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

Link

Einfache Versionsverwaltung ohne Plugin

Mit automatischem Backup und Logbuch

Wenn man programmiert, muss man eine gewisse Ordnung einhalten, um nicht irgendwann aus dem „Versions"-Takt zu kommen. Stellen Sie sich vor, Sie aktualisieren die Scripts in der Version 1.001.0025 einer Datei. Eine Woche später rufen Sie die Datei wieder auf, merken aber nicht, dass Sie aus Versehen die Version 1.001.0023 erwischt haben … Mir ist so etwas jedenfalls schon passiert, deshalb habe ich ein kleines Ordnungssystem geschaffen, das die Ordnung innerhalb der Datei sicherstellt.

Im Grunde ist es ganz einfach und problemlos in jede existierende FileMaker Lösung zu integrieren. Alle notwendigen Informationen und Scripts finden Sie wie gewohnt in der Beispieldatei.

Nachdem ich mehrmals mit den Versionen durcheinander gekommen bin, habe ich mir überlegt, wie ich eine Struktur in meine Dateien bringen kann. Eigentlich ist FileMaker dazu die ideale Plattform – das Programm bringt alles mit, was nötig ist. Um das Rad nicht neu zu erfinden, habe ich mich erst einmal in den einschlägigen Foren etc. umgeschaut. Leider musste ich feststellen, dass es zwar Ansätze für eine Versionsverwaltung gibt, aber keine komplette Lösung. Also habe ich mich selbst daran gemacht, meinen Code zu versionieren, Ordnung in die vielen Dateien zu bringen und nebenbei auch noch einige andere Programmierer-Probleme zu beseitigen.

Mein Plan sah folgendermaßen aus: Die jeweils aktuelle Version sollte nummeriert und bei entsprechenden Änderungen um eins erhöht werden (Jobnummer). Bei größeren Änderungen sollte eine Erhöhung der Versionsnummer und bei wichtigen Änderungen eine Anpassung der Hauptnummer erfolgen. Die Form habe ich auf 0.000.0000 festgelegt. Zudem sollte die Datei mit der „alten" Version im Verzeichnis „Version" abgelegt und die untergeordneten Nummernkreise ab Änderung der Nebennummer sollten wieder auf Null gesetzt werden.

Zusätzlich hatte ich noch ein paar weitere Anforderungen an das System. Zum Beispiel sollte nach jeder Arbeit an der Datei notiert werden können, was gemacht wurde. Erst danach sollte eine Erhöhung der Versions-, Neben- oder Hauptnummer erfolgen. Zudem sollte es möglich sein, Hinweise, Aufträge oder andere Anforderungen für zukünftige Versionen mit aufzunehmen.

Da eine Versionsänderung nur dann erlaubt sein soll, wenn ein Programmierer eingeloggt ist, muss das System erkennen können, ob sich ein User oder ein Admin bzw. Developer anmeldet.

Zu diesem Zweck habe ich drei Anmeldemöglichkeiten im System erstellt:

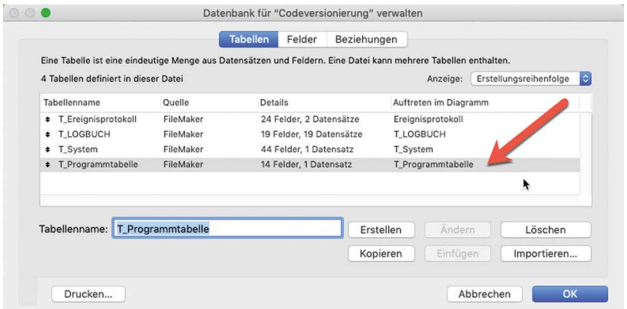
Nr.	Anmelde-name	Passwort	Bemerkung
1	Admin	admin	Anmeldung möglich, Einsichtnahme in Scripts erlaubt, Versionsänderung nicht
2	Developer	dev	Versionsänderungsabfrage beim Schließen des Programmes
3	User	user	Kein Zugang zu Scripts etc.
4	Gast	gast	Kein Zugang zum Programm

