Logging

Overview

From Wikipedia page on Logfile:

> *In computing, a **logfile** or simply **log** is a file that records events taking place in the execution of a system in order to provide an audit trail that can be used to understand the activity of the system and to diagnose problems. The act of keeping a logfile is called logging.*

Logging, as it applies to errors has been partially discussed on the Error custom functions & error source management page. However; this page is being created to discuss logging in a general sense, as it applies to the applications we write using FileMaker.

You can download a file that contains a proposed logging implementation. This is still a work in progress, but the basic concept is there.

Log Levels

The use of log levels allows your application to create log entries for multiple purposes, but managed by a single logging system. You may, for example, choose to include debug-level logging where you create a log entry at the beginning of every script. Or, you may only choose to include error-level logging where you only create log entries when an error occurs. This log level can be used to filter entries at the time of creation (based on a system setting, for example), or at time of viewing (based on a found set, for example).

I propose the following log levels as a standard that is recommended to be implemented by all logging system: (this is copied from http://stackoverflow.com/a/8021604/1327931)

level	name	description
1	error	The system is in distress, customers are probably being affected (or will soon be) and the fix probably requires human intervention. The "2AM rule" applies here- if you're on call, do you want to be woken up at 2AM if this condition happens? If yes, then log it as "error".
2	warn	An unexpected technical or business event happened, customers may be affected, but probably no immediate human intervention is required. On call people won't be called immediately, but support personnel will want to review these issues asap to understand what the impact is. Basically any issue that needs to be tracked but may not require immediate intervention.
3	info	Things we want to see at high volume in case we need to forensically analyze an issue. System lifecycle events (system start, stop) go here. "Session" lifecycle events (login, logout, etc.) go here. Significant boundary events should be considered as well (e.g. database calls, remote API calls). Typical business exceptions can go here (e.g. login failed due to bad credentials). Any other event you think you'll need to see in production at high volume goes here.
4	debug	Just about everything that doesn't make the "info" cut. Any message that is helpful in tracking the flow through the system and isolating issues, especially during the development and QA phases. We use "debug" level logs for entry/exit of most non-trivial methods and marking interesting events and decision points inside methods.
5	trace	We don't use this often, but this would be for extremely detailed and potentially high volume logs that you don't typically want enabled even during normal development. Examples include dumping a full object hierarchy, logging some state during every iteration of a large loop, etc.

For those following my suggestions/recommendations/custom functions, please note that the above set of log levels does not correspond to my previously used log levels. Even though some of the names are the same, their intended use, as described above is different than I have been using them.

Here are some links to documentation on other possible log levels:

- http://en.wikipedia.org/wiki/Syslog#Severity_levels
- http://httpd.apache.org/docs/2.2/mod/core.html#loglevel
- http://en.wikipedia.org/wiki/Log4j#Log_level

Modular

I propose that the Modular FileMaker format be used for creating a logging system. An example of this can be found in the GitHub project. Using the approach in the sample file, multiple "log writers" could be created for writing the log data to a different destination (memory, file, external system, etc.).

Implementation

In FileMaker, scripts are the only place that a log entry would be initialized from. All data being logged should be generated from within the relevant context **before** it is sent to a logging process. In other words, the logging process is not responsible for adding any additional data to the log.

When an error occurs, logging should be initialized by the script that generated the error.

To facilitate the retrieval of environmental data from within the correct context, the following custom function is recommended:

LogData (logLevel)

Returns a Let formatted dictionary of environment information based on the specified logLevel.

Name

What is an appropriate name for a general-purpose Logging system in FileMaker?

- ScriptLog
- SystemLog
- ApplicationLog
- EventLog

Further discussion about this topic can be found on the associated Google Group.