Beginning PHP and MySQL

Table of Contents

1: Introducing PHP 1

2: Configuring Your Environment 3

3: PHP Basics 8

4: Functions 15

5: Arrays 17

6: Object-Oriented PHP 22

7: Advanced OOP Features 27

8: Error and Exception Handling 29

9: Strings and Regular Expressions 32

10: Working with the File and Operating System 42

11: PEAR 49

12: Date and Time 50

13: Forms 55

14: Authenticating Your Users 58

15: Handling File Uploads 62

16: Networking 65

17: PHP and LDAP 69

18: Session Handlers 69

19: Templating with Smarty 74

20: Web Services 80

24 MVC and the Zend Framework 84

# 1: Introducing PHP

PHP was spawned by a Canadian developer's Web page traffic counter. 1

PHP 6 was still a beta release at the time of writing. It was stable enough to write about, but some of the features might have changed before the final release.

PHP (Personal Home Page) began when Rasmus Lerdorf developed a Perl/CGI script to know how many visitors were reading his online resume in 1995. 2

By the release of PHP 3.0 in June 1998, more than 50,000 developers were using PHP for their Web pages.

Two leading developers, Zeev Suraski and Andi Gutmans, rewrote the PHP parser and dubbed it the Zend scripting engine. This was the PHP 4 release.

PHP 4.0 was released on May 22, 2000 – about 18 months after the first official announcement of the new development effort. Many considered this to be PHP's official debut within the enterprise development scene. 3

Native session handling was added.

PHP 4.0 included ISAPI support allowing it to work in conjunction with Microsoft's IIS Web server.

Another 4.0 bonus for Windows users was native COM/DCOM support. 4

Native Java support was also offered. PCRE (Perl Compatible Regular Expressions) was also added in 4.2.0.

It was PHP 5 that would establish PHP as the 800 pound gorilla of the Web scripting world.

PHP 5 offered vastly improved OOP capabilities. This was PHP 5's most visible feature.\

PHP 5 added try/catch exception handling.

XML support is now based on the libxml2 library and SimpleXML was introduced. 5

Native supporrt for SQLite was also introduced in version 5.

At press time, PHP 6 was in beta and scheduled for release at the end of 2007.

Some though making this a major release number was curious because the only particularly significant feature added was

Security improvements have been made to lesson the impact of "inexperienced programmers running with scissors." 6

PHP developers have more than 180 libraries with well over 1,000 functions availaible. 8

In addition to manipulating databases, it can:

* Create and manipulate Flash and PDF files
* Evaluate password strength
* Parse complex strings with Regular Expressions
* Authenticate uses against flat files, databases, and even Microsoft's Active Directory
* Communicate with many protocols like LDAP, IMAP, POP3, NNTP, and DNS
* Tightly integrate with a wide variety of credit-card processing solutions.

Native support is offered for more than 25 database products. 9

PHP offers comprehensive support for both procedural and OOP programming.

PHP allows you to quickly capitalize on your current skill set with very little time investment.

# 2: Configuring Your Environment

Although Apache 2 was released more than five years ago, 1.x remains in wide use today. 13

If you are using Apache on Windows, it is recommended to go with version 2 because this was a complete re-write and is much more stable.

If you plan to use Apache on Windows, use the Zip archive version of PHP rather than the installer. 14

The Apache manual can be found in the docs directory in both XML and HTML formats.

The PHP documentation is located under the manual directory. You can download alternate formats from the Web site, including the Docbook source. 15

The php-ini-recommended version of php.ini has more security enhancements than php.ini-dist but may not be fully compatible with some legacy applications. 16

Apache 2 was completely rewritten to take advantage of optimizations for the Windows platform. 17

When installing Apache, consider changing the destination folder to C:\ - this will create c:\apache2 rather than installing under the default location in Program Files. 18

After unzipping PHP to the appropriate folder, you will add the following to Apache's httpd.conf file:

LoadModule php6\_module c:/php6/php6apache2.dll

AddType application/x-httpd-php .php

PHPIniDir "C:\php6"

19

Rename the php.ini-dist file to php.ini and copy it to c:\php. 20

Microsoft has developed FastCGI in conjunction with Zend to make interaction between IIS and third party vendors better. It is available but not yet intended for production environements.

To use FastCGI you need IIS 5.1 or greater. 21

The best way to test your PHP installation is by running a script:

<?php

phpinfo();

?>

23

Running phpinfo() is a great way to learn about your PHP installation. It has extensive information about the server, operating system environment, and available extensions.

24

Don't forget to add index.php to Apache's DirectoryIndex directive. 25

On Linux, you can get a complete listing of the more than 200 possible PHP configuration flags with the following command:

./configure /help

26

On Windows, to enable one of the many extensions that come bundled with the default build, you need to uncomment them in php.ini.

On both Windows and Linux it is possible to change PHP's behavior at runtime through php.ini. 27

These directives can be manipulated in various ways:

* Through php.ini
* Via Apache's httpd.conf and .htaccess files
* Through a PHP script

The following aspects can be controlled through the php.ini file:

* Language options
* Safe mode
* Syntax highlighting
* Miscellaneous
* Resource limits
* Error handling and logging
* Data handling
* Paths and directories
* File uploads
* Fopen wrappers
* Dynamic extensions
* Module settings

After you're comfortable with a parameter's function, consider deleting the comments to speed up subsequent editing.

28

If PHP is installed as a CGI binary, php.ini is read every time PHP is invoked. If it is installed as an Apache module, it is read only once when the Apache daemon is first started – thus requiring restarting Apache if changes are made. 29

For Apache httpd.conf and .htaccess file directives, you just use one of the following prefixes:

* php\_value: (sets value of the specified directive)
* php\_flag: (sets value of the Boolean directive)
* php\_admin\_value: (cannot be overridden in .htaccess)
* php\_admin\_flag: (cannot be overridden in .htaccss)

The following disables the short tag and cannot be overridden by others:

php\_admin\_flag short\_open\_tag Off

You can use ini\_set at the top of a script to override a php.ini setting in that script. To override the maximum execution time:

ini\_set("max\_execution\_time", "60");

There are four scopes for when a directive can be modified:

* PHP\_INI\_PERDIR (php.ini,httpd.conf,.htaccess)
* PHP\_INI\_SYSTEM (php.ini, httpd.conf)
* PHP\_INI\_USER (within user scripts)
* PHP\_INI\_ALL (anywhere)

30

engine = On /Off determines if the PHP engine is available at all.

The short\_open\_tag directive determines whether or not the <? ?> syntax is allowed. This can conflict with HTML so it can be turned off. 31

The asp\_tags (<% %>) directive is no longer available as of PHP 6.

