PHP 5 Release

by
Peter Lavin

As published in Spider, September 2004

Introduction

July 13 2004 marks the official release date of PHP 5. But so what? Well, as early as 2002 PHP was the most dominant server-side scripting language and it has continued to grow by leaps and bounds since. What started out as a hobby language called “Personal Home Page” has grown up to become “PHP Hypertext Preprocessor”.

PHP is *the* “P” in the acronym for the open source web platform, “LAMP” (Linux, Apache, MySQL and PHP). Apologies to Perl and Python programmers but Perl, if it ever was king, has been dethroned and Python is, at this point anyway, a mere pretender.

It is in this context that we should view the importance of this release. Computer languages, like their natural counterparts, must keep pace with changes in the environment or risk becoming irrelevant. Release 5 consolidates PHP’s position as the top server-side scripting language and addresses some of the shortcomings of the language. With that aside let’s look at some of the important new features introduced in this version.

What’s New

A quick listing of the major features, more or less in order of importance, follows:

1) Improved object model
2) Support for newer versions of MySQL
3) Introduction of SQLite
4) Completely revised XML support
5) A new extension called SimpleXML

Let’s deal with each of these items in turn.

Improved Object Model

The new object model probably warrants the most attention as it is the main focus of this release. In PHP 4, the limited support for object-oriented (OO) programming seemed more an indication of intention rather than something that was actually useful. Given the limited support for OO programming I did not see that there was much incentive to use it. Well things have changed appreciably. This release incorporates all the major features required by any language that wishes to be considered object-oriented. This includes a slightly different

approach to constructors, and the addition of copy constructors, destructors and private and protected data members.

Support for interfaces has also been introduced. At the early stages of development, there was discussion of possible support for multiple inheritance of classes. Fortunately, at least in my view, this did not come to pass. Instead, just like Java, only multiple interfaces may be inherited. This is a sensible approach that reaps the benefits of multiple inheritance without the drawbacks. While not strictly a requirement of OO programming, exceptions may now be handled using a try/throw/catch structure.

I won't go into all the nitty-gritty details but any OO programmer, especially those familiar with Java, will find a complete and easy-to-understand object model; while procedural programmers can continue to ignore the object-oriented features of the language. Suffice it to say that OO programming with PHP is now the real thing and no longer a case of "let's pretend".

These changes necessitated changes to the scripting engine behind PHP, namely the Zend engine; hence the Zend II engine. Of particular importance is the fact that, in many situations, objects are now dereferenced instead of being unnecessarily replicated. By all accounts these changes have made for more efficiency and fewer bugs though extended use will be the true judge of this.

Apart from object-oriented language additions, a number of new functions have been added. The majority of these relate to arrays, streams, iconv functions (converting from one character set to another) and ibase functions (the Borland database format). One or two of the miscellaneous changes jumped out at me as particularly useful, "image_type_to_extension()" for example. Have a look at the PHP site for documentation of these changes and I'm sure you'll find something equally useful.

As with any language update, backwards compatibility is always a concern. Obviously, in cases where features have been added, there is no question of breaking existing code. Otherwise, unless you have been pushing the language to the extreme, almost all of your code should run unchanged under PHP 5. And of course, if you have OO code it can be vastly improved by rewriting it to take advantage of these changes.

MySQL Support

The preeminent scripting language needs to keep pace with the preeminent open-source database. And for this reason, some people might well view these changes as more important than any of the others. However that may be, we

now have programmatic access to the additional functionality available in MySQL versions 4.1 and higher.

In order for this extension to be available PHP must be compiled with support for it. The features available in MySQL are now supported by PHP and in an object-oriented fashion. For example you can now create a prepared statement in the following fashion (example taken from the php.net website):

```
/* create a prepared statement */
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {

   /* bind parameters for markers */
   $stmt->bind_param("s", $city);

   /* execute query */
   $stmt->execute();

   /* bind result variables */
   $stmt->bind_result($district);

   /* fetch value */
   $stmt->fetch();

   printf("%s is in district %s\n", $city, $district);

   /* close statement */
   $stmt->close();
}

/* close connection */
$mysqli->close();
```

Procedural programmers can relax – the same effect can be achieved in a non-object oriented way.

SQLite

SQLite is a database engine that is incorporated right into PHP. It is not going to replace any of the existing databases supported by PHP but may be the ideal solution in a number of situations.

If your web host supports PHP 5 then creating a database as easy as:

```
$mydb = new SQLiteDatabase("mydb.sqlite");
```

Then simply use SQL syntax to create and populate tables.

This database engine would certainly not be the solution for presenting a catalogue of a few thousand items but it may be ideal for a small site that does

not otherwise have database support. Even when other databases are available SQLite performs exceptionally quickly and, for small, read-only applications where speed is critical, it might be the ideal solution for any website.

XML

In PHP 5 all XML support is now provided by the libxml2 XML toolkit (http://www.xmlsoft.org/). The underlying code for the Simple API for XML (SAX) and for the Document Object Model (DOM) has been rewritten. DOM support has been brought in line with the standard defined by the World Wide Web Consortium (W3C). Unified treatment of XML under libxml2 makes for a more efficient and easily maintained implementation.

This is very much an improvement over PHP 4, both in terms of performance and functionality.

SimpleXML

The online documentation for SimpleXML warns, “This extension is *EXPERIMENTAL”*, so we won’t spend too much time on it but it’s to XML what SQLite is to databases. It provides easy access to XML and a quick way of converting XML documents into PHP data types. The “simplexml_load_file()” can be used to create an associative array that you can easily iterate through using a “foreach” loop. Again, the libxml2 toolkit was used to create this extension.

Conclusion

You can now take OO programming seriously in PHP. It incorporates all the major features necessary. Support for objects has been grafted on to the language in much the way that C++ was grafted on to C, but as with C++, you can choose to use objects or simply revert to procedural programming. That PHP is a hybrid language should not be viewed as a negative feature but as an advantage. There are some instances where you will simply want to insert a snippet of PHP and others where you will want to make use of the language’s object-oriented capabilities. Purists may be offended but then again they probably aren’t programming web applications, given the hybrid nature of such beasts.

While PHP has excellent support for many different databases, support for MySQL should quite rightly be a top priority and PHP 5 keeps pace with the changes made to MySQL.

SQLite is a nice addition for reasons of economy and performance. If your web host levies charges for each database used, SQLite may be not only be cheaper but also yield performance improvements.

In the past, XML support in PHP has been the cause of some embarrassment. This is no longer the case. The unified approach to support for all aspects of XML makes things much easier for developers and should also make the future improvement of PHP easier.

Of course none of this will make any difference until PHP 5 is installed on your web server. If that decision is yours, you can see that there are compelling reasons to do so and if not, you'll have to wait until your web host decides to upgrade to PHP 5 but my expectation is that you won't have to wait very long.

Resources

http://www.xmlsoft.org/
http://www.php.net/
http://www.zend.com/
http://www.sqlite.org/
http://www.zend.com/news/zendpr.php?id=49

About the Author

Peter Lavin runs a Web Design/Development firm in Toronto, Canada. For more information visit http://www.softcoded.com. He may be reached at peterlavin@sympatico.ca.