anklicken der jeweiligen Länderflagge auf der Startseite gesetzt wird – die Auswahl trifft.

%HVFKULIWXQJB$Q]HLJH

```
)DOOV
6SUDFKH
:HQQ,VW/HHUHVFKULIWXQJBB                BB7H[W)RQW
HVFKULIWXQJBBYHUGDQD                    BHVFKULIWXQJBB
6SUDFKH +
:HQQ,VW/HHUHVFKULIWXQJBB                BB7H[W)RQW
HVFKULIWXQJBBYHUGDQD                    BHVFKULIWXQJBB
6SUDFKH (
:HQQ,VW/HHUHVFKULIWXQJBB(               BB7H[W)RQW
HVFKULIWXQJBB(YHUGDQD                   BHVFKULIWXQJBB(
6SUDFKH +(
:HQQ,VW/HHUHVFKULIWXQJBB(               BB7H[W)RQW
HVFKULIWXQJBB(YHUGDQD                   BHVFKULIWXQJBB(
```

Eine simpel definierte Formel steuert die Sprachausgabe des Menüs.

Dieses Menü lässt sich später auch problemlos um eine zusätzliche Sprache zu erweitern. Das Feld **Beschriftung1_D** steht für einen Hauptmenüpunkt mit einem Hauptmenü-Logo in Deutsch, das Feld **Beschriftung2_E** steht für einen Untermenüpunkt mit einem Untermenü-Logo in Englisch. Leider gibt es in diesem Shop jedoch keine Verwendung für einen Untermenüpunkt. Die Auswahl des Menüpunkt-Logos erfolgt aber über das Feld **Icon**, das ebenfalls über eine Formel die Funktion „IstLeer (Beschriftung2_D)“ abfragt.

Das Menü nun auf jedem Layout platzieren – und schon lässt sich eine Struktur erkennen. Der nächste Schritt bestand darin, die Navigation über das Menü zu realisieren. Dafür erstellten wir die Felder **D_LayoutNam**e und **E_LayoutName**, in denen die Layoutnamen für Deutsch und Englisch hinterlegt wurden. Über die auf der Startseite festgelegte Sprache wird die globale Variable **$$Language** generiert, die bei jeder Menüaktion abgefragt bzw. in das globale Feld Sprache als Feldwert gesetzt wird.

Die Sprachen haben wir wie folgt definiert: „D“ für Deutsch, „E“ für Englisch sowie „HD“ für den Händlerbereich in Deutsch und „HE“ für Händlerbereich in Englisch definiert. Dadurch ließ sich die Navigation über diesen Parameter wesentlich vereinfachen. Per hinterlegter Taste nun wird der aktuelle Layoutname in Englisch oder Deutsch als Scriptparameter an das Navigations-Script übergeben und über „Gehe zu Layout – Layoutname durch Berechnung“ springt unser Shop in das gewünschte Layout.

Die Herausforderung – der Händlerbereich

Nun standen wir vor der eigentlichen Herausforderung: der Händlerbereich mit Bestellfunktion. Neben einer Kunden- und Bestelltabelle wurden auch Nummernkreise benötigt. Um später verschiedene Auswertungen erstellen zu können, haben wir ein neues Feld **SessionID** in der System-Tabelle als fortlaufende Nummer erstellt. Dadurch konnten wir ohne großen Aufwand eine kleine Zusatzfunktion einbauen (siehe dazu „Bestellung senden“).

Als zweiter Nummernkreis ist die Bestellnummer zu definieren, die jedoch erst beim Anklicken eines Bestätigungsbuttons generiert werden soll. Außerdem finden wir es extrem lästig, fortlaufende Belegnummern (was eine Bestellnummer auch ist) bei einem Reset des Systems per Hand oder Script zu pflegen. Daher haben wir uns auch hier für eine unkonventionelle Lösung entschieden.

Wir haben eine Formel erstellt, die die letzte vergebene Nummer ausgibt – nun muss der Wert im „Bestell-Script“ nur noch um den Wert 1 erhöht werden. Die **SessionID** ist somit „fast“ unendlich fortlaufend und die Bestellnummer generiert sich selbst – bei keiner vorhandenen Bestellung muss natürlich die Bestellnummer „1“ gesetzt werden.