Output buffering was established to work around the problem of people trying to alter the headers after output was already sent. If you don't anticipate needing this feature, turn it off because it degrades performance slightly. 32

The output\_handler directive allows you to pass all output through the specified function before it is returned to the browser (like ob\_gzhandler that can compress content). 33

The zlib\_outpout\_compression directive can be set to On or to an integer value (to assign a compression level). This will cause all output to be compressed before sending it to the browser. Compression is an HTTP/1.1 feature supported by nearly all modern browsers.

The safe\_mode directive, which disallows many PHP functions, if often turned on by ISPs in a shared hosting environment. It will disallow functions like chown(), exec(), system(), and the backtick operator 35

Due to confusion over its name and unintended consequences caused by file ownership, safe\_mode has been discontinued in PHP 6.

safe\_mode\_include\_dir and safe\_mode\_exec\_dir directives allow for folders where ownership isn't checked or exec() can be run is safe\_mode is enabled.

The disable\_functions directive allows you to disable specified PHP functions. This directive is NOT dependent on safe\_mode.

disable\_functions = "exec, system";

37

The disable\_classes directive allows you to disable certain classes you may have in your class libraries. This directive is not dependent on safe\_mode.

The ignore\_user\_abort (default On) flag allows you to tell PHP to continue processing even if the user terminates the session (which you may want to do if you are updating data or completing a financial transaction). The scope of this directive is PHP\_INI\_ALL.

PHP can highlight source code. One way is to give the script a .phps extension.

To use this, you need to add the following to httpd.conf:

AddType application/x=httpd-php-source .phps

38

Setting expose\_php to Off will prevent Apache from revealing the presence of PHP in its header signature (if there is one). 39

If you set ServerSignature to Off in httpd.conf, Apache will not broadcaast its server signature.

The max\_execution\_time setting (default 30) determines the maximum seconds a PHP script can run. Setting this to 0 will disable any time limit at all.

The memory\_limit parameter sets the maximum amount of memory a PHP script can be allocated. The default is 8M. 40

The register\_globals setting has been disabled by default since 4.2.0 for security reasons. If on, it would allow users to send information that should only be modified in cookies (like the session id) via URLs. 41

With this setting off, you can't use the global sessionid variable – you have to use $\_COOKIE['sessionid']. 41

The register\_globals feature is no longer available in PHP 6

42

It is recommended that you set register\_long\_arrays to Off (default is on) for performance reasons. This feature is no longer available in PHP 6. This registered input arrays like GET,POST,COOKIE,SYSTEM,ENVIRONMENT using the deprecated HTTP\_\*\_VARS syntax.

You can declare $argc and $argv variables to mimic argc and argv if you have set register\_argc\_argv to On (the default). This will allow you to access GET variables as elements of $argv.

I noticed $\_SERVER['argc'] and $\_SERVER['argv'] variables.

The post\_max\_size declaration will set an upper limit on the amount of POST data that can be submitted. The default is 8M

The magic quotes functions have been eliminated in PHP 6 because they have long been a source of confusion. You are better off using addslashes and stripslashes functions to escape quotes. 43

You can adjust the default MIME type returned by PHP scripts with the default\_mimetype directive. The default is text/html. You may need to alter this if you are using PHP to generate other content types (such as WML – Wireless Markup Language). The scope of this directive is PHP\_INI\_ALL. 44

Since PHP 4, PHP has sent CONTENT-TYPE headers of iso-8859-1. You can change this with default\_charset if you need to return Japanese or other such text. The scope is PHP\_INI\_ALL.

The include\_path parameter serves as the base path for functions such as include(), require(), and fopen\_with\_path(). You can specify multiple directories by using semicolons. The scope is PATH\_INI\_ALL. 45

By default this will be set the the path specified by the PHP\_INCLUDE\_PATH environment variable.

The extension\_dir parameter specifies where loadable extensions are located. 46

The enable\_dl parameter allows a user to load an extention at runtime.

The Fopen wrappers are five directives pertinent to accessing remote files.

allow\_url\_fopen allows PHP to treat remote files almost as if they were local. The default is On and scope is PHP\_INI\_ON. 47

The from string designates the password that will be used for the anonymous user used to perform FTP connections.

The extension directive is used to dynamically load a module. On Win32, it might look something like this:

extension = php\_java.dll

on Linux, it would look like this:

extension = php\_java.so

48

PDT (<http://www.eclipse.org/pdt/>) is the likely fron-runner to become the de facto PHP IDE for hobbyists and professionals alike. 49

Virtual private server hosting uses virtualization to give you your own OS on a shared server. 51

Most hosting providers offer automated installers for installing popular third-party software such as Joomla!, WordPress, and phpBB. 53

# 3: PHP Basics

There are four types of PHP delimiters. The most common is <?php ?>. 56

You should not use short-tags syntax because this conflicts with XML and thus XHTML.

57

Another delimiter offered (especially to get around Microsoft FrontPage problems) is script:

<script language = "php">

print "This is another PHP example;"

</script>

58

You can escape in and out of PHP code with multiple <?php ?> blocks. Variables declared in one code sequence are remembered in another – they are not in different scope. 59

PHP allows several commenting styles:

* Single line C++ style: //
* Shell syntax: #
* Multi-line C syntax: /\* \*/

phpDocumentor is an open source project that reads PHP source code and generates documentation from DocBlocks. It can generate several formats, including HTML and PDF. It is suggested that you become familiar with this type of thing. <http://www.phpdoc.org/>

60

The official syntax calls for the print statement to have parentheses, although they aren't required. It returns an integer. 62

The printf statement is ideal for building a mix of static and dynamic text. 63

printf ("Bar inventory: %d bottles of tonic water." ,100);

The %d is a placeholder known as a type specifier.

Commonly used type specifiers:

* %b:integer presented as binary number
* %c: Integer presented as an ASCII character
* %d: Integer
* %f:Floating point number
* %o:Integer presented as octal number
* %s: String
* %u: Integer presented as an unsigned number
* %x: Integer presented as a lowercase hexadecimal number
* %X: Integer presented as an upper case hexadecimal number

64

The sprintf is similar to printf except that the output is assigned to a string rather than rendered to the browser. 65

Scalar datatypes are capable of containing a single item of information (Boolean,integer,float,string).

0755 would be octal, 0xC4E hexadecimal. 66

Historically, PHP treated strings as arrays, allowing you to access characters as array positions.

Compound datatypes allow for multiple items of the same t ype to be aggregated under a single representative entity. 67

PHP supports two compound datatypes – array and object. 68

Unlike other datatypes in PHP, an object must be declared:

class Appliance{

private $\_power;

function setPower($status){

$this->\_power = $status;

}

}

$blender = new Appliance;

An object is instantiated using the new keyword.

