How does a PHP application work?

Most of us are familiar with how a web application works. We know that when we enter an URL in our browser, it'll contact a web server who'll return some HTML. The browser will then make follow-up requests and then render the web page. (This is a simplification! But that's the main idea.)

But what's happening on the web server returning HTML? How is the web server getting the HTML that it returns to our browser? And, to be even more specific, how does this happen in PHP?

This isn't something that most of us know how to answer. We know (or at least should know!) how to build a PHP website. But what happens between our PHP code and the browser is a bit of a mystery.

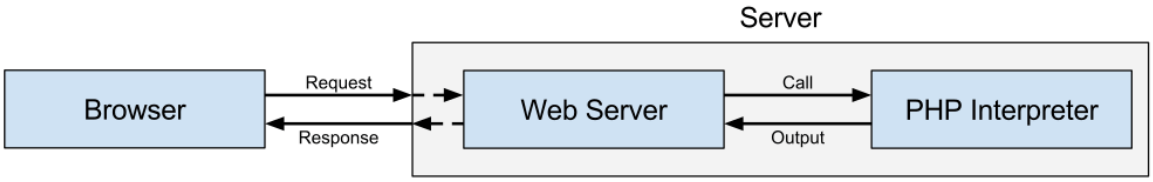
There's a good chance that you're thinking, "Well, my site loads. Why should I think twice about this?" And that's true! But the truth is that knowing how a PHP application works can be useful.

It can affect how you design your PHP application. But it's also important if you're looking to improve its performance. For example, it's hard to understand PHP application caching if you don't understand how a PHP application works.

The life of a PHP request

If you come from a compiled language (like Java or C#), the biggest thing to understand is that PHP isn't one. It's an interpreted language. This means that PHP will need to interpret and compile the code of your application for each request made to it.

Now, you can use caching and other optimizations to speed this process up. But the nature of how requests to your PHP application work won't change. They'll always look like this:



As you can see in the diagram above, we always start with a browser making a request for a web page. This request is going to hit the web server. The web server will then analyze it and determine what to do with it.

If the web server determines that the request is for a PHP file (often index.php), it'll pass that file to the PHP interpreter. The PHP interpreter will read the PHP file, parse it (and other included files) and then execute it. Once the PHP interpreter finishes executing the PHP file, it'll return an output. The web server will take that output and send it back as a response to the browser.

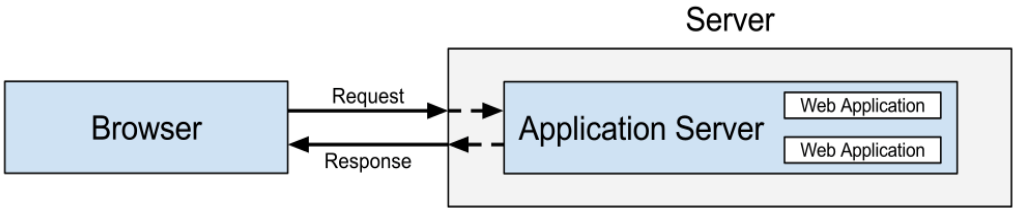
A PHP application is just a script

This was the goal of explaining the life cycle of a PHP request was to get to this statement. This is the truth behind any PHP application whether it's a Laravel or Symfony application. (Or even a CMS like Drupal or WordPress.) They're all nothing more than scripts much like a shell script.

In this scenario, the web server is nothing more than an intermediary. Each time that someone makes a request for a PHP page, it's just asking the web server to run a specific PHP script. If the PHP script returns any output, the web server sends it back.

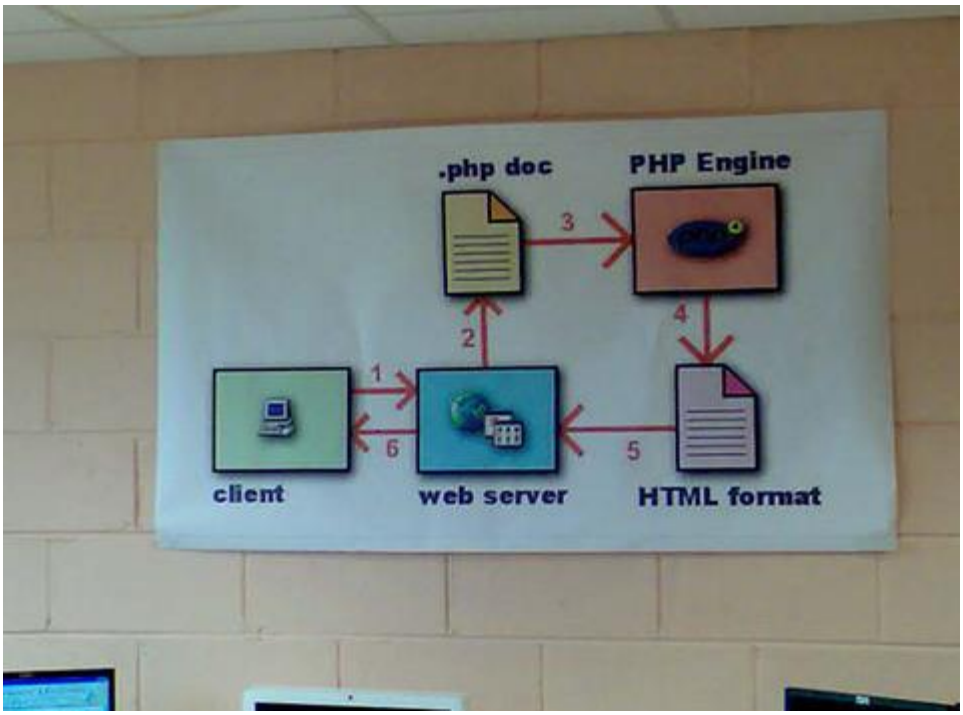
This is different to what you'll see in with other web programming languages. These web programming languages create what we'll call more traditional web applications. These traditional web applications often work using an application server. Some well-known application servers are Tomcat for Java, IIS for .NET, Gunicorn for Python, Unicorn for Ruby.

Working with an application server is different in some fundamental ways. The most important one is that a traditional web application resides within the application server. It's not a script that your web server can call. In fact, these application servers also act as web servers. The diagram below should help you visualize this.



As you can see, the web application is inside of the application server. (There can even be more than one web application per application server!) This application server is also your web server. Any request that you make to it is the same as a request to the web application. It's not an intermediary like it is with PHP.

You Can Understand It Like This :



"image : basic diagram showing how php works, source : flicker, google"

PHP is a scripting language which writes HTML codes for us (but why?) because they are supposed to change every single hour, day or months (who cares, it depends upon the data we collect). so that we don't have to write every single change by ourselves, we made it automatic through PHP.

Websites which change very often(like facebook) is widely known as dynamic website, the HTML code for dynamic websites is practically not possible to write by manually or by ourselves (i have explained why), so we made a program which does this heavy lifting/work for us, we just have to tell what it have to do, by writing PHP code. this way we save our time, and made possible to produce dynamic contents.