Structured Variables in FileMaker

Sylvain Parent 2017

Sylvain Parent

Jefo Logistique, Saint-Hyacinthe, Québec, Canada
LinkedIn

On the menu this morning

1. What are structured variables
   list, dictionary and matrix
2. Build structured variable with CLOB
3. Persistent variables
4. Demo of an Encapsulated FileMaker Solution
5. What is an *objectoid* module
6. *Demo of the Ephemeral Matrix*

Structured variables

- FileMaker list (list separated by ¶ or a list of words)
- Array (build with CLOB or similar)
- Dictionary (SFR or similar)
- Matrix or two dimensional array (CLOB or similar)
- A Structured variables who contain any combination of array, dictionary, matrix and even structured variable.

A structured variable is first a
structure of information
then it is a technique.

All structured variables must be serialized but not all the time

$$myVariable [x]

The evil loved because misnamed variable that could become your “best”

According to the FileMaker documentation

- In $$myVariable [x] x is a repetition number
- Everybody (maybe just me) inferred that $$myVariable will give acces to all of the repetitions
  when in fact $$myVariable = $$myVariable [1]
- So we cannot use it as an array
- Can they be used for anything other than naming buttons?

If we change the meaning of X
a few doors open.

$$myVariable[referenceID]

Where referenceID is a number referring to any object, information or structured variable.

Persistent Variables ($$myVariable [referenceID])

- Persistent variable is a private global variable accessible only by its ID via scripts or CFs dedicated to its manipulation.
- A module built on a persistent variable can only use integer as ID.
- A module can emit the ID in this case it will be significant for the module.
- A module can receive ID issued by other in this case it will be significant for the user of the module.
- The Persistent variables looks like an object in object-oriented programming (object*oid*)
- The Persistent variables are not object.

referenceID may refer to

- An object that already has a native ID in FileMaker
  - Layout ID, Script ID, Window ID
- An information created by convention
  - kOnObjectEnter, kOnObjectValidate, kInterfaceLanguage
- An ID build by logic
  - $$buttomName [$$buttomID]
    where $$buttomID = windowID + layoutID + buttonNumber

referenceID may refer to

- Any structured variables
  - List, dictionary, array, matrix
- A cell of an ephemeral matrix
- ….

Objectoid PV

1. Filemaker PV are conceptually similar to real-world objects: they too consist of state and related behaviour.

2. A FileMaker PV stores its state in variables ($$) and exposes its behaviour through custom functions or scripts.

3. CF or scripts operate on a PV's internal state and serve as the primary mechanism for communication to the PV.

4. Hiding internal state and requiring all interaction to be performed through a PV's custom functions or scripts is known as data encapsulation — a fundamental principle of object-oriented programming.

5. We have a "Good" PV if the interface is independent of the implementation
   if $$myVariable [referenceID] can be replaced by any other variable, or element of a list, dictionary, CLOB or anything that may contain information in memory.

Demo

Software Object

1. Software objects are conceptually similar to real-world objects: they too consist of state and related behavior.

2. An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages).

3. Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

4. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming.

5. We have a «Solid» object if the interface is independent of the implementation.

FileMaker Object*oid*

- Software objects are conceptually similar to real-world objects: they too consist of state and related behavior.

- Filemaker module are conceptually similar to real-world objects: they too consist of state and related behavior.

FileMaker Object*oid*

- An object stores its state in fields (variables in some programming languages) and exposes its behaviour through methods (functions in some programming languages).

- A FileMaker module stores its state in table/fields and variables and exposes its behaviour to other module through scripts or customs functions.

FileMaker Object*oid*

- Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

- Custom functions or scripts operate on a module's internal state and serve as the primary mechanism for module-to-module communication.

FileMaker Object*oid*

- Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming.

- Hiding internal state and requiring all interaction to be performed through a module's custom functions or scripts is known as data encapsulation — a fundamental principle of object-oriented programming.

FileMaker Objectoid

- We have a “Good” object if the interface is independent of the implementation.
- We have a “Good” module if the interface is independent of the implementation.

Module as an object

- The interface is independent of the implementation
  if the structure of the module a can be replaced by any other structure or external solution without any change in module B.

- Most of the time there is an information dependency that must be recreated.:- (

An objectoid module must receive or return one or more of these elements:

An objectoid module must receive or return one or more of these elements:

- A word or a sentence

An objectoid module must receive or return one or more of these elements:

- A word or a sentence
- An array (or list)

An objectoid module must receive or return one or more of these elements:

- A word or a sentence
- An array (or list)
- A matrix

An objectoid module must receive or return one or more of these elements:

- A word or a sentence
- An array (or list)
- A matrix
- A string (JSON, XML, CSV, ExecuteSql result, etc.)

An objectoid module must receive or return one or more of these elements:

- A word or a sentence
- An array (or list)
- A matrix
- A string (JSON, XML, CSV, ExecuteSql result, etc.)
- A structured variable

An objectoid module must receive or return one or more of these elements:

- A word or a sentence
- An array (or list)
- A matrix
- A string (JSON, XML, CSV, ExecuteSql result, etc.)
- A structured variable
- All this in the form of serialized chain or a reference in memory.

An objectoid module :

An objectoid module :

- Can receive several parameters. Each of them can be from one time to another different structured variable.

An objectoid module :

- Can receive several parameters. Each of them can be from one time to another different structured variable.
- Can return several parameters. Each of them can be from one time to another different structured variable.

An objectoid module :

- Can receive several parameters. Each of them can be from one time to another different structured variable.
- Can return several parameters. Each of them can be from one time to another different structured variable.
- Must see the use of parameters as communication channels and not as arguments to a calculation.

Structured variables can be stored

Structured variables can be stored

- in a Let () variable,

Structured variables can be stored

- in a Let () variable,
- in a local variable

Structured variables can be stored

- in a Let () variable,
- in a local variable
- in a global variable

Structured variables can be stored

- in a Let () variable,
- in a local variable
- in a global variable
- in a persistent variable PV

Structured variables can be stored

- in a Let () variable,
- in a local variable
- in a global variable
- in a persistent variable PV
- inside the sidereal space where the Get (ScriptParameter) and Get (ScriptResult) exists

Structured variables can be stored

- in a Let () variable,
- in a local variable
- in a global variable
- in a persistent variable PV
- inside the sidereal space where the Get (ScriptParameter) and Get (ScriptResult) exists
- must be serialized on demand for PSOS or file-to-file communication

Structured variables can be serialized

Structured variables can be serialized

- With XML or any *XMLoid* technique

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique
- With Dictionary technique (SFR)

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique
- With Dictionary technique (SFR)
- With Property List technique

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique
- With Dictionary technique (SFR)
- With Property List technique
- With CLOB technique

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique
- With Dictionary technique (SFR)
- With Property List technique
- With CLOB technique
- With JSON functions (v16)

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With FMStandards technique
- With Dictionary technique (SFR)
- With Property List technique
- With CLOB technique
- With JSON functions (v16)
- With plug-in

Structured variables can be serialized

- With XML or any *XMLoid* technique
- With EMS's undocumented
- With JSON functions (v16)
- With plug-in

Short! It can become complicated

Or stay simple

CLOB

3+3+0+0+0|TheEnd

Ephemeral matrix demo

The ephemeral (butterfly) live only a few hours and takes the opportunity to mate in full flight. The male dies after mating and the female after laying.