Ian Jablonowski

You’re interested in this whole new idea of “frameworks,” but why spend time on the learning curve when the system you already have in place works fine? Well, you might find yourself using the same functions or classes over and over; modifications of simple tools you would think would already exist in PHP.

Well, frameworks are your solution. A framework provides all the standard functions you would hope would be included in PHP, as well as a structured format for your files and directories. What are the advantages to this? Well, the code you need to handle cookies, sessions, dates, downloads, uploads, input, output, dates, pagination, templates and most security issues already exists in structured files. The only thing left is to make use of the provided functions.

The framework that I have currently been using the most is referred to as CI, or CodeIgniter. It requires no use of the command line or any process installed on your server. It has a very well written documentation, a low footprint and happens to be one of the higher performing frameworks. What I find most attractive however, is that rather than constricting you in to a MVC model (more on this later), CodeIgniter simply presents the model to you, and it is up to the programmer to follow the model as loose or strict as they want.

The site I’m using CodeIgniter on is called VentStatus at http://www.ventstatus.com. VentStatus is a tool used to track the users and servers of a VOIP (voice over internet protocol) program called Ventrilo. Ventrilo acts similar to a game server. It allows users to join a server with multiple channels. Each channel acts as a room; (in most cases) users can only speak to other users in the same channel. There are multiple channels, as well as sub channels, per server. VentStatus allows people to check what a Ventrilo server looks like without actually logging in. It also constantly queries Ventrilo servers for things like their user count, their uptime, channels, and current users. Using this data it can link up who has been on what server for how long, and what other servers share the same channels and users. CodeIgniter is a good choice for this kind of website. An external process is called to check the status of servers, so CodeIgniter’s caching and speed helps to make up for lost time calling the process.

Most frameworks are based off of the MVC framework. M stands for model, C for controller, and V for view. Though this isn’t exactly the way I would go about describing the MVC model to anybody.

I would start with the Controller part of MVC. The controller is what is responsible for converting the address bar into directions for your program. The first directory in a given URL specifies a file and class to call. The second directory specifies the function of that class to execute. For example, http://www.ventstatus.com/servers/status/ finds the servers.php file, calls the Servers class, and then executes the status function. Any directories given after /status/ are converted in to variables given to the status function. Basically, the model function gives the framework direction. The advantage of this is what we call search engine friendly URLs, or URLs that are readable to humans, as well as search engines.

The next part to introduce would be the Model aspect of MVC. In my head models act as containers full of tools you’re able to make use of in any area of you web application. For example, in my users model, I have a list of functions that relate to manipulating user data. Some of my functions in include adding users, updating users, checking a users last online times and combinations of the past three based on input variables. This helps your program stay organized as you can call any set of tools you need as they are needed, rather than including mass amounts of functions that are unneeded in some areas of your site. Models are not restricted to any certain Controller, and are not called for by the address bar.

Last, but from my perspective is one of the more important functions of CodeIgniter is the view part of MVC. The view part of CodeIgniter allows you to keep your content and presentation separate, which has been a repeated idea throughout design. All of your data is taken and made in Model and Controller, and theoretically there should be nothing left to calculate by the time View swings around. Thankfully though, CodeIgniter is not strict, and you can break the rules if you need. A view page acts as a template for all the data outputted by the controller. So if you were to put somebody else in charge of site design, who really had no idea how to program or wasn’t familiar with PHP, you could have them or anybody else work on the page without affecting any actual functioning code. This allows allow you to make consistent header and footer templates to include on every page in your site, rather than coding the same thing over and over again.

To throw it all together into my current project, I have a way to search for users, channels, and servers. Now obviously each of these types of searches are going to allow different ways to display and sort data, but each should be outputted on a search like page. So our controller is called “search” and contains three functions, users, channels, and servers. Each of these call a few models to help the search, and then are displayed through a corresponding View file pertaining to the search type. However, each page contains the same header and footer View.

Now one of the great things about CodeIgniter is the prewritten libraries and classes to help you write code quicker. These range from database classes to text formatting. They can be included as needed, and really help simplify things. The whole point of CodeIgniter is to help you by not “reinventing the wheel” so to speak. You have more time to focus on the fun things like developing new software rather than trying to find out how to account for daylight savings time in Utah.

The turnoff however, is the learning curve often seen when working with a new framework. CodeIgniter doesn’t require users to shape their code into some molded format; it just provides an outline or guide to keep things organized. The well written documentation helps users with any questions they might have about CodeIgniter specific functions. And if something comes up that may pertain to only you, the support forums are filled with nice people eager to help.

But to every good thing there is a downside. CodeIgniter has its downfalls here and there. I’ve ran into a few hills while working on VentStatus, but the time CodeIgniter has saved me while developing makes up for it and then some.

For one, the way CodeIgniter formats its URLs is a little buggy, not to mention confusing. For example, anything that is not an image, php, css, or html document is routed through CodeIgniter’s URL system. Linking to something like a pdf in a subdirectory of your web server won’t work unless you add an exception to your .htaccess file or CodeIgniter files. There may be tricks around this, but I haven’t looked too far into it yet. Another problem may be specific to my webhost (Dreamhost) who uses some non standard practices in their Apache setup. However, I found support quickly in CodeIgniter’s support forums.

Overall there isn’t much I can really say is bad about CodeIgniter. Nothing about CodeIgniter is mandatory. It basically allows you to code as you normally would, but with the addition of a huge amount of resources. The directory, file, and naming structure are all mostly optional. If you wanted, all your code could happen inside the Controller aspect of CodeIgniter and everything would work out just as fine as if it were split up into the three parts. CodeIgniter is definitely the solution for a developer looking for a speedy, useful, easy going framework with a minimal learning curve.  
  
Word count (including the last two, and these): 1371