Die Tabelle ***Kunden*** erhielt nun noch ein kleines, aber feines Login-System, das im Login-Screen Kundenummer und Passwort vergleicht und den Zutritt entweder gestattet oder verweigert. Hierfür generierten wir in der Haupttabelle das globale Feld gMessage, um Statusmeldungen – z. B. bzgl. eines falschen Passwortes – mehrsprachig einfügen zu können.

Für den Auswahlprozess des Artikels dient uns die Tabelle ***Order***. Wenn der Kunde bzw. User bei einem Artikel den Button „ARTIKEL BESTELLEN“

Artikel bestellen, wie in jedem anderen Shop ...

anklickt, wird ein Script ausgelöst, das über Variablen die Artikelnummer, den Artikeltext sowie die Anzahl und die Farbe des Artikels – in der jeweiligen Sprache bzw. Maßeinheit – übergibt und unter Verwendung des primären Schlüssels der SessionID (**$$SessionID**) einen neuen Datensatz in der Tabelle ***Order*** erstellt. Wir erinnern uns, das die Artikeldaten aus einer separaten Datei kommen.

Der Shop hat nach erfolgreichem Login schon auf die Händler-Layouts umgeschaltet und das zweite Menü („Menue_II“), das dieselben Felder und Eigenschaften wie „Menue_I“ enthält, wird angezeigt. Neben „Kontaktformular“, „Impressum“, „Katalog bestellen“, „Unsere AGBs“, „Passwort ändern“ und den Kategorien „Neuheiten-“ und „Outlet-Auswahl“, die nur Händlern zur Verfügung stehen, gehen wir nun an den Warenkorb.

Bestellung senden

Sofern Artikel in den Warenkorb gelegt wurden, erscheinen diese nun via Portal noch einmal aufgelistet nach Nummer, Text, Staffelpreise, Menge, Farbe und Preis. Hier kann der Kunde seine Eingaben prüfen und ändern. Der Mindermengen-Zuschlag wird automatisch kalkuliert und ausgegeben. Nach Auswahl der Zahlungsart – hier Nachnahme, Bankeinzug bzw. hinterlegte Zahlungsvereinbarung und eventuell ein kleiner Text – kann auch schon der Button „BESTELLUNG SENDEN“ betätigt werden. Falls Sie die Auswahl der Zahlungsart vergessen haben, wird der Shop Sie durch eine Nachricht (in **gMessage**) daran erinnern! Der Kunde sieht seinen Warenkorb nochmals in Form eines Bestellscheins und die Frage „Wollen Sie Ihre Bestellung so absenden?“ erscheint. Durch Drücken auf „Ja“ kommen Sie in die Druckansicht und per Schleife wird die neue Bestellnummer generiert. Drücken Sie auf „Nein“, kommen Sie zurück zum Warenkorb.

Durch die **SessionID** lässt sich eine Historie bilden, die z. B. dem Händler beim nächsten Login die nicht abgeschlossene Bestellung anzeigt und ihn eventuell dazu animiert, neben einer neuen Bestellung die alte gleich mit auszuführen. Sicherlich eine sinnvolle Funktion, die auch verkaufsfördernd wirken kann.

Nun kamen wir zur nächsten Hürde auf dem Weg zu einem vollwertigen Online-Shop: der E-Mail-Versand. Mit FileMaker 10 mag das problemlos gehen, zum Entwicklungszeitpunkt gab es weder FileMaker 10 noch diese Funktion. Die Frage stellte sich auch hier, ob nicht der bessere Lösungsansatz eine PHP-basierte Version ist. Leider mussten wir davon Abstand nehmen, da der Zeitrahmen hierfür einfach nicht genug Spielraum bot.

Unser besonderer Dank gilt Herrn Othmar Strässle1, der hierbei den entscheidenden Tipp gab. Beim FileMaker Server Advanced besteht die Möglichkeit, einen FileMaker Pro parallel laufen zu lassen. Somit ist es möglich, via Intervall ein oder mehrere Scripts durchlaufen zu lassen. Notwendig war lediglich ein Mail-Plugin für das Senden von Mails und eine neue Datei, genannt ***Robot.fp7***.