Neben der Verwaltung der Versionen gibt es zwei weitere Ziele, die ich mir gestellt habe. Zum einen soll mein Programm automatisch ein Backup von sich selbst erstellen und diese Funktion auch beim Anwender ausführen (nach Vorgabe). Denn auch wenn man Anwender mehrfach auf die Wichtigkeit von Backups hinweist, machen die wenigsten von ihnen eine regelmäßige Sicherung ihrer Datenbank. Daher möchte ich ohne viel Aufwand einstellen können, dass nach jedem x-ten Programmaufruf beim Schließen der Datei ein Backup erstellt und unter einem vom Datum abhängigen Namen abgespeichert wird.

Zum anderen hätte ich gern ein Ereignisprotokoll, in dem dokumentiert wird, welcher User wann welche Änderung an frei definierbaren Daten durchgeführt hat (DSGVO beachten!). Eventuell soll auch ein automatisches Zurücksetzen der Änderungen ermöglicht werden.

Nachdem die Ziele definiert sind, beginnen wir nun mit der Programmierung. Mindestvoraussetzung für die Erstellung der notwendigen Scripts ist FileMaker 16. Eine „abgespeckte" Version, die ab FileMaker 12 arbeitet, können Sie per E-Mail unter *info@fmqu.de* abfordern.

Für die Codeversionierung benötigen wir drei Tabellen sowie zusätzlich mindestens eine Tabelle für die Daten Ihres Programmes. In der Tabelle ***T_System***, die nur aus einem Datensatz besteht, werden alle programmtypischen Daten erfasst, zum Beispiel der Name des Programmes, der Autor, die Haupt-, Versions- und Jobnummer usw. Die Tabelle ***T_Ereignisprotokoll*** enthält alle Daten für das Ereignisprotokoll, die Daten des Logbuches sind in der Tabelle ***T_LOGBUCH*** zu finden.



Der Aufbau dieser Tabellen ist mehrere A4-Seiten lang, Sie können die Vorlagen bitte direkt aus der Beispieldatei entnehmen.

Als vierte Tabelle wird die eigentliche Programmtabelle entweder neu angelegt oder Sie benutzen eine bereits bestehende FileMaker Datei. Dann müssen Sie allerdings alle Scripts und Tabellen dieser Datei in die bestehende kopieren und anpassen. Wichtig in dieser Tabelle ist nur das Feld **Verbinder**, mit dem die Verbindung zu den anderen Tabellen hergestellt wird.

Schauen wir uns nun die Scripts ein wenig genauer an. Zum Starten der Codeversionierung benötigen Sie ein Script, welches die zusätzlichen Aufgaben bei Start der Datenbank einbindet.

