PHP

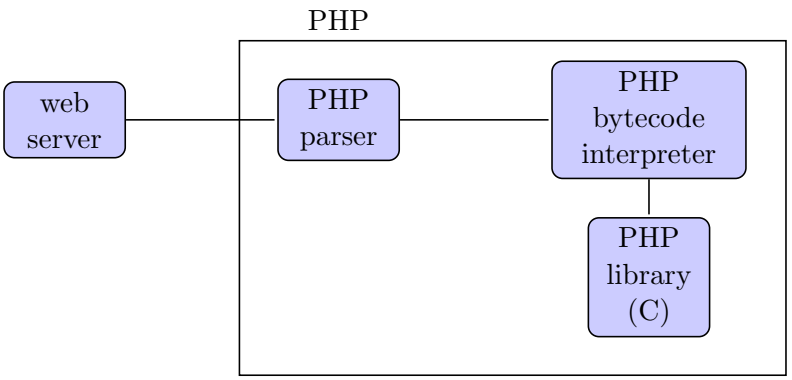
Let's briefly investigate one particular programming language for dynamic webpages: PHP. Invented by Rasmus Lerdorf, a SYDE grad ('93).

We can think of PHP as an extension of the server-side includes we saw last time. It is a low-level language: unlike WIG, it doesn't provide an abstraction over the state. It just processes directives on pages. There are some frameworks that build on top of PHP and provide higher-level functionality, e.g. CakePHP.

Programming Model. PHP goes beyond SSI by including more features from typical programming languages: functions, objects, looping, and lots of built-in functions. However, as in SSI, the `echo` statement is still one of the most important statements.

Database Integration. Basically, you get to write SQL commands in your PHP script, parse the results, and manually integrate them into your scripts. This is a significant increase in power over SSI, because you can store large amounts of data persistently. Frameworks like CakePHP can help by automatically relating PHP classes to database tables, but you'll probably still find yourself writing SQL code.

Execution Model. Like Apache SSIs, PHP scripts are executed on the server. PHP usually runs as either a module included in your web server, or as a CGI script. In any case, when a web server knows that it has a PHP page to process, it hands it off to the PHP interpreter and sends the resulting page to the client.



There is a PHP parser, which converts to bytecode, and then a bytecode interpreter. Apparently this interpreter is slow, but most of the computations occur in the PHP library, which is written in C and fast. What recently-seen concept does this remind you of?

There has also been work on PHP compilers, e.g. HPHP and `hpc`, which take different approaches to compile PHP to binaries ahead of time. HPHP[1] is a source-to-source translator converting PHP to C++. ("Three main steps: static analysis, type inference, and code generation"). It does not handle the funky functional-language features of PHP. `hpc`[2] is a full compiler for PHP, but it produces slower code at the moment, since it is more researchy and less productiony than HPHP[3]. See its webpage for a number of applications of compiler technology: one can use it on PHP to compile scripts into executables, extensions, pretty-print, obfuscate, optimize, and analyze.

Multiple servers

Note that the front-end web servers, the middleware (which executes application logic), and the databases can all reside on different machines. There can be multiple machines at each level. Domain-specific languages and concepts like MapReduce help use all these machines productively.

Client-side Programming

Let's briefly examine all of the domain-specific languages on the client side.

- HTML/CSS: domain-specific languages for content and layout. Typically doesn't contain "code" (programs).
- JavaScript: the main programming language in the browser. A functional language. Often executed by just-in-time compilers. Manipulates the document's HTML and CSS.
- Flash: Another DSL which runs in plugins in browsers. Contains a virtual machine (which is why it took so long to be ported to Linux x64).

There are also domain-specific languages which run on top of JavaScript, like jQuery[4], which make it easier to write certain kinds of JavaScript code. (It extensively uses higher-order functions, by the way).

Client-side programming enables the development of applications which respond instantly to user input, which make their features much more usable (response time matters).

[1] `http://developers.facebook.com/blog/post/358`
[2] `http://www.phpcompiler.org`
[3] `http://blog.paulbiggar.com/archive/a-rant-about-php-compilers-in-general-and-hiphop-in-particular/`
[4] `http://jquery.com`