/RRS7LPHU

```
◆ ◆◆
   Option:   Ein
◆ ❸
   Option:   Ein
◆
   Text:     Custom MAIL Scripts
```

```
◆
   Name:     "Lehmann_Order"
◆
   Name:     "Lehmann_Kontakt"
◆
   Name:     "Lehmann_Registration_E"
◆
   Name:     "Lehmann_Registration_H"
◆ ❹
   Layout:   "Messages List" (Messages)
◆
   Text:     Regular Scripts
◆
   Name:     "Get Mail"
◆
   Name:     "Send Mail"
◆
   Text:     Custom Scripts
◆ ❹
   Layout:   "System" (System)
◆ ❸
   Feld:     System::Counter_LehmannSystem
   Formel:   System::Counter_LehmannSystem + 1
◆
◆                                          ▼
   Formel:   System::Counter_LehmannSystem  80
◆
◆
   Name:     $Start
   Wert:     Hole (HostZeitstempel)
◆
   Name:     "Lehmann_KHK_Import"
◆
◆
   Name:     "Lehmann_PDF, Generate"
◆
◆
   Name:     "Lehmann_UploadFiles"
◆
◆ ❹
   Layout:   "System" (System)
◆ ❸
   Name: System::Counter_LehmannSystem
   Formel:   0
◆
   Name:     $End
   Wert:     Hole (HostZeitstempel)
◆ ❸
   Name:     System::Counter_LehmannSystem Time
   Formel:   $End - $Start
◆ ❶◆
   Option:   Dateneingabeüberprüfung unterdrücken
   Option:   Ohne Dialogfeld
◆
◆ ❷                                        ▲
◆
◆ ❹
   Layout:   "Logfile" (Logfile)
◆ ❶◆
◆ ❹
   Layout:   "Messages List" (Messages)
◆ ▼●
◆
   Text:     Auto Server Restart
◆                                          ▼
   Formel:   Hole (SystemUhrzeit) "03:55:00"
             "04:05:00"
◆ ❷
   Event:    Dokument/Anwendung öffnen
   Scripttext: "shutdown /r /f"
◆ ❷                                        ▲
```

Das Interval-Script regelt den Mailversand, sowie die PDF-Generierung

Nun war es möglich, die verschiedensten Arten von E-Mails zu versenden, z. B. bei einer neuen Bestellung, einer neuen Registrierung usw. Mit dieser Variante war es auch möglich, das nächste anstehende Feature des Kunden zu realisieren: die Generierung des PDF-Katalogs.

Ein PDF-Katalog, der sich selbst aktualisiert

Da in der Vergangenheit die Generierung eines PDF-Kataloges immer per Hand angestoßen und dieser auch per Hand versendet wurde, lag es nun nahe, die PDFs über den Webshop zum Download anzubieten und in ein Medienfeld zu packen. Die bereits vorhandene Schleife könnte dies ebenfalls managen, wenn nicht ab und an der Server unter Windows Server 2003 einfrieren würde. Auch hier wusste Othmar Strässle Rat und empfahl, irgendwann nachts einen Restart durchzuführen. Gesagt – getan. In die Schleife wurde „Event senden“ eingebaut, fertig. Nächstes Problem bitte ...

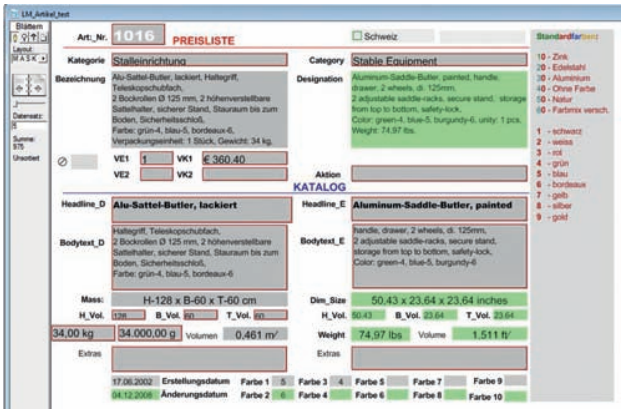
```
◆                                                    ▼
   Formel:   Hole (SystemUhrzeit) "03:55:00" "04:05:00"
◆ ❷
   Event:    Dokument/Anwendung öffnen
   Scripttext:  "shutdown /r /f"
◆ ❷                                                  ▲
```

Per „Event senden“ wird der Server gegen 4 Uhr morgens neu gestartet.