`Autoopen`

```
# ------------------------------------------------
#  ©  2013 by Burkhard Quenzler, BeQuSoft
#  Autoopen
#  Script-Parameter: keine
#  Allgemeines Startscript
#  Version: 1.01
# ------------------------------------------------
Gehe zu Layout
  [ "Codeversionierung" (T_Programmtabelle) ;
  Animation: Ohne ]
Fenstertitel setzen
  [ Aktuelles Fenster ; Neuer Titel:
  T_System::Programmnamekurz &
  " - © 2016/18 by "& T_System::Programmautor &
  " - Version: " & T_System::Hauptnummer & "."  &
  T_System::Versionsnummer & " - Lizenznehmer: " &
  T_System::Lizenzuser & " - " & T_System::Firma ]
Wenn
  [ Hole(Kontoname) = "Developer" ]
Gehe zu Layout
  [ "T_LOGBUCH" (T_LOGBUCH) ; Animation: Ohne ]
Gehe zu Datens./Abfrage/Seite
  [ Erste(r) ]
Variable setzen
  [ $offen ; Wert: 0 ]
Variable setzen
  [ $i ; Wert: 0 ]
Variable setzen
  [ $$Abschluss ; Wert: 0 ]
Variable setzen
  [ $$Versionsnummeraktuell ; Wert: T_System::Hauptnummer &
  "."& T_System::Versionsnummer &"."& T_System::Jobnummer ]
Schleife (Anfang)
Wenn
  [ T_LOGBUCH::erledigt = "" ]
Variable setzen
  [ $offen ; Wert: $offen+1 ]
Variable setzen
  [ $i ; Wert: $i+1 ]
Ende (wenn)
Gehe zu Datens./Abfrage/Seite
  [ Nächste(r) ; Nach letztem beenden: Ein ]
Schleife (Ende)
Wenn
  [ $offen ≠ 0 ]
Eigenes Dialogfeld anzeigen
```

```
  [ $offen & " Offene Aufträge" ; "Es sind noch " & $offen &
  " offene Aufträge vorhanden. Möchten Sie diese jetzt
  einsehen? ¶ "  & "… ]
◆ Ende (wenn)
◆ Feldwert setzen
  [ T_System::Wunsch ; "" ]
◆ Feldwert setzen
  [ T_System::Abbruch ; "" ]
◆ Sonst
◆ AnwenderAbbruchZulassen setzen
  [ Aus ]
◆ Fehleraufzeichnung setzen
  [ Ein ]
◆ Werkzeugleisten ein-/ausblenden
  [ Ausblenden ]
◆ Formatierungsleiste zulassen
  [ Aus ]
◆ Menüset installieren
  [ "[Dateistandard]" ]
◆ Ende (wenn)
◆ Variable setzen
  [ $$System ; Wert: Hole(SystemPlattform) ]
◆ Feldwert setzen
  [ T_System::Zaehler ; T_System::Zaehler +1 ]
◆ Feldwert setzen
  [ T_System::Ordnerpfad ; Hole(DateiPfad) ]
◆ Wenn
  [ Hole(LetzteMeldungswahl)=2 ]
◆ Wenn
  [ $i > 509 ]
◆ Eigenes Dialogfeld anzeigen
  [ "Aufträge" ;
  "Es waren weit über 500 Aufträge in der Datei vermerkt.
  Bitte löschen Sie Sie die Aufträge, die Sie…" ]
◆ Ende (wenn)
◆ Aktuelles Script verlassen
  [ Textergebnis: ]
◆ Ende (wenn)
◆ Gehe zu Layout
  [ "Codeversionierung" (T_Programmtabelle) ;
  Animation: Ohne ]
◆ Script ausführen
  [ Angegeben: Aus Liste ; "Startfenster aufrufen" ;
  Parameter: ]
◆ #
```

„Autoopen“ wird als Startscript eingebunden und sorgt dafür, dass der Titel des Programmes mit der Versionsnummer gesetzt wird. Anschließend prüft es, ob offene Vorschläge, Hinweise oder Aufträge vorliegen. Ist das der Fall, wird das Layout der Tabelle ***T_LOGBUCH*** aufgerufen – vorausgesetzt, Sie sind als Admin oder Developer angemeldet.

Das Script „Autoclose“ wiederum übernimmt genau diese Funktion beim Schließen des Programmes. Es fügt notwendige Arbeitsschritte in den Schließvorgang der Datenbank ein.

`Autoclose`

```
◆ #  ©  2013 by Burkhard Quenzler, BeQuSoft
◆ #  Script-Parameter: keine
◆ # Allgemeines Startscript ohne Parameter
◆ #  Version: 1.01
◆ AnwenderAbbruchZulassen setzen
  [ Aus ]
◆ Fehleraufzeichnung setzen
  [ Ein ]
◆ Wenn
  [ $$Abschluss <> 1 ]
◆ Script ausführen
  [ Angegeben: Aus Liste ; "Backup nach Zähler" ;
  Parameter: ]
◆ Wenn
  [ Hole(Kontoname) = "Developer" ]
◆ Script ausführen
  [ Angegeben: Aus Liste ; "Versionsverwaltung" ;
  Parameter: ]
◆ Ende (wenn)
◆ Werkzeugleisten ein-/ausblenden
  [ Ausblenden ]
◆ Ende (wenn)
◆ Programm beenden
```

Dieses Script prüft, ob sich der Entwickler abmeldet oder ein anderer User. Im ersteren Fall wird die Versionsabfrage durchgeführt, indem das Script „Versionsverwaltung“ aufgerufen wird.

