FileMaker & The Great Null Dilemma

We All Have a ~~World~~ Code View

This blog post exists because you, yes you the FileMaker developer, has both a perspective and opinion about "How things should be done". You have a mental picture of what defines a "standard" when it comes to development. You have either created your own system and methods or you "buy into" using someone else's - either outright or in part. You collect insight as you grow and become a better developer. Along the way your perspective changes.

You may favor code portability over legibility. You may favor being concise over being verbose. You may want your code to be self-documenting over having to create documentation. You may be biased towards zero dependancies (good luck with that in any modern code environment).

Or, you may favor the opposites of any or all of those situations.

To put this bluntly. NONE of them are "right" in their absolute.

For example, in text based coding languages, the fact that some developers like to use 2 spaces, where another may like 4 and yet some will standardize on a tab character means that we all have differing perspectives and coding desires. Sometimes it's chosen. Sometimes it's imposed.

Based on the comments which started on the Empty strings (null values) page on this web site, this blog post and the comments serve as a place for open discussion about the notion of "standards" within the world of FileMaker development - hence the topic titled **"The Great Null Dilemma"**.

Defining "Standards"

The definition of standards can be very murky unless clarified and agreed upon. The key point here is "**agreed upon**" for the **benefit of those choosing to adopt said standards**. If you're a single developer, there's not much disagreement - because what you say goes.

However, if you perform any degree of work which may be jointly shared, then arriving at an agreed upon set of standards is what promotes collaborative understanding. And a good degree of personal benefit if I may say so myself.

My own personal viewpoint comes from having worked with FileMaker since the mid 90's and also having worked with a variety of content management systems created within PHP. PostNuke, Typo3, Xaraya and Drupal. I've also created many scripts and done work with both PHP and Perl based on using repositories of code, such as CPAN and PEAR.

In most every situation where I've agreed with myself to use someone else's code and take advantage of it, I've had to do a good deal of learning. I had to learn how things worked and why they were done that way. I had to seek to understand. Once I had it, I could take advantage of what I had learned over and over. I am also able to come up to speed quickly on any updates to what I had already learned.

Please note, I didn't have to agree with how they did things, because the advantage to me was leverage. Leverage allows you to arrive at an end result faster than if you didn't have it. In other words. I can take advantage of a PEAR module (a pre-existing chunk of code which does something pretty specific in PHP) and save myself the time and hassle of coding it up myself.

I also became familiar with frameworks such as Cake PHP, Code Ignighter and more recently Zend. I also made an investment in really learning the Drupal CMS. Knowing these things does not make me a coding expert, but I do think it gives me a perspective beyond just working with FileMaker.

Having been involved with using Drupal at an early stage I watched the adoption of Drupal increase significantly. From a code perspective, I would like to say that it was solely because it was "good code". Yet you'll find many developers in the PHP world who will say it sucks. "Because it's not OOP", or some other reason (Drupal is actually using more and more OOP code now). My personal opinion for the reasoning behind increased adoption and acceptance of Drupal is because of the documentation. Both within the code and outside. Documentation is the line of communication between what the code does and why it does it that way. It allows another developer to **find the understanding necessary for greater leverage**.

It was for this reason that I started this web site.

When you have something documented, which can be referenced, it serves as the blueprint for why something works the way it does. Again, you don't have to agree with it if your main objective is leverage.

Everyone is free to write all the code they want. Personally, I don't want to write everything. There's little point - unless I feel like I have an endless amount of time to do so.

The Perspective Difference

Some developers love to sling technical terms around what they do within FileMaker. They like to apply Object Oriented principles to their scripting and how they structure a database. They'll say they've created the coolest FileMaker system based on MVC (Model View Controller). I know, because I've done it. I've had the pride that comes with applying some cool sounding technical term or process to the world of FileMaker development. However, I know they don't really apply. Maybe their premise does, but it's not quite accurate - because FileMaker is not like other coding environments. FileMaker is like FileMaker, just like PHP is like PHP.

The truth is this. Your perspective about your own code is unique to you based on what you've created yourself, what you've adopted from others and how you've done things in the past. Changing to a "new way" of doing things always takes more effort than doing them the way you've always done them. Yet, it's easy to become stagnate and fall behind because you don't evolve with the "rest of the herd".

Inevitably, most developers will think their way is the best. I'll admit this to myself. It bolsters ego and validates what you know. Yet, I must also admit that my perspective is pretty unique - in that I'm the only one who has it.

When it comes to "standards", this is where perspective can be removed from the development equation - if, and ONLY if, you desire it. When a set of standards frees you from having to code everything yourself because you can leverage pre-existing content based on those standards you stand to gain the greatest advantage.

Herein lies "The Great Null Dilemma".

If an agreed upon standard outlines something which clashes against your personal perspective, you end up having to make a decision. Change your ways - or - don't. Stay with the project - or - don't. In the world of PHP frameworks there are plenty. In the world of FileMaker, there are few. The ones that are documented are the ones which exist for consumption - otherwise they exist in obscurity.

The fact that a standard exists does not mean you must use all of it - unless you expect what you create to interconnect with other things created under that same set of rules - and be understood by those who adopt the standard.

As I watched the world of Drupal grow up from 3.6 to 4.7 to 5, 6, 7 and now 8, I've seen every type of "coder" personality. From flexible to inflexible.

Yet despite the personality type, or personal desires regarding the "how" something should be done, there was one thing that kept some of the perspective out and kept the project growing.

Documentation.

Make your Case

That's what this site is about. Documentation about the standards that "we" agree on. If we agree (we being the majority who have adopted) that using an empty custom function named Null does create a dependency, yet that dependency is acceptable because most projects opt for readability and comprehension over portability. Then we let the documentation speak to that fact.

If our concern is greater adoption of the standard, because that validates our own work and efforts, and portability is a higher priority, then it's quite obvious that we kill the empty custom Null function.

Regardless of whether you agree with everything, some things or nothing here, if leverage is your objective, and you've obviously got a stake in the game - then make your case. Show, why you feel the way you do and convince others of what you say. Group participation and documentation is what makes for a good set of standards.

Hopefully, the content you find on this site will ultimately benefit your choice to continue to work with the FileMaker platform. Whether you adopt or not. If you do choose to use these standards then participate. Just don't expect that "these" standards will be "your" standards - they may not be.

So, let's open the debate and create something that helps those of us who are here.

We can't be all things to all people - lets not try to. It's really hard - and tiring.