Der Kunde benötigt je ein einzelnes PDF für den Artikel, die Artikelgruppe und das Sortiment. Dabei macht es überhaupt keinen Sinn, die PDFs permanent generieren zu lassen. An einem einzelnen Artikel werden nicht jeden Tag werden Änderungen gemacht, d. h. es müssen auch nicht immer alle Artikel per PDF aktualisiert werden. Eine Art Indikator musste her – eine Funktion, die erkennt, ob ein Artikel verändert wurde oder nicht. Der „Änderungszeitstempel“ konnte hier nicht helfen. Leider hatten wir schon des Öfteren das Problem, dass ein User-Rechner nicht über die korrekte Zeit/das korrekte Datum verfügte und somit keinen verlässlichen Wert lieferte. Uns fiel ein, dass jeder Artikel über eine Liste oder per Suche in der ***LM_Artikel.fp7*** ausgewählt wird. Daher haben wir unter dieses Script einfach per „Feldwert setzen“ in das Feld **Aktualisierung** eine „1“ gesetzt. Die Schleife konnte nun die geänderten Artikel über den Wert „1“ finden und die neuen PDFs über Artikel, Artikelgruppe und Sortiment im Fünf-Minuten-Takt generieren. Und das wesentlich angenehmer für den Server ...

Nach einer Testphase stellten wir fest, dass das Sortiment-PDF mit den jeweiligen Bildern, Spezifikationen und Texten eine Größe von rund 35 MB hatte und das ist einfach zu groß! Nach kurzer Überlegung fiel uns das kostenlose **Moo-Plugin**[3] ein. Mit der Zip-Funktion war es dann ein Leichtes, zwischen der PDF-Generierung und dem Einfügen in ein Medienfeld eine „Pack-Aktion“ einzufügen. Nun ist das Sortiment-PDF zwar immer noch rund 27 MB groß, aber das ist immerhin besser als 35 MB.

Der „alte“ Katalog wird erwachsen



Vorher: Die „alte“ Artikel-Datenbank wurde zum Warenwirtschaftssystem umgebaut.

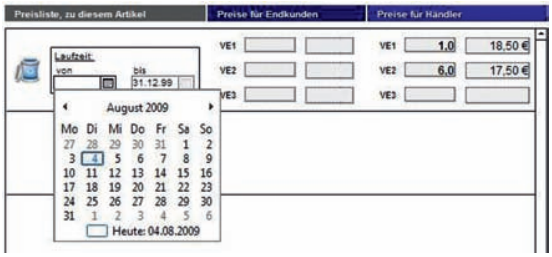
Nachher: Die Grundstrukturen bleiben erhalten.

Nun waren wir soweit, den alten Katalog umzubauen. Die Datei sollte die klassische Artikelpflege und die Einstellungen erlauben. Es sollten aber auch Auswertungen (über Artikel und Bestellungen) und Aktionen für Sonderpreise und natürlich Abbildungen eingepflegt werden können. Und das am besten gleich so, dass diese Aktionen nur einmal angelegt werden müssen und dann „von selbst“ wieder zurückgesetzt werden.

[3] *www.mooplug.com*

Preislisten verwalten, Bilder uploaden

Zwischenzeitlich stand fest, dass früher oder später auch Endkunden beliefert werden sollen, was wir in der Programmierung gleich berücksichtigen wollten. Über eine neue Tabelle ***Artikel_Preisliste*** und eine Beziehung sollte dieser Aspekt umgesetzt werden. Hierzu erstellten wir lediglich in der Haupttabelle der IWP-Datei ein globales Datumsfeld, das beim Betreten der Seite aktualisiert wird. Mit den Feldern **Laufzeitvon, Laufzeitbis** und der **Artikelnummer** konnte nun eine Beziehung erstellt werden, die immer den aktuellen Preis bzw. das aktuelle (Aktions)-Bild liefert. So ist es für den Kunden auch möglich, ein Standardbild und einen Preis für den Zeitraum x bis y anzulegen. Nach der z. B. zweiwöchigen Sonderaktion stellen sich die Aktionswerte durch Erreichen des Datums automatisch wieder auf die Standard-Einstellungen zurück. Der Pflegeaufwand für den Kunden ist somit minimal, was ihn besonders gefreut hat.

