FileMaker Erweiterung

7.0

Foreign affairs

Verschlüsselte Beziehungen

Claus M. Niemeier, Hamburg
Dr. Christopher Busch, Hamburg
mail@christopherbusch.de

Seit kurzem gibt es das „Troi File-Plugin" für die neue Plugin-API von FileMaker 7. Und auch die übrigen Troi-Plugins, allen voran das neue „Troi Encryptor-Plugin" (vormals „Troi Coding-Plugin"), sind nicht nur für FileMaker 7 geeignet, sondern explizit für die neue Plugin-Schnittstelle konzipiert. Grund genug, hier die beiden genannten Plugins im Allgemeinen einmal vorzustellen und an einem speziellen Beispiel aus der Programmierpraxis zu erläutern.

Hinweis: Für dieses Beispiel benötigen Sie neben FileMaker Pro 7 die folgenden Plugins: ***Troi Encryptor*** *(mind. Vers 2.0),* ***Troi File*** *(min. Vers. 3.0),* ***Troi Text*** *(mind. Vers. 2.7) sowie* ***Troi Dialog*** *(mind. Vers. 3.5). Sie können diese Plugins zu Testzwecken laden [1] und bei Gefallen entsprechende Lizenzen über den K&K-Verlag käuflich erwerben.*

Da wäre zunächst das **Troi File-Plugin** in der aktuellen Version 3.0.3 zu erwähnen: Mit diesem Plugin lassen sich die üblichen Dateioperationen aus FileMaker heraus ausführen, die man ansonsten mit dem Finder (Macintosh) oder dem Explorer (Windows) erledigen muss, wie z.B. Dateien verschieben oder auswählen. Neu hinzugekommen ist mit der letzten Version die Möglichkeit in einem Ordner-Auswahldialog (Funktion „TrFile_SelectFolderDialog") unter Windows per Button einen neuen Ordner anzulegen sowie die Funktion „TrFile_Exists", mit der man die Existenz einer Datei prüfen kann. Bestand in früheren Versionen die einzige Möglichkeit einer Existenzüberprüfung darin, einen beliebigen Wert einer Datei (z.B. die Größe) abzufragen um deren Existenz prüfen, so liefert diese Funktion nun einfach eine „0" oder „1" als Ergebnis zurück.

Als weiteres Plugin verwenden wir in unserem nachfolgenden Beispiel das **Troi Encryptor-Plugin** in der Version 2.0b2, welches uns für diesen Artikel freundlicherweise von ***Troi Automatisering*** vorab zur Verfügung gestellt wurde, nachdem wir eine Inkompatibilität der vorangehenden Betaversion mit dem **File-Plugin** festgestellt hatten. Bis zum Erscheinen dieser Ausgabe dürfte das Problem aber insofern auch für alle übrigen Benutzer behoben sein, da die Vorversion ohnehin nur bis Ende März funktionsfähig war. Wir benutzen das Plugin in unserem Beispiel für die Verschlüsselung von Feldern; es erweitert FileMaker aber auch um Funktionen zur Prüfsummengenerierung („Hashfunktionen") und zur Textkonvertierung („Safe-ASCII").

Kommen wir nun zu einem praktischen Beispiel für die Anwendung der beiden Plugins: der Verschlüsselung von Beziehungen, genauer gesagt der Verschlüsselung eines Beziehungsschlüssels.

In vielen Branchen z.B. der Medizin, gibt es häufig das Problem, dass einerseits sensible Daten vor dem Missbrauch durch Unberechtigte geschützt werden müssen, gleichzeitig kann es aber auch sehr nützlich sein, Daten statistisch auswerten zu können: Während etwa in der Medizin der betreuende Arzt wissen muss, an welchen Krankheiten sein Patient leidet oder bisher gelitten hat, ist es für den Doktoranden, der eine statistische Auswertung über Kopfschmerzpatienten mittleren Alters macht, unerheblich, ob ein Herr Meier oder Schulz davon betroffen ist.

