

PHP Functional Library Overview

Introduction

The following paper will provide an overview of the rationale, advantages and basic structure associated with a PHP functional library. The content is based on my personal experience gained from using PHP extensively for the last 7+ years in a corporate setting (MIS).

Rationale and Advantages

A deployment of a functional library has value if the count of application programs (PHP) has the potential to grow beyond a few programs. The mechanics of reproducing the same functions natively in each application program would become quite hard to manage with an increasing program count. Additionally, to include the raw functions in each application program chews up a lot of unnecessary application program space.

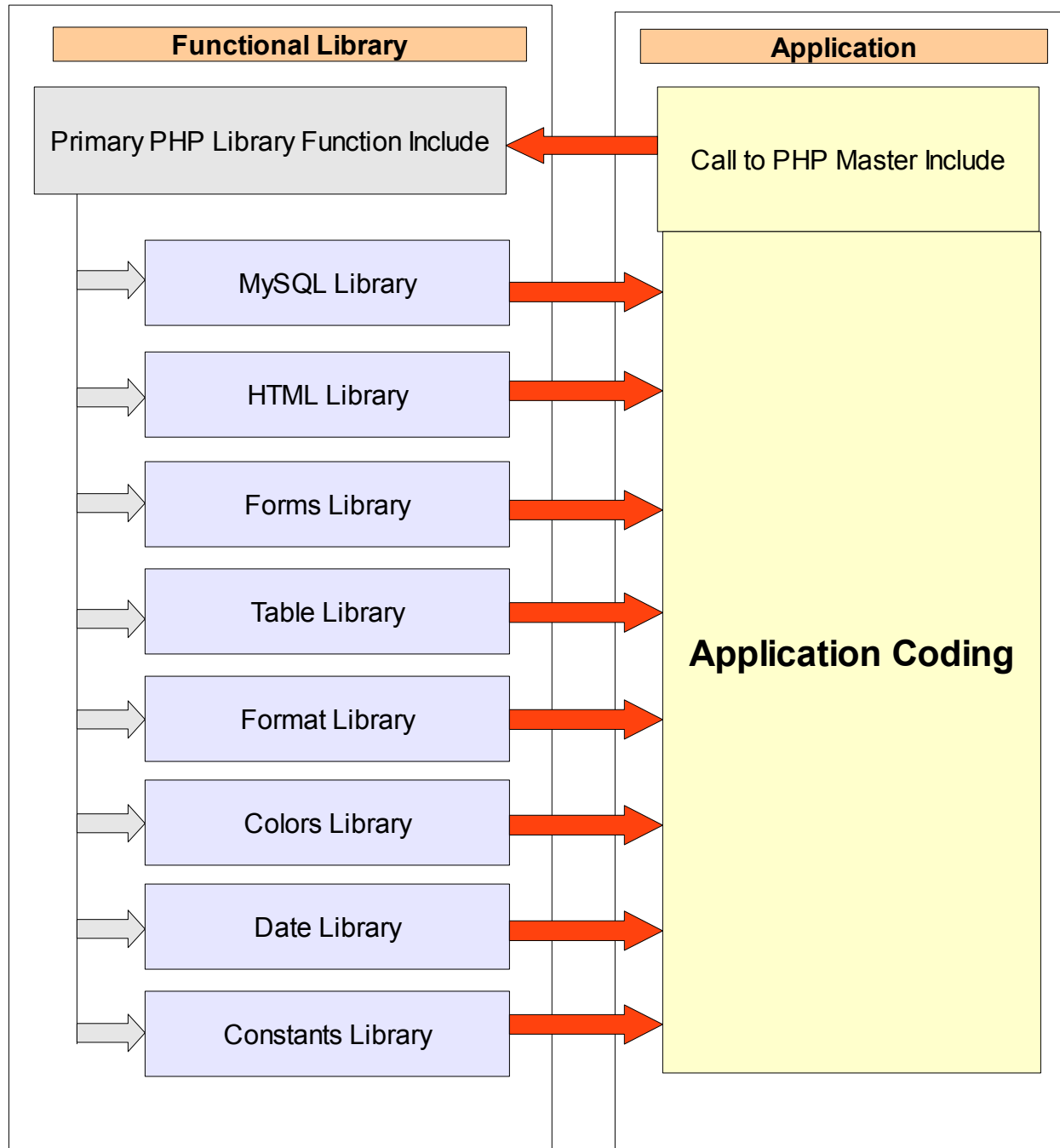
The use of a functional library offers a consistency of execution each time the function is called by the application. In a library setting, if you make a mistake in the core library function, you will realize it on the first call (or even at the initial "include" statement if it's a basic run-time error in the function). Once the library function is correctly written, calls to it should execute consistently.

One of the main reasons I started deploying a functional library was to increase the speed of coding the basic applications. Early on, I found myself in the situation where I was hard coding the basics of the html generation repetitively in each application. The initial library members consisted of functions to support the core html parsing necessary to support the applications. In today's environment coding complex applications (1000-3000 lines of code) is appreciably faster and more fault-tolerant in the functional library environment versus a traditional hard coding environment. The efficiencies that you experience are a direct result of the functional library that you assemble. The more extensive your library, the faster the coding will be.

Just about everything that the functional library supports could be achieved with object oriented programming. I choose to go the functional library route because I did not want to go through the repetitive situation of instantiating an O-O within the application. I preferred doing the functional call "on the fly" as needed by the application. Because I'm currently using recursive functionality (one function makes a call to another function etc.), the complexity of the functional library approaches or may even surpass that of a object oriented system.

PHP Functional Library Overview

The Basic Structural Layout



PHP Functional Library Overview

Conclusion:

The use of a functional library will provide a quicker, more consistent coding environment. Because the native html etc. is not hard coded in each application, the chances of coding errors are reduced (definitely not eliminated though). Going through the rigors of developing and deploying a function library is probably best served in an environment that contains a number of programs.

One possible down side of a functional library is that the coding of the applications becomes very specific to the supporting functional library. In essence, the developer is using a second language layer on top of the basic php to program the applications. If the applications are to be supported, then good documentation becomes very important.

Please refer to the web site for addition papers on the actual mechanics and some starting examples of functional library members and uses.