# Understanding MVC in PHP

This article series (continued in Implementing  MVC in PHP: The Controller, Implementing  MVC in PHP: The View, and Implementing  MVC in PHP: The Model) demonstrates how to build an MVC web framework using PHP 5. This article covers the basics of MVC web frameworks, building the foundation classes for a framework that the other three articles in this series will build.

With the introduction of PHP 5 and its new OOP features developers can now seriously talk about building solid APIs and more complex MVC frameworks for the web in PHP. This was possible before, but the new features in PHP 5 make it easier to integrate more advanced features into MVC frameworks, such as SOAP and WSDL.

In this article I assume that you have a solid understanding of object-oriented programming and that you have at least scanned the upcoming changes to the OOP structure of PHP in PHP5.  What is MVC?

MVC is the idea that you have three different pieces that work in unison to form a complex application. A car is a good real-world example of MVC. With a car you have two views: the interior and the exterior. Both take input from the controller: the driver. The brakes, steering wheel and other controls represent the model: they take input from the controller (driver) and hand them off to the views (interior/exterior) for presentation.  MVC on the Web

The ideas behind MVC frameworks are quite simple and extremely flexible. The idea is that you have a single controller (such as index.php) that controls the launch of applications within the framework based on arguments in the request. This usually includes, at a minimum, an argument defining which model to invoke, an event, and the usual GET arguments. From there the controller validates the request (authentication, valid model, request sanitization, etc.) and runs the requested event.

For instance, a request for /index.php?module=foo&event=bar might load a class called foo and run foo::bar(). The advantages of this method include:
 - A single entry point for all applications.
 - Removing the headaches involved with maintaining numerous scripts, each with their own relative paths, database connections, authentication, etc.
 - Allowing the consolidation and reuse of code.  Why Create My Own MVC Framework?

This article doesn't really advocate "You should write your own MVC web framework!" as is tries to explain "This is how MVC web frameworks work in theory, and why they are so great."

As of this writing, there are very few true MVC frameworks written in PHP. In fact, there is only one that I know of, Solar, that is entirely pure PHP 5 code. Another one out there is Cake, which is trying to be the "Ruby on Rails of PHP." I, personally, have a few problems with both of these frameworks. Both Solar and Cake fail to leverage existing code in PEAR, Smarty, etc. Cake appears a bit disorganized at the moment. Finally, Solar is the work of mostly a single person (not that Paul isn't a great coder or person, but there is only a single gatekeeper at the time of this writing). These may not be issues that concern you, and if they don't concern you, by all means check these two out.

Read More