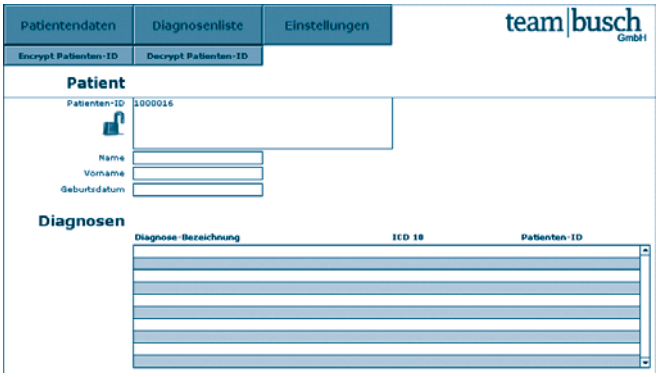

Abb. 1 Beispiel Patientenmaske mit zugehörigen Diagnosen

Vom Datenmodell her ist der Fall klar: Patienten und Diagnosedaten werden in jeweils eigenen Tabellen abgelegt, die über eine eindeutige Beziehung (1:n) – in unserem Fall die eindeutige Patienten-ID – miteinander verbunden sind. Darin unterscheidet sich unser Beispiel nicht von herkömmlichen Datenbankanwendungen ähnlichen Zweckes. Abb. 1 zeigt diese Daten.

Neu hingegen ist die Möglichkeit, diese Patienten-ID auf der Seite der Patientenstammdaten mittels der Funktion „Encr_EncryptRijndaelAES" zu verschlüsseln. Diese Funktion verschlüsselt ein beliebiges Textfeld, indem es aus einem selbstzuwählenden Schlüsseltext einen 16 Byte großen Schlüssel zuzüglich Zufallsdaten generiert, mit dem unsere Patienten-ID dann verschlüsselt wird. Sowohl die Patientenstammdaten als auch die Diagnosedaten bleiben unangetastet; lediglich der Bezug zueinander erfordert den richtigen Schlüssel, um die herkömmliche Patienten-ID wieder herzustellen.

Aufmerksame Leser werden sich nun fragen, was das ganze mit dem **Troi File-Plugin** zu tun hat. Wer schon zahlreiche Passwörter, Scheckkarten-PINs usw. im Kopf behalten muss, weiß einfache Methoden zu schätzen, um Schlüsseldaten vorrätig zu halten. Eine solche Methode ist die Speicherung eines Schlüsseltextes auf einem handelsüblichen USB-Stick.

Für die Aufbewahrung unseres Schlüsseltextes haben wir uns eine einfache Datei namens ***EnDecryptionKey.txt*** ausgedacht. Wir legen dazu einfach diese Textdatei auf die oberste Ebene eines beliebigen USB-Sticks. Beim Anschließen dieses „USB-Schlüssels" lässt sich nun diese Datei per Script suchen und der Inhalt einlesen:

```
Suche USB-Stick mit KeyFile
◆ Fehleraufzeichnung setzen
    Option: Ein
◆ # Variablen initialisieren
◆ Feldwert setzen
    Feld: Settings::var_Pfad_zum_KeyFile
    Formel: ""
◆ Feldwert setzen
    Feld: Settings::var_Zaehler
    Formel: 1
◆ # Laufwerksliste einlesen
◆ Feldwert setzen
    Feld: Settings::var_Laufwerke
    Formel: TrFile_ListDisks ("") & "¶"
◆ # Schleife über Laufwerke: Test ob Keyfile existiert
```

◆ Schleife (Anfang) ▼

◆ Feldwert setzen
Feld: Settings::var_Pfad_zum_KeyFile
Formel: TrText_GetLine (""; Settings::var_Zaehler; Settings::var_Laufwerke) & Settings::PfadTrenner_Fng & Settings::var_Name_KeyFile