Wenn Sie alle diese Funktionen mit Versionierung, Logbuch und Ereigniskontrolle nutzen möchten, müssen Sie bitte ebenfalls alle benötigten 24 Schritte wie oben beschrieben in Ihre Datei oder in die neue FileMaker Datei kopieren.

Alle Vorarbeiten sind damit erledigt, nun gilt es nur noch, das Ganze zu einer FileMaker Datei zusammenzubauen. Dabei können Sie die Scripts ganz bequem aus der Beispieldatei kopieren, die wie gewohnt zum Download bereitsteht.

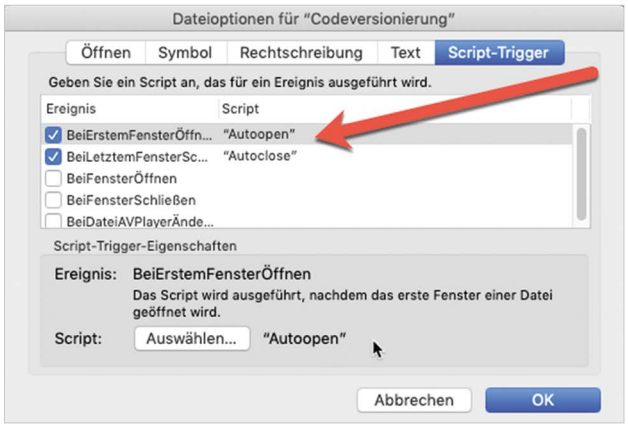
Ich habe diese Vorgehensweise gewählt, weil es irrelevant ist, ob Sie eine neue FileMaker Datei erstellen oder eine bereits vorhandene Datei mit der Codeversionierung versehen möchten.

Importieren Sie zunächst die Tabellen und anschließend die Scripts in Ihre vorhandene oder neue Datei. Bitte halten Sie diese Reihenfolge unbedingt ein, da sonst die Verknüpfungen zur Tabelle verlorengehen.

Ist das erledigt, öffnen Sie bitte unter dem Menüpunkt „Ablage“ (Mac) bzw. „Datei“ (Windows) die Dateioptionen in Ihrer Datenbank.

Aktivieren Sie die Option „Umschalten zu Layout“ und geben Sie dort Ihr Hauptlayout ein – in unserem Beispiel das Layout „Codeversionierung“.

Alle weiteren Einstellungen bleiben hiervon unberührt.



Nun veranlassen Sie mit dieser Einstellung den ordnungsgemäßen Start der Scripts.

Klicken Sie auf den Reiter „Script-Trigger“ und weisen Sie dem Trigger „BeimErstenFensterÖffnen“ das Script „Autoopen“ und dem Trigger „BeimLetztenFensterSchließen“ das Script „Autoclose“ zu. Zum Schluss müssen Sie die Scripts an Ihre Bezeichnungen und Namen anpassen und schon haben Sie eine Datei mit eigener Versionsverwaltung!

Schauen Sie sich an, wie unterschiedlich das Programm auf die jeweiligen User reagiert. Nur als Entwickler (Developer) können Sie die Versionsverwaltung aufrufen. Prüfen Sie, ob im Dateiverzeichnis die Ordner „Backup“ oder „Version“ entstanden sind und die jeweilige Datei ordnungsgemäß gespeichert wurde.

Ich hoffe, Ihnen eine Anregung zur einfachen Versionsverwaltung an die Hand gegeben zu haben. Mit Sicherheit werde ich diese Datei auch in Zukunft weiter pflegen und mit zusätzlichen Details versehen. Falls Sie Hinweise oder Anregungen für mich haben, würde ich mich sehr darüber freuen. Für Rückfragen bin ich gern unter meiner Adresse *info@fmqu.de* zu erreichen. ■