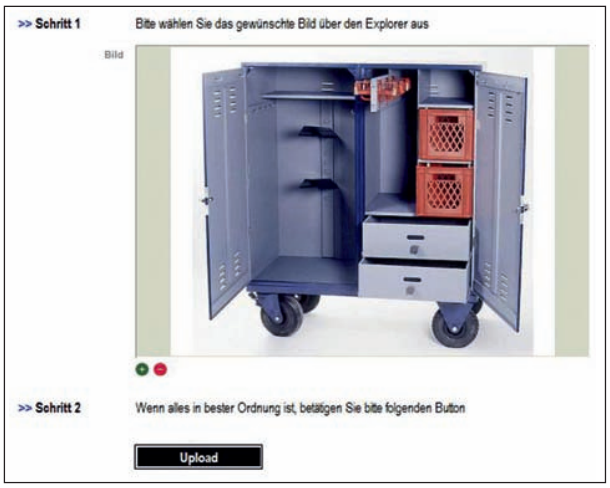


Preise und Bilder werden spielend einfach verwaltet: Laufzeitdatum von/bis eingeben, Parameter setzen, fertig.

Das Upload-Problem war etwas kniffliger: Ein zusätzlicher Wunsch des Kunden war es, dass pro Artikel nicht wie früher nur ein Bild angezeigt wird, sondern gleich mehrere. Also eine Art Slideshow musste her, die wir über eine Beziehung und ein Portal realisierten. Außerdem ließ es die verfügbare DSL-Leitung beim Kunden nicht zu, größere Bild-Uploads durchzuführen. Da normalerweise Webbilder auf den Webserver gehören, experimentierten wir mit einer etwas anderen Variante.

Das Plugin **Troi-File** besitzt die nette kleine Funktion „TrFile_CreateThumbnail“. Mit dieser Funktion ist es möglich, von einem Medienfeldinhalt eine „Daumennagelgroße Vorschau“ (Thumbnail) zu generieren.

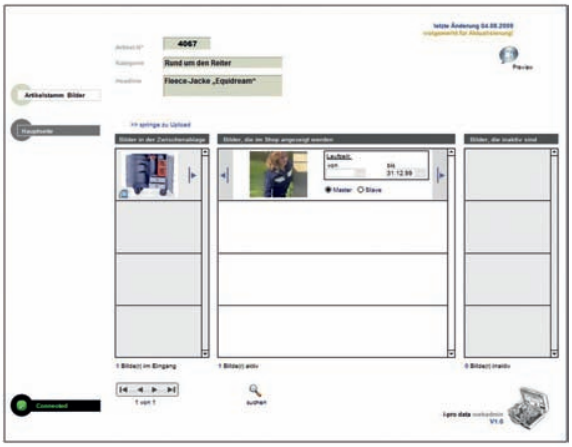
Wir mussten daher nur noch ein globales Medienfeld generieren, in das man das „neue“ Bild laden konnte, das dann per Script einfach als Thumbnail in der neuen Tabelle ***Cache*** abgelegt wird.



Der Upload von neuen Bildern ist kinderleicht.

Wir hatten inzwischen getestet, wie diese „Thumbnails“ im Web und auf dem Ausdruck der PDFs aussehen – der Kunde und wir konnten keine nennenswerten Qualitätsverluste feststellen. Und dabei sind diese Thumbnails nur zwischen 5 und 30 Kb (!!!!) groß. Wir hatten also das Upload-Problem unkonventionell, aber erfolgreich gelöst.

Die Zuordnung der abgelegten Bilder in der Tabelle ***Cache*** lösten wir – für unseren Geschmack – klein, aber fein: Wir erstellten ein neues Layout sowie drei Beziehungen: Die erste Beziehung sollte die Bilder in der ***Cache***-Tabelle unabhängig vom Artikel liefern, die zweite Beziehung sollte die aktiven Bilder anzeigen und die dritte sollte die Möglichkeit schaffen, Bilder inaktiv zu setzen ohne sie dabei zu löschen. Diese sollten ebenfalls angezeigt werden können.



Nach dem Artikelaufruf können die Bilder komfortabel verwaltet werden.