You can convert values from one datatype to another by using type casting.

$score = (double) 13;

69

The string gettype(mixed var) function returns the type of the specified variable. There are eight possible return values:

* array
* boolean
* double
* integer
* object
* resource
* string
* unknown

71

The boolean settype(mixed var, string type) function will set the type of a variable.

There are functions such as is\_integer,is\_null, is\_array, etc. to determine if a variable is of a certain type. 72

If you want to use a dollar sign in quotes and have it taken as a literal dollar sign, you need to escape it with a backslash.

$dollarString="\$10.00";

A variable name always begins with a dollar sign. 74

Although not required, good programming practice dictates that all variables be declared before use, preferably with a comment.

You can assign values to variables either by value or by reference. 75

PHP 4 introduced assignment by reference. You can assign by reference by appending an ampersand to the equals sign:

<?php

$value1="Hello";

$value2 =& $value1;

$value2 = "Goodbye"; // $value1 and $value2 both equal "Goodbye"

?>

You can also use this syntax:

$value2 = &$value1;

76

There are four types of scope:

* Local variables
* Function parameters
* Global variables
* Static variables

You can use global variables within a function only if you have declared the variable as such with the GLOBAL keyword inside the function that will use it. 78

An alternative means is to use the $GLOBALS array. 79

A static variable will retain its value between function calls. It is declared using the STATIC keyword. 79

Static scoping is particularly useful for recursive functions. 80

PHP has useful superglobal variables such as $\_SERVER that will give you information about your operating environment and about your user. Some of what is available is dependent on OS and Web server. The following code will give you a listing:

foreach ($\_SERVER as $var => $value){

echo "$var => $value <br />";

}

You can display the user's IP address like this:

printf("Your IP address is: %s", $\_SERVER['REMOTE\_ADDR']);

82

Although the variables found in $\_SERVER vary by Web server and OS, you can typically expect to see those defined in the CGI 1.1 specification (<http://hoohoo.ncsa.uiuc.edu/cgi/env.html>).

Some of these include:

* HTTP\_REFERER (URL of the page that referred the user to the current location)
* REMOTE\_ADDR (client's IP address)
* REQUEST\_URI (path component of the URL)
* HTTP\_USER\_AGENT(information about the client's user agent)

83

The $\_GET superglobal contains any parameters passed by the GET method.

The $\_POST superglobal contains information passed via POST.

The $\_COOKIE superglobal contains information passed along as cookies (typically set using PHP's setcookie function).

The $\_ENV superglobal offers information regarding PHP's underlying server environment. 85

The $GLOBALS and $\_REQUEST superglobals should not be used because they are not secure (explained in chapter 21).

NOTE: On my system $\_ENV is empty.

The $\_SESSION superglobal contains information about all the session variables. 86

Using $$ for variable variables is covered on page 86.

The define function is used to define constants.

define ("PI", 3.141592);

Note that constants are not referenced with a dollar sign.87

Operands are the inputs of an expression.

An operator is a symbol that specifies a particular action in an expression. 88

The @ operator is the error suppression operator.

The period is the concatenation operator.

== means is equal to and === means is identical to.

PHP follows the standard precedence rules used in elementary school math class. 89

The multiplication operator has left-to-right associativity while the assignment operator has right-to-left associativity.

The .= operator is the concatenation-assignment operator. 90

Ternary:

($a == 12) ? 5 : -1;

If $a is equal to 12, return 5, otherwise return -1

93

You should not use the comparison operators to compare strings or anything other than numbers as you may get unexpected results. There are a substantial number of predefined functions for comparing strings.

You can learn a lot about binary encoding and bitwise operators and why they are important from Randall Hyde's massive online reference (<http://webster.cs.ucr.edu/>) 94

When strings are enclosed in double quotes, both variables and escape sequences are parsed. 95

When a string is included in single quotes, variables and escape sequences are NOT parsed. 96

You do need to escape the single quote character, however. 97

Heredoc syntax allows for outputting a large amount of text without using quotes.

<<<EXCERPT

...

EXCERPT;

Heredoc syntax follows the same parsing rules as double quoted strings except that you don't need to escape quotations. 98

The closing identifier must be flush left – it cannot be preceded by a space or have a space after it.

If (expression){

statement

}

99

The switch statement can be thought of as a variant of if-else.

switch($category){

case "news":

echo "News";

break;

case "weather":

echo "Weather";

break;

default:

echo "You did not make a choice";

}

101

while (expression){

statements

}

102

The do...while statement verifies the condition at the end of the loop so it will executed at least once:

do{

statements

}while(expression);

103

for (initial; condition; eachIteration){

statements

}

104

The foreach statement is good for looping through arrays. It has two syntax variables:

foreach (array\_expr as $value){

statement

}

105

The second system is suitable for working with key/value pairs in arrays:

foreach (array\_expr as $key => $value){

statement

}

106

A code example:

$links = array("The Apache Web Server" => "www.apache.org",

"Apress" => "www.apress.com",

"The PHP Scripting Language" => "www.php.net");

echo "<b>Online Resources</b>:<br />";

foreach ($links as $title => $link){

echo "<a href=\"http://$link\">$title</a><br />";

}

The break statement causes execution to jump outside the loop. The continue statement causes execution to jump back to the start of the loop. The goto statement was added to PHP 6 to support jumping to a label. 108

PHP offers four file-inclusion statements:

* include
* include\_once
* require
* require\_once

109

You can conditionally include files by using the include statement within a conditional block.

The code in the include file must be within a <?php ?> block, not just a collection of statements. 110

If the allow\_url\_fopen configuration directive is enabled and the URL wrapper is supported, you can include remote files like this (including GET parameters):

include "http://www.wjgilmore.com/index.html?background=blue";

There are two important differences between include and require:

* With require() the file will be included no matter what – even if it is in a conditional block that evaluates to false.
* The script will halt if the require() statement cannot include the file

111

# 4: Functions

More than 1000 functions are built into the standard PHP distribution. 113

Outputting a function in a larger string via concatenation:

echo "Five raised to the third power ewuals " . pow(5,3) . ".";

114

If you know the name of a function, you can go directly to its help page by appending it to the php.net URL:  
<http://www.php.net/pow>

function functionName(parameters)

{

function-body

}

115

By default, function arguments are passed by value rather than by reference. 116

You can have a parameter passed by reference by preceding it with an ampersand:

function calculateCost(&$cost, $tax){

...

}

117

My note: this also works if the signature parameter isn't defined by reference but the calling code uses the by reference syntax.