◆ Feldwert setzen
Feld: Settings::var_Zaehler
Formel: Settings::var_Zaehler + 1

◆ Verlasse Schleife wenn ◆
Formel: TrFile_Exists (""; Settings::var_Pfad_zum_KeyFile)=1 ODER Settings::var_Zaehler > MusterAnzahl (Settings::var_Laufwerke; "¶")+1

◆ Schleife (Schluss) ▲

◆ # *Der Einfachheit halber nochmal testen, ob Keyfile existiert oder nur die Laufwerksliste zu Ende ist.*

◆ Wenn
Formel: TrFile_Exists(""; Settings::var_Pfad_zum_KeyFile) = 0 ODER ZeichenLinks(Settings::EnDecryptionKey_Fng; 1)= "$" ODER IstLeer (Settings::EnDecryptionKey_Fng)

◆ Feldwert setzen
Feld: Settings::var_Pfad_zum_KeyFile
Formel: ""

◆ Eigenes Dialogfeld anzeigen
Titel: "KeyFile nicht gefunden!"
Mitteilung: Die Datei <EnDecryptionKey.txt> wurde bei keinem angeschlossenen Volujme auf oberster Ebene gefunden.";
Taste: Schade

◆ Ende (wenn)

Dabei werden zunächst mit der Funktion „TrFile_ListDisks“ alle vorhandenen Laufwerke in eine Variable eingelesen. Anschließend werden in einer Schleife alle Laufwerke auf die Existenz der angegebenen Datei geprüft. Der Einfachheit halber kommt zur Auswahl eines einzelnen Laufwerks die Funktion „TrText_GetLine“ des **Troi Text-Plugins** zur Anwendung; man kann aber auch eine Formel schreiben, um eine einzelne Zeile aus der Laufwerksliste auszuschneiden.

Alternativ ließe sich auch mit der Funktion „TrFile_SelectFileDialog“ des **Troi File-Plugins** eine Schlüsseldatei direkt auswählen; aus thematischen Gründen verzichten wir aber an dieser Stelle auf die Details zu dieser Funktion.

Mit der Funktion „TrFile_GetContents“ des **Troi File-Plugins**, welche im Formelfeld **EnDecryptionKey_Fng** in der Tabelle „Settings“ als nichtgespeicherte Formel steht, wird dem Formelfeld der aktuelle Inhalt der Schlüsseldatei – genauer gesagt in unserem Beispiel die ersten 64 Zeichen – zugeordnet:

```
EnDecryptionKey_Fng [Text] =
TrFile_GetContents
("-unused";
var_Pfad_zum_KeyFile; 0;
64)
```

An der Stelle der Dateiauswahl sei auf einige systembedingte Fallstricke im Umgang mit dem **Troi File-Plugin** hingewiesen:

Zum einen unterscheiden sich die Pfadtrenner bei Windows (Backslash \) vom Macintosh (Doppelpunkt :). Wer häufiger FileMaker-Datenbanken für beide Systeme entwickelt, dem sei eine Platzhalter-Formel empfohlen:

```
PfadTrenner_Fng [Text] =
Falls (
 Abs (
 Hole (System Plattform)
 ) = 1; ":"; "\\"
)
```

Dabei werden unter unter FileMaker 7 bestimmte Sonderzeichen mit einem „\“ (Backslash) maskiert; der doppelte Backslash („\\“) der Formel erzeugt in der Ausgabe auf der Windows-Plattform also einen einfachen Backslash als Pfadtrennzeichen.

Zum anderen sucht die Funktion „TrFile_SelectFileDialog“ standardmäßig nach Textdateien. Wenn man z.B. auf dem Macintosh ein Keyfile verwendet, welches nicht vom Typ „Text“ ist, bekommt man es im Auswahldialog schlichtweg nicht angezeigt, wenn man nicht zuvor mit der Funktion „TrFile_SetDefaultType“ den entsprechenden Typ voreingestellt hat beziehungsweise alle anzeigen lässt.

Hat man diese Klippen der Schlüsselauswahl einmal umschifft, steht dem erfolgreichen Ver- und Entschlüsseln der Patienten-ID mit den Scripts „Encrypt Patient_ID“ bzw. „Decrypt Patient_ID“ nichts mehr im Wege:

Encrypt Patient_ID

◆ Wenn ▼
Formel: Patienten::Encrypted_01_Fng = 1

◆ # *Patienten-ID ist bereits verschlüsselt*

◆ Aktuelles Script verlassen ◆

◆ Ende (wenn) ▲

◆ #

◆ Wenn ▼
Formel: ZeichenLinks (Settings::EnDecryptionKey_Fng;2) = "$$" ODER IstLeer (Settings::EnDecryptionKey_Fng)

◆ # *Es gibt keine verschlüsselte Patienten-ID, z.B. weil die Schlüsseldatei nicht vorhanden ist.*

◆ Eigenes Dialogfeld anzeigen
Titel: Fehler
Mitteilung: Datei mit Schlüssel nicht vorhanden
Taste: OK

◆ Alle Scripts abbrechen

◆ Sonst ◆

◆ # *Patienten-ID verschlüsseln*

◆ Feldwert setzen
Feld: Patienten::ID_Patient
Formel: Encr_EncryptRijndaelAES ("-Unused"; Settings::EnDecryptionKey_Fng; Patienten::ID_Patient)

◆ Ende (wenn) ▲

Decrypt Patient_ID

◆ Wenn ▼
Formel: Patienten::Encrypted_01_Fng = 0

◆ # *Patienten-ID ist bereits entschlüsselt*

◆ Aktuelles Script verlassen ◆

◆ Ende (wenn) ▲

◆ Wenn ▼
Formel: ZeichenLinks (Settings::EnDecryptionKey_Fng; 2) = "$$" ODER IstLeer (Settings::EnDecryptionKey_Fng)

◆ # *Kein Schlüssel vorhanden*

◆ Eigenes Dialogfeld anzeigen
Titel: Fehler
Mitteilung: Datei mit Schlüssel nicht gefunden!
Taste: OK

◆ Alle Scripts abbrechen

◆ Sonst ◆

◆ # *Patienten-ID entschlüsseln*

◆ Feldwert setzen
Feld: Patienten::ID_Patient
Formel: Encr_Code ("-DecryptRijndaelAES"; Settings::EnDecryptionKey_Fng; Patienten::ID_Patient)

◆ Ende (wenn) ▲

Damit nicht eine bereits verschlüsselte ID noch einmal verschlüsselt wird bzw. nicht der Versuch unternommen wird, eine bereits entschlüsselte ID noch einmal zu entschlüsseln um dann im schlimmsten Fall einen Fehlercode in das Patienten-ID-Feld zu schreiben, prüfen die entsprechenden Scripts vorab, ob sie überhaupt benötigt werden oder ob ein Schlüssel überhaupt existiert.

Natürlich lässt dieses Beispiel noch Fragen offen, etwa nach der Sicherheit des USB-Schlüssels als auch der leichten Kopierbarkeit der Schlüsseldatei. Dennoch bietet die Verschlüsselung von Beziehungsfeldern interessante Möglichkeiten, wie dieses Beispiel zeigt. ■

Zu diesem Beitrag sind Beispieldateien als E-Mail verfügbar

[1] http://www.troi.com

Dr. Christopher Busch
(Jg. 1960) studierte Physik, Mathematik, und Medizin und promovierte in molekularer Neurobiologie. Er ist Inhaber der Hamburger Firma Team|busch GmbH, die überwiegend mit der Konzeption und Erstellung von FileMaker-Lösungen befasst ist. Das Impressum des FileMaker Magazins kennt ihn seit der dritten Ausgabe.

***Claus M. Niemeier** (Jg. 1969) studierte Mathematik und Chemie auf Lehramt, arbeitete für eine Hamburger Werbeagentur im EDV-Support und als FileMaker-Entwickler. Seit Juli 2004 ist er Mitglied im team|busch.*