Also integrierten wir in das Layout drei Portale, die die neuen Bilder, die aktiven Bilder und die inaktiven Bilder anzeigen. Über „Pfeil-

Buttons“ können die einzelnen Bilder zwischen den Portalen hin- und hergeschoben werden.

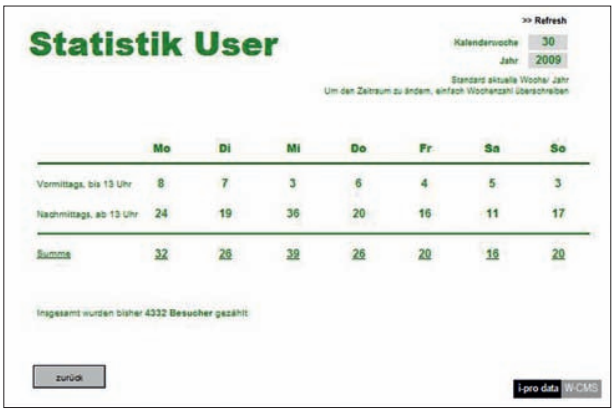
Die Laufzeitfunktion aus der Preisliste durfte natürlich auch hier nicht fehlen. Hinzu kam noch ein Feld **Type**, welches die Identifikation des Hauptbildes (Master & Slave-Prinzip) übernehmen sollte, damit immer mindestens ein Hauptbild für Listen etc. definiert ist. Weiterhin legten wir das Feld **Status** an, das zur Identifikation von aktiven oder inaktiven Bilder diente.

Mittlerweile war es auch sinnvoll (gerade durch die Änderungen an Preisliste und Bildern) den Artikelstamm in die IWP-Datei zu übertragen. So hatten wir nun zur Artikelverwaltung eine Tabelle ***Artikel_Stammdaten***, eine ***Artikel_Preisliste*** und eine ***Artikel_Bilder***. Somit entwickelte sich die IWP-Datei zum Datastore und Webshop – die „alte“ Artikeldatenbank wurde über Beziehungen verknüpft und beinhaltet keine Stammdaten mehr. Sie dient als Schaltzentrale für den Shop, über die sich die Datenquellen, die Aktions-Container und die Top-Produkte auf der Startseite etc. verwalten lassen.

Einfach alles kann durch den Kunden administriert werden.

Kundenstamm und Statistik

Natürlich wünschte sich unser Kunde auch eine komfortable Verwaltung seiner Kunden und Webshop-Besucher. Die Standardfelder hierfür waren schnell erstellt und auch für die Statistikfunktion hatten wir recht schnell einen Lösungsweg gefunden. Über die automatisch generierte **SessionID** und mithilfe diverser Beziehungen konnten wir nun auch die Anzahl der Zugriffe mit Datum und Uhrzeit ermitteln und ausgeben.



Eine Besucherstatistik ist auch ohne größere Anstrengung möglich.

Fazit

Nach sechsmonatiger Entwicklung in drei Schritten ist für uns hinreichend bewiesen, dass mit FileMaker auch Webshops erstellt werden können. Nach anfänglichen Kinderkrankheiten funktioniert diese Lösung mittlerweile so gut, dass der Kunde bereits die fünfstellige Umsatzschwelle überschritten hat. Der E-Mail-Versand und die Generierung des PDF-Katalogs funktionieren einwandfrei und die vielen „kleineren“ (hier nicht alle erwähnten) Funktionen erfreuen selbst den Entwickler noch immer.

Dank dieser Lösung konnte der Kunde die Kosten für Aktualisierung, Software-Updates und die manuelle Katalog-Erstellung wesentlich minimieren. Außerdem hat er nun die Möglichkeit, kurzfristige Änderungen schnell und vor allem selbstständig durchzuführen.

Derzeit wird an der Einbindung von Adobe Flash für die Slideshow sowie von Schnittstellen zu Zahlsystemen wie z. B. **PayPal** – unter Verwendung des Web Viewers – gearbeitet. Doch das füllt sicherlich einen weiteren Beitrag im FileMaker Magazin.

Nachtrag

Wir haben im Webshop einen Test-Account eingerichtet: einfach auf der Startseite auf „Händler“ klicken und einloggen. Bestellungen mit diesem Namen sind unter diesem Account natürlich nicht verbindlich!

URL: ***www.equiweb.de***
Name: **FMM**
Passwort: **123456** ♦