Default values can be assigned to function parameters:

function calcSalesTax($price, $tax=.0675){..}

118

Default values must be defined after variables without defaults and can only use constants (they can't be defined as function calls). You can make an argument optional by defining it as nothing:

function calcTaxOptional($price,$tax=""){...}

The following code skips the first optional parameter:

calculate (10, "", 3);

119

You can use the return statement to return a value from the function and terminate the function. If return is used in the global scope, the script ends.

$total = $price + ($price \* $tax);

return $total;

or

return $price + ($price \* $tax);

The list function can be used to set a list of variables from an array:

$colors = array("red","green","blue");

list($red,$green,$blue)=$colors;

function userProfile(){

$user[]="Howard";

$user[]="howard@example.com";

$user[]="Programmer";

return $user;

}

list($name,$email,$title) = userProfile();

echo $green . "<br />";

echo $email;

120

This feature is quite useful and will be used often throughout the book. 121

Recursive functions (functions that call themselves) have considerable practical value and are used to divide an otherwise complex problem into a simple case. The usual examples are with factorials. In this case, we use a more practical example that builds an amortization table:

<?php

function amortizationTable($paymentNum, $periodicPayment, $balance,

$monthlyInterest) {

$paymentInterest = round($balance \* $monthlyInterest,2);

$paymentPrincipal = round($periodicPayment - $paymentInterest,2);

$newBalance = round($balance - $paymentPrincipal,2);

print "<tr>

<td>$paymentNum</td>

<td>\$".number\_format($balance,2)."</td>

<td>\$".number\_format($periodicPayment,2)."</td>

<td>\$".number\_format($paymentInterest,2)."</td>

<td>\$".number\_format($paymentPrincipal,2)."</td>

</tr>";

# If balance not yet zero, recursively call amortizationTable()

if ($newBalance > 0) {

$paymentNum++;

amortizationTable($paymentNum, $periodicPayment, $newBalance,

$monthlyInterest);

} else {

exit;

}

} #end amortizationTable()

# Loan balance

$balance = 200000.00;

# Loan interest rate

$interestRate = .0575;

# Monthly interest rate

$monthlyInterest = .0575 / 12;

# Term length of the loan, in years.

$termLength = 30;

# Number of payments per year.

$paymentsPerYear = 12;

# Payment iteration

$paymentNumber = 1;

# Perform preliminary calculations

$totalPayments = $termLength \* $paymentsPerYear;

$intCalc = 1 + $interestRate / $paymentsPerYear;

$periodicPayment = $balance \* pow($intCalc,$totalPayments) \* ($intCalc - 1) /

(pow($intCalc,$totalPayments) - 1);

$periodicPayment = round($periodicPayment,2);

# Create table

echo "<table width='50%' align='center' border='1'>";

print "<tr>

<th>Payment Number</th><th>Balance</th>

<th>Payment</th><th>Interest</th><th>Principal</th>

</tr>";

# Call recursive function

amortizationTable($paymentNumber, $periodicPayment, $balance, $monthlyInterest);

# Close table

print "</table>";

?>

121

The author then showed an example of setting up a PHP library with tax related functions in it called taxes.library.php. He noted that you should not use an extension that would return unparsed text as a user could browse to this file and see sensitive code. 124

# 5: Arrays

An array is an ideal way to store, manipulate, sort, and retrieve data sets. 127

An array has been traditionally defined as a group of items that share certain characteristics, including datatype. Each item is distinguished by a special identifier known as a key. PHP forgoes the requirement that all the items must be of the same datatype. 128

The keys can be numerical or associative.

$states = array(0 => "Alabama", "1" => "Alaska");

$states = array("OH" => "Ohio", "PA" => "Pennsylvania");

You can also create multidimensional arrays:

$states=array(

"Ohio" => array("population" => "11,353,140", "capital" => "Columbus"),

"Nebraska" => array("population" => "1,711,263","capital" => "Omaha")

);

echo $states["Ohio"]["population"];

129

Built-in array functions all rely on using a central feature known as an array pointer.

Unlike other languages, PHP does not require you to set the size of an array or even create it in advance. You can create an array with the following syntax:

$state[0] = "Delaware";

If the array is numerical, you can add a new element without specifying the key:

$state[] = "New Jersey";

130

The array construct is a more formal way to create an array:

array array([item1 [,item2 ... [, itemN]]])

The list construct makes simultaneous variable assignments from values extracted from an array. 131

void list(mixed...)

This construct is particularly useful when extracting information from a database or file.

Here is an example extracting data from a text file delimited by |:

$users = fopen("users.txt");

while($line = fgets($users,4096)){

list($name,$occupation,$color) = explode("|",$line);

printf("Name: %s <br />",$name);

printf("Occupation: %s <br />", $occupation);

printf("Favorite color: %s <br />",$color);

}

fclose($users);

The range function allows you to fill an array with a range of values. 132

The following would give you an array of all the possible values of a die:

$die = range(0,6);

You can also use range for character sequences and specify a step value. 133

The print\_r function will output the contents of an array or object in a readable fashion.

If you send TRUE as the optional second parameter to print\_r the output will be returned to the caller rather than stdout.

The is\_array function will tell you if a variable is an array or not. 134

PHP has several functions for adding and removing array elements:

* array\_push
* array\_pop
* array\_shift
* array\_unshift

A traditional queue is a data structure employing first-in-first-out (FIFO). A stack is a data structure using last-in-first-out (LIFO).

The array\_unshift function adds elements to the front of an array:

array\_unshift($states,"California","Texas");

135

You can use array\_push to add values to the end of an array:

array\_push($states,"California","Hawaii");

The array\_shift function returns and removes an item from the front of an array. The array\_pop function returns and removes an item from the end of an array.

Thus you would use array\_shift for a queue and array\_pop for a stack.

The in\_array function searches for a value in an array. 136

The array\_key\_exists function looks for a key in an associative array.

The following will look for Ohio in an array and print the admission date if it exists:

if (array\_key\_exists("Ohio",$states)){

printf("Ohio joined the union on %s", $states["Ohio"]);

}

137

The array\_search function searches an associative array for a value and returns its key if found, otherwise FALSE.

The array\_keys function returns an array of all the keys found in the array.

The array\_values function will return all the values from an array. 138

The key function returns the key at the current pointer for the specified array. 139

The following code uses key to list all the keys in an array:

$capitals = array("Ohio" => "Columbus", "Iowa" => "Des Moines",

"Arizona" => "Phoenix");

echo "<p>Can you name the capitals of these states?</p>";

while($key = key($capitals)) {

echo $key."<br />";

next($capitals);

}

this prints:

Ohio

Iowa

Arizona

Note that the key function does not advance the array pointer. You use the next function to move the array pointer.

The current function returns the value at the current pointer in the array.

The each function returns an array of the current keys and values from an array and advances the pointer by one. FALSE is returned if at the end of the array. The each function will return an array of 4 with numeric and associative keys. 140

The next and prev functions move the array pointer forward or backward and return the value.

The reset function moves the pointer back to the start of the array. The end function moves to the last position of the array. 141

The array\_walk function passes each element of an array to the user specified function. 142

The user defined function must two (or three) parameters. The first parameter is the current value of the array and the second is the current key.

The following shows how you can use array\_walk as a "sanity checker" for form data, stripping tags from each parameter before entering in a database:

<form action="submitdata.php" method="post">

<p>

Provide up to six keywords that you believe best describe the state in

which you live:

</p>

<p>Keyword 1:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p>Keyword 2:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p>Keyword 3:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p>Keyword 4:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p>Keyword 5:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p>Keyword 6:<br />

<input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>

<p><input type="submit" value="Submit!"></p>

</form>

<?php

function sanitize\_data(&$value, $key) {

$value = strip\_tags($value);

}

array\_walk($\_POST['keyword'],"sanitize\_data");

?>

The count function will return the number of values in the array. 143

The sizeof function is an alias of count and is identical. 144

The array\_count\_values returns an associative array with unique values as keys and the count of their occurrence as values. 144

The array\_unique function returns an array of unique values with all duplicates removed. 145

The array\_reverse function reverses an array's element order. 146

The array\_flip function reverses the roles of keys and values in an array.

The sort function sorts an array in place. It changes the array and always returns void regardless of outcome. 147

The optional sort\_flags parameter may have one of these values:

* SORT\_NUMERIC
* SORT\_REGULAR
* SORT\_STRING

The natsort function sorts an array according to how humans would sort it. It would, for example, sort like this: picture8.jpg,picture9.jpg,picture10.jpg. 150

The natcasesort is the same as natsort except that it is case insensitive.

The ksort function sorts an array by its keys. 151

The usort function allows you to sorrt an array by a user-defined comparison algorithm. It's prototype is:

void usort(array array, callback function\_name)

151

The array\_merge function combines multiple arrays into a single array. 153

If the same associative key already exists, it will be overwritten by the newer element. The array\_merge\_recursive function will instead create an array within the array for keys of the same name.

The array\_combine function will produce a new array consisting of a submitted set of keys and corresponding values. It's prototype:

array array\_combine(array keys, array values)

155

The array\_slice function returns a portion of an array based on start and offset values.

The array\_splice function removes all elements of an array found within a specified range, returning an array of those items removed. 156

The array\_intersect function returns a key-preserved array consisting only of those values present in the first array that are also present in each of the other input arrays.

157

The array\_intersect\_assoc function returns only matches where key/value pairs are included in all the arrays. 158

The array\_diff function returns only those values in the first array that are not contained in any of the other arrays. 159

The array\_diff\_assoc function considers key/value pairs not in the other arrays.

The array\_rand function will return a random number of keys. It's prototype:

mixed array\_rand(array array [, int num\_entries])

160

The shuffle function randomly reorders an array.

The array\_sum function adds all the values of the array. 161

The array\_chunk function breaks an array into a multidimensional array or smaller arrays.

<?php

$cards = array("jh","js","jd","jc","qh","qs","qd","qc",

"kh","ks","kd","kc","ah","as","ad","ac");

// shuffle the cards

shuffle($cards);

// Use array\_chunk() to divide the cards into four equal "hands"

$hands = array\_chunk($cards, 4);

print\_r($hands);

?>

162

# 6: Object-Oriented PHP

Early attempts at incorporating OOP into PHP (version 4) were poorly implemented. PHP 5 substantially improved OOP. 163

The author recommends PHP Objects, Patterns, and Practice Second Edition (Apress 2007) by Matt Zandstra. 164

OOP shifts focus from logic to data – from a program's features to the real-life entities it models.

OOP has three foundational concepts:

* Encapsulation
* Inheritance
* Polymorphism

The practice of separating the user from the true inner workings of an application through interfaces is known as encapsulation.

The developer can change the implementation of an object without the user being affected because the interface does not change. 165

Polymorphism (Greek for having multiple forms) defines OOP's ability to redefine a class's characteristics (morph) or behavior depending on the context in which it is used. 166

(This has to do with being able to override methods in a subclass. I would say it also has to do with implementing interfaces).

A class is a template that defines the attributes and behaviors of an entity.

In OOP parlance, attributes and behaviors are known as fields and methods.

While no official PHP code conventions exist, you should consider following the PHP Extension and Application Repository guidelines ([http://pear.php.net](http://pear.php.net/)). These conventions are used in this book.

167

Objects are created using the new keyword:

$employee = new Employee();

168

Although not required because PHP is loosely typed, it is common practice to declare fields at the beginning of the class.

It is common practice to declare fields with a scope descriptor (public,private,protected).

Fields are referred to with the -> operator (without the dollar sign):

$object->field

When referencing a field from within the object or class, use the $this keyword:

$this->name = "Howard";

169

PHP supports five class field scopes:

* public
* private
* protected
* final
* static

Public fields are generally considered to be taboo as it robs the developer of controls such as data validation. 170

As of PHP 6, you can use var instead of public.

Private fields are only accessible from within the class in which they were defined. They are not directly accessible by the objects instantiated from the class. 171

Private fields are also not available to subclasses. If you want such fields to be available, make them protected.

A protected field is available to inherited classes but not objects. If you plan to extend a class you should make the fields protected rather than private. 172

You can declare fields or methods as final, meaning they can't be overridden by a subclass.

PHP does not have the property functions that some other OOP languages have – you can simulate this via getter and setter methods.

The mutator or setter method is responsible for both hiding property assignment implementation and validating class data before assigning it to a class field. It's prototype folloes:

boolean \_\_set([string property name],[mixed value to asign])

173

Example:

<?php

class Employee{

var $name;

function \_\_set($propName, $propValue){

echo "Nonexistent variable: \$$propName!";

}

}

$employee = new Employee();

$employee->name ="Mario";

$employee->title="Executive Chef";

?>

results in:

Nonexistent variable: $title!

You could use the \_\_set function to extend the class with new properties like this:

$this->$propName = $propValue;

174

The \_\_set and \_\_get methods aren't really sufficient for managing properties in a complex object-oriented project. You are better off creating getter and setter methods for each private field.

175

Class constants are created like this:

const NAME = 'VALUE';

176

You can call a class constant like this:

echo Myclass::MYCONSTANT;

177

A method is similar to a function except that it defines the behavior of a class.

Methods are declared using the same syntax as functions except that they use a scope descriptor.

PHP supports six methods scopes:

* public
* private
* protected
* abstract
* final
* static

178

Private methods can only be called by the class – they can't be called by subclasses.

Class methods marked as protected can only be called by the class or subclassses. 179

Abstract methods can only be declared within the class and must be implemented by subclasses. Only an abstract class can have an abstract method. 180

abstract class Employee {

abstract function hire();

abstract function fire();

abstract function promote();

abstract function demote();

}

181

A final method cannot be overridden by a subclass.

Type hinting is a feature added to PHP 5.

private function takeLunchbreak(Employee $employee){...}

182

Type hinting only works with objects and arrays – you can't use type hints with integers, strings, etc.

Special methods called constructors and destructors automate object creation and destruction.

PHP 4 constructors were simply methods with the same name as the class, which made it difficult to rename classes. PHP 5 changed this. For backward compatibility, however, if a class does not contain a constructor with the new naming method a method with the same name as the class will be searched for.

183

PHP uses the \_\_construct function name:

function \_\_construct([argument1, argument2, ..., argumentN]){...}

PHP does not automatically call the parent constructor – you must manually invoke it with the parent keyword. 184

<?php

class Staff

{

protected $name;

protected $title;

function \_\_construct()

{

echo "<p>Staff constructor called!</p>";

}

}

class Manager extends Staff

{

function \_\_construct()

{

**parent::\_\_construct();**

echo "<p>Manager constructor called!</p>";

}

}

$employee = new Manager();

?>

185

You can call the constructor of one class from another even if they are not related. Use the syntax Classname::construct(). 186

PHP does not support constructor overloading like other OOP languages do.

Use the function name \_\_destruct to create a destructor function.

Static members of a class are shared by all instances of the class. A hit counter for a Web page would be an example of this.

Static members are referred to using the self keyword rather than $this. The syntax is as follows:

<?php

class visitors

{

private static $visitors = 0;

function \_\_construct()

{

self::$visitors++;

}

static function getVisitors()

{

return self::$visitors;

}

}

/\* Instantiate the visitors class. \*/

$visits = new visitors();

echo visitors::getVisitors()."<br />";

/\* Instantiate another visitors class. \*/

$visits2 = new visitors();

echo visitors::getVisitors()."<br />";

?>

188

The instanceof keyword was introduced in PHP 5. It lets you determine if an object is an instance or a class or an instance of a subclass of that class or implements a specified interface. 189

if($manager instanceof Employee) echo "Yes";

There are a number of helper functions to help manage classes.

The class\_exists function will tell you if a given class is exists within the running script. The get\_class function returns the name of the class an object belongs to or FALSE if it isn't an object.

The get\_class\_methods function returns an array of all the method names define by a class. 190

The get\_class\_vars function returns an associative array with all the variable names and their values defined in the class.

The get\_declared\_classes function returns an array of all the classes defined within the current script.

The get\_object\_vars method will return an associative array with all the variable names and their values for the specified object (NULL will be assigned to variables with no value).

The get\_parent\_class function returns the name of the parent class of an objecct (or the base name if it is a base class).

The interface\_exists function determines if an interface exists.

The is\_a function returns true if the object belongs to the specified class or is a child of that class. 191

The is\_subclass\_of function determines if an object belongs to a subclass of the specified class.

The method\_exists function determines if the specified object has a a method by the specified name.

It's common practice to put each class in a separate file for organizational reasons. The author uses a naming convention like MyClass.class.php

To get around the need to use require\_once statements for each class, PHP 5 introduced the concept of autoloading objects. With autoloading you can define a \_\_autoload function that is called whenever a class is referenced that wasn't defined in the script:

function \_\_autoload($class){

require\_once("classes/$class.class.php");

}

192

# 7: Advanced OOP Features

The following features will be covered in this chapter:

* Object cloning
* Inheritance
* Interfaces
* Abstract classes
* Namespaces

193

One of the major improvements made in PHP 5 was that all objects are passed by reference by default. Cloning allows you to get around this and makes it easier to copy an object when you don't want to treat it by reference.

An interface is a collection of unimplemented method definitions and constants that serves as a class blueprint.

The namespace feature is new to PHP 6.

All the features described in this chapter are only available in PHP 5 or above.

194

PHP 4's treatment of objects as just another data type (rather than references) was one of its major drawbacks, making it hard to implement things like design patterns.

You can explicitly make a copy of an object using the clone keyword:

destinationObject = clone sourceObject;

195

You can define a \_\_clone method in a class that will be executed whenever the object is cloned. 196

Class inheritance is accomplished by using the extends keyword. 199

If the parent class has a constructor, it will execute provided that the subclass does not also have a constructor. 201

If you want the parent constructor to fire in a subclass that has its own constructor, you have to call it explicitly:

parent::\_\_construct($name);

202

When parent::\_\_construct is called, PHP will search upward in the class hierarchy until a class with a constructor is found. 203

You can also reference a constructor explicitly by class name:

function \_\_construct($name){

Employee::\_\_construct($name);

Executive::\_\_construct();

echo "<p>CEO object created!</p>";

}

An interface defines a general specification for implementating a particular service, declaring required methods and constants without specifying how they are implemented.

PHP 5 implements interfaces like this:

interface InterfaceName

{

CONST 1;

CONST N;

function methodName1();

function methodName2();

}

204

It's common practice to preface the names of interfaces with the letter I to make them easier to recognize.

A class implements an interface using the implements keyword. All methods of the interface must be implemented or else the class must be declared as an abstract class.

You can implement multiple interfaces by separating their names with commas. 206

An abstract class is a class that is not meant to be instantiated but instead serves as a base class for other classes. 207

# 8: Error and Exception Handling

PHP has several features for handling errors:

* Configuration directives
* Error logging
* Exception handling

213

Exception handling was introduced in PHP 5.

The error\_reporting directive determines the error reporting sensitivity level. There are 14 levels. 214

During the development state, you'll probably want to set this to E\_ALL. 215

You can use logical directives like OR or NOT (~) to set this directive:

error\_reporting = E\_ERROR | E\_PARSE | E\_CORE\_ERROR

Enabling display\_errors will cause errors to be displayed in the browser.

The display\_startup\_errors directive determines if errors are displayed when PHP starts up. You should have this disabled when your site goes live. 216

You should keep log\_errors enabled at all times. Where the log statements are recorded depends on error\_log directive.

If this is set to syslog, errors are sent to the system log.

Enabling track\_errors causes the most recent error message to be stored in the $php\_errormsg variable. 217

PHP also allows you to send custom messages to the system syslog. 218

You need to call the define\_syslog\_variables function before you can call any of the other logging functions.

The openlog function opens a connection to the platform's system logger:

int openlog(string ident, int option, int facility)

The system log is closed with closelog. Use the syslog function to send a message to the log.

int syslog(int priority, string message)

219

The following will write a warning to the log (under windows, this will go to the application section of event viewer):

<?php

define\_syslog\_variables();

openlog("CHP8",LOG\_PID,LOG\_USER);

syslog(LOG\_WARNING,"Chapter 8 example warning.");

closelog();

?>

220

Exception handling is new in PHP 5. 221

Exception handling syntax promotes the separation of error handlers from the general application logic, resulting in more organized, readable code.

Almost all languages have borrowed from the C++ language's try/catch syntax. 222

Currently PHP only offers a single handler called Exception, but you can extend this with your own custom exception classes.

You can code for multiple exceptions by coding multiple catch blocks.

throw new Exception();

223

The overloaded constructor for Exception takes two parameters – message and error code. 224

throw new Exception("Something bad just happened",4);

throw new Exception("Something bad just happened");

throw new Exception("",4);

There are six methods available to the base Exception class:

* getMessage()
* getCode()
* getLine()
* getFile()
* getTrace()

These methods are declared as final so you can't override them.

The following code extends the Exception class to have error messages based on language (a mapping file is used to map error codes to their language-specific string):

<?php

class MyException extends Exception {

function \_\_construct($language,$errorcode) {

$this->language = $language;

$this->errorcode = $errorcode;

}

function getMessageMap() {

$errors = file("errors/".$this->language.".txt");

foreach($errors as $error) {

list($key,$value) = explode(",",$error,2);

$errorArray[$key] = $value;

}

return $errorArray[$this->errorcode];

}

} # end MyException

try {

throw new MyException("english",4);

}

catch (MyException $e) {

echo $e->getMessageMap();

}

?>

226

The following code illustrates catching multiple exceptions. The user can submit an HTML form with an email address. This accounts for no email address, improper email address, SQL injection, etc.

<?php

/\* The InvalidEmailException class is responsible for notifying the site

administrator in the case that the e-mail is deemed invalid. \*/

class InvalidEmailException extends Exception {

function \_\_construct($message, $email) {

$this->message = $message;

$this->notifyAdmin($email);

}

private function notifyAdmin($email) {

mail("admin@example.org","INVALID EMAIL",$email,"From:web@example.com");

}

}

/\* The subscribe class is responsible for validating an e-mail address

and adding the user e-mail address to the database. \*/

class subscribe {

function validateEmail($email) {

try {

if ($email == "") {

throw new Exception("You must enter an e-mail address!");

} else {

list($user,$domain) = explode("@", $email);

if (! checkdnsrr($domain, "MX"))

{

throw new InvalidEmailException("Invalid e-mail address!", $email);

} else {

return 1;

}

}

} catch (Exception $e) {

echo $e->getMessage();

} catch (InvalidEmailException $e) {

echo $e->getMessage();

}

}

/\* This method would presumably add the user's e-mail address to

a database. \*/

function subscribeUser() {

echo $this->email." added to the database!";

}

} #end subscribe class

/\* Assume that the e-mail address came from a subscription form. \*/

$\_POST['email'] = "someuser@example.com";

/\* Attempt to validate and add address to database. \*/

if (isset($\_POST['email'])) {

$subscribe = new subscribe();

if($subscribe->validateEmail($\_POST['email']))

$subscribe->subscribeUser($\_POST['email']);

}

?>

227

Note that an exception from the base class can occur (it is checked for first) or an exception from our custom class. 228

# 9: Strings and Regular Expressions

PHP supports two regular expression implementations: POSIX and Perl. 231

The PEAR Validate\_US package can be used for validating the syntax for phone numbers, SSNs, ZIP codes, and state abbreviations.

POSIX stands for Portable Operating System Interface for Unix. 232

You can test for several expressions simultaneously with |. 233

POSIX regular expressions has three methods for locating character sequences:

* brackets
* quantifiers
* predefined character ranges

The string php will find php. The sequence [php] will match anything containing p or h.

[A-Za-z] matches any alphabetical character.

Quantifiers are used to match a quantity of characters or their position.

* p+ matches any string containing at least 1 p
* p\* matches anything with 0 or more p
* p? matches 0 or 1 p
* p{2} matches any string with a sequence of 2 p
* p{2,3} matches anything with 2 or 3 p
* p{2,} matches at least 2 p
* p$ any string with p at the end
* ^p matches any string with p at the beginning
* [^p-zA-Z] matches any string *not* containing and of the chacacters.
* p.p matches a string with p followed by any character followed by p
* ^.{2}$ matches any string containing exactly two characters
* <b>(.\*)</b> matches any string enclosed in <b></b>
* p<hp)\* matches any strong with a p followed by 0 or more instances of hp.

Use the backslash as an escape character if you want to search for one of the special characters. 234

Several predefined character ranges, also called character classes, are available:

* [:alpha:] lower or upper case letters
* [:alnum:] alpha numeric
* [:cntrl:] control characters such as tab, escape, or backspace
* [:digit:] 0-9
* [:graph:] Printable characters from ASCII 333-126
* [:lower:]
* [:punct:] ~`@#$%^&\*()-\_+={}[]:;'<>,./
* [:upper:]
* [:space:] whitespace characters
* [:xdigit:] hesxadecimal characters

235

My example:

\((.\*)\) finds everything enclosed in parentheses (including the parentheses). Sometimes gets more than I want.

PHP offers the following POSIX extended regular expression functions:

* ereg
* ereg\_replace
* eregi
* eregi\_replace
* split
* spliti
* sql\_regcase

235

The ereg function performs a case sensitive regular expression search, retuning TRUE if the pattern was found.

boolean ereg(string pattern,string string [,array regs]{

You could use the following to determine if a username is all lowercase:

if (ereg("([^a-z])",$username))

The optional regs argument will contain an array of all matching parts in parentheses. 236

The following pattern looks for an alphanumeric string from 8-10 characters:

^[a-zA-Z0-9]{8,10}$

237

My example: find 5 letter words:

^[:alnum:]{5}$

ereg\_replace replaces the matched strings with the string provided.

The php ereg functions allow you to back-reference substrings in parentheses using \0 \1 \2, etc. The \0 references the entire string. 238

str\_replace is much faster than ereg\_replace when there aren't complex regex expressions invovled.

The split function breaks a string into various elements based on the pattern string:

array split(string pattern, string string [, int limit])

The following will break a string apart based on horizontal tabs and newline characters:

split("[\n\t]", $test));

The sql\_regcase function can be used to create case-insensitive regex patterns for implementations that don't support case insensitive searches (p would be converted to [Pp]. 239

Perl has long been considered one of the most powerful parsing languages ever written. PHP supports Perl regular expressions.

Perl regex is a derivation of the POSIX implementation.

Perl regular expressions use forward slashes (/) to enclose the expression. 240

The following matches f followed by 2-4 occurrences of o:

/fo{2,4}/

You can use modifiers to tweak the pcre – telling it to perform a case-insensitive search, ignore comments embedded in the expression, etc.

Here are some common modifiers:

* i (case-insensitive)
* g (global. Find all occurances)
* m (treat string as multiple lines, allowing ^ and $ to match start and end of any line)
* s (treat string as single line)
* x (ignore white space and comments within the regex)
* U (stop at first match)

These modifiers are placed after the trailing forward slash.241

A metacharacter is an alphabetical character preceded by a backslash that symbolizes a special meaning. Here are a few useful ones:

* \A (match only at beginning)
* \b (match a word boundary)
* \B (match anything but a word boundary)
* \d (digit character)
* \D (non-digit character)
* \s (whitespace character)
* \S (non-whitespace character)
* [] (encloses a character class)
* $ (matches the end of the line)
* ^ (matches the beginning of the line)
* . (any character except for the newline)
* \ (quotes the next metacharacter)
* \w (any string with solely alphanumeric or underscore. Same as [a-zA-Z0-9\_])
* \W (string omitting alphanumeric and underscore)

The following will patch pisa and lisa but not sand:

/sa\b/

242

The following will return the first case-insensitive occurrence of the word linux:

/\blinux\b/i

Finally, this one returns all instances of a string matching a dollar sign followed by one or more digits:

/\$\d+/g

The following are the PHP PCRE functions:

* preg\_grep
* preg\_match
* preg\_match\_all
* preg\_quote
* preg\_replace
* preg\_replace\_callback
* preg\_split

The prototype for preg\_grep is:

array preg\_grep(string pattern, array input [,flags])

The following example looks for items beginning with p:

<?php

$foods = array("pasta","steak","fish","potatoes");

$food= preg\_grep("/^p/",$foods);

print\_r($food);

?>

returns:

Array ( [0] => pasta [3] => potatoes )

If you want to remove instances of the array that are blank, filter it through array\_values(). 243

The flags parameter, introduced in 4.3, accepts one value: PREG\_GREP\_INVERT. This would caluse the array to be filled with items that *don't* match.

The preg\_match function searches a string for a specific pattern and returns TRUE if a match is found. The prototype is:

int preg\_match(string pattern, string string [, array matches]

[, int flags [, int offset]]])

Example:

<?php

$line= "vim is the greatest word processor ever created!";

if (preg\_match("/|bVim\b/i",$line, $match)) print "Match found!";

?>

The preg\_match\_all function matches all occurances of a pattern in a string:

int preg\_match\_all(string pattern, string string, array pattern\_array

[, int order])

The default order is that $pattern\_array[0] will contain an array of all complete pattern matches, $pattern\_array[1] is an array of all matches to the first parenthesized regexp, and so on. 244

The following finds all strings enclosed in bold HTML tags:

<?php

$userinfo = "Name: <b>Zeev Suraski</b> <br> Title: <b>PHP Guru</b>";

preg\_match\_all ("/<b>(.\*)<\/b>/U", $userinfo, $pat\_array);

print $pat\_array[0][0]." <br> ".$pat\_array[0][1]."\n";

?>

returns:

**Zeev Suraski**

**PHP Guru**

244

The preg\_quote function inserts a backslash in front of every character that is considered to be a special character.

The preg\_replace function replaces all occurrences of pattern with replacement:

mixed preg\_replace(mixed pattern, mixed replacement, mixed str [, int limit])

245

The pattern and replacement patterns can be arrays.

The preg\_split function operates exactly like split except that the pattern can be defined in terms of a regular expression. 247

<?php

$delimitedText = "+Jason+++Gilmore+++++++++++Columbus+++OH";

$fields = preg\_split("/\+{1,}/", $delimitedText);

foreach($fields as $field) echo $field."<br />";

?>

returns:

Jason

Gilmore

Columbus

OH

248

In addition to the regular expression based string functions, PHP offers more than 100 functions capable of manipulating every aspect of a string.

The strlen function counts the number of characters in a string. 249

PHP has four functions for string comparison:

* strcmp
* strcasecmp
* strspn
* strcspn

The strcmp function returns one of three values:

* 0 if the strings are equal
* -1 if string 1 is less than string 2
* 1 is string 1 is greater than string 2

250

The strspn function returns the length of the first segment in a string constianing characters also found in another string. The following code uses this to ensure that a password does not consist entirely of numbers:

<?php

$password = "3312345";

if (strspn($password, "1234567890") == strlen($password))

echo "The password cannot consist solely of numbers!";

?>

251

The strcspn function returns the length of the first segment that does not contain characters in the second string.

There are four functions available for case conversion:

* strtolower
* strtoupper
* ucfirst
* ucwords

252

There are several functions for converting strings to and from HTML. 254

The nl2br function converts all newline characters to <br />.

The htmlentities function will convert entities like the copyright character to &copy . 255

The htmlspecialchars function will turn & into &amp;,< inti &lt;, " inti &quot; and ' inti &#039;, etc. 257

My note: htmlspecialchars\_decode will convert special character back.

This is especially useful for converting potentially harmful HTML text entered into message boards, etc.

The strip\_tags function just removes all tags.

If you are going to use htmlspecial chars and nl2br together, use nl2br last or else the <br /> will be converted to something visible.

258

The get\_html\_translation\_table function returns the specified translation table (example: HTML\_ENTITIES). You can use this returned array with the strtr function to build you own custom translation table (to convert <b> to <strong>).

String strtr(string str, array replacements)

259

<?php

$table = array("<b>" => "<strong>", "</b>" => "</strong>");

$html = "<b>Today In PHP-Powered News</b>";

echo strtr($html, $table);

?>

The strip\_tags function will remove all HTML tags from a string:

string strip\_tags(string str [, string allowable\_tags])

The fgetss function performs a similar feature with a file.

Regular expressions can be slow. There are numerous functions that can be used instead of regex when the conversion isn't that complicated. 260

The strtok function tokenizes a string. It must be called repeatedly until the tokenization is complete, but the string argument only needs to be specified once.

String strtok(string str, string tokens)

<?php

$info = "J. Gilmore:jason@example.com|Columbus, Ohio";

// delimiters include colon (:), vertical bar (|), and comma (,)

$tokens = ":|,";

$tokenized = strtok($info, $tokens);

// print out each element in the $tokenized array

while ($tokenized) {

echo "Element = $tokenized<br>";

// Don't include the first argument in subsequent calls.

$tokenized = strtok($tokens);

}

?>

produces:

Element = J. Gilmore

Element = jason@example.com

Element = Columbus

Element = Ohio

261

The explode function defines a string into an array of substrings:

array explode(string separator