FMK | FileMaker Konferenz

Weniger schlecht programmieren

Fehler vermeiden

Roland Schneider

TAO SOLUTIONS

Über den Sprecher

Roland Schneider

Inhaber von TAO SOLUTIONS

Jahrgang 1969

Erstkontakt: 1996 FileMaker 2.1

Warum "weniger schlecht"?

- Perfekter Code ist eine Illusion
- Ziel: Code, den andere verstehen können
- Ziel: Code, den dein zukünftiges Ich versteht
- Selbstreflexion führt zur kontinuierlichen Verbesserung
- Wartbarkeit > Perfektion

Enter Find Mode

- Script wird pausieren
- on: Wird selten verwendet

Gegenmaßnahme:

„Warnung wenn Pause = On“ mit MBS Plugin einrichten

Variablen Tipp Fehler

- FileMaker warnt nicht, wenn Ihr Euch bei einem Variablennamen vertippt.
- Das Script wird nicht machen, was Ihr erwartet
- Im schlimmsten Fall merkt man es erst zu spät

Gegenmaßnahme:

- MBS Plugin einrichten

Redundanz

- Dopplungen von Code vermeiden
- Änderungen an mehreren Stellen = Fehlerquelle
- Script mit Parametern sinnvoll erweitern (aber nicht zu viel)
- DRY-Prinzip (Don't Repeat Yourself)

Zeitverschiebung beachten

- FileMaker Server hat unterschiedliche Systemzeit
- Clients haben unterschiedliche Systemzeit
- Evtl. Offset beachten

Gegenmaßnahme:

```
If [ Abs ( Get ( CurrentHostTimestamp ) - Get ( CurrentTimestamp ) ) > 300 ]
       Show Custom Dialog [ "Achtung" ; "Systemzeit weicht ab!" ]
       Exit Application
End If
```

Layout-Trigger vermeiden

- Layout-Trigger = versteckte Komplexität
- Schwer zu debuggen
- Können Performance-Killer sein
- Besser: Explizite Script-Aufrufe

Gegenmaßnahme:

- Selten benutzen, wenn dann gut dokumentieren
- Extra Layouts für Scripte verwenden z.B. BLD_Artikel

Quick & Dirty rächt sich

- Kein Sofort-Programmieren im Call oder in Besprechungen
- Lieber Zeit für Prüfung und Fixen ansprechen
- Gewissenhaft vor Schnelligkeit
- Vorsicht: Code nicht ändern, nur weil man ihn heute nicht mehr versteht

Plugin-Abhängigkeiten

- Abhängigkeiten dokumentieren
- Beim Systemstart prüfen, ob das Plugin installiert ist
- $$MBS.installed = „OK“
- Wenn nicht, Plugin installieren
- Bei Fehler, warnen
- Script abbrechen, wenn notwendige Plugin nicht vorhanden sind
- Prüfung auf $$MBS.installed = „OK“

Script-Architektur

- Keine Monsterscripts
- Maximal eine Bildschirmseite pro Script
- Subscripts verwenden
- Wenig Verschaltelungen
- Parameter-Verzweigung vermeiden

Kleiner Tipp

Go to Record/Request/Page [Next ; Exit after last: Off]

Konventionen

- Eine Konvention festlegen und durchziehen
- Welche? Egal - Hauptsache konsistent
- Stringenz: Kein Hüh und Hott
- Aussagekräftige Namen verwenden

```
Set Variable [ $z ; Value: Case( $NurAktuellenDS; 1; Get ( FoundCount )
Loop [ Flush: Always ]
    Set Variable [ $j ; Value: $j + 1 ]
    Set Variable [ $i ; Value: $i + 1 ]
    #     Aktuelle Kriterien einsammeln ($ak_n: Aktuelles Kriterium_l
          ak_1 nimmte die KundenID auf; $ak_2 das Sammelrechnungsflag
    Set Variable [ $ak_1 ; Value: Schildvermietungen::_ID_Kunde ]
    Set Variable [ $ak_2 ; Value: Case( Schildvermietungen::IstSammelr
    #     $ZW: Speichert die zu übergebenden Werte.
```

Kommentare

- Was habe ich mir dabei eigentlich gedacht?!?!
- Heute kommentieren für morgen
- Erst die Kommentare, dann den Code
- URL bei kopierten Lösungen angeben

Fehler erwarten - immer!

- Versucht, robust programmieren
- Get(FoundCount) prüfen
- Get(LastError) auswerten
- Bei Fehler Anwender informieren
- Pause on Error
- Nicht wird schon gutgehen

Kontextunabhängig programmieren

- Keine Annahmen über aktuellen Kontext
- Sondern Kontext gezielt setzen
- Am Ende eines Scripts: Go to Layout [original layout]

Code-Klarheit

- Keine unnötigen Abkürzungen

  Nicht

  ```
  Get(FoundCount)
  ```

  sondern

  ```
  Get(FoundCount) > 0
  ```

- Selbstdokumentierender Code

  Nicht

  ```
  Perform Find [ Restore ]
  ```

  sondern

  ```
  Enter Find Mode [ Pause: Off ]
  Set Field [ Personen::Vorname ; "k" ]
  Perform Find []
  ```

Testen - aber richtig!

- NICHT als Admin testen: da klappt es immer!
- Mit echten Userrechten testen
- Im Team gegenseitig testen
- Mit Kunden-Mitarbeiter testen
- Multiplikator-Effekt nutzen

Das "Heilige Script"

- Gut getestet
- Gut dokumentiert
- Nicht anfassen ohne Review

Effizienz vs. Wartbarkeit

- Gut kopiert ist besser als schlecht selbst gemacht
- https://www.briandunning.com/filemaker-custom-functions/

- ABER: Quelle als Kommentar
- Strenge Auswahl der Quellen
- Verstehen, was man kopiert
- Testen, testen, testen

Tägliche Routinen

- Täglich aufräumen
- Nicht benutze Scripts entfernen
- ….Copy entfernen
- Kommentare aktualisieren
- TODOs abarbeiten
- Code Reviews

Wie werde ich besser?

- Dokumentation Als "Abendlektüre“ studieren
- Stammtische besuchen
- Konferenzen besuchen
- Foren aktiv nutzen
- Fragt eine KI
- Im Team programmieren
- Fehler teilen
- Best Practices diskutieren

- Spaß haben!

Q & R

Vielen Dank für Eurer Interesse!

Welche Fehler kennt ihr?

Roland Schneider
www.tao-solutions.de
rs@tao-solutions.de

Vielen Dank unseren Sponsoren und Konferenz-Partnern

Claris

FileMaker Magazin

24U SOFTWARE

CamelCase
camelcase.com

CAOS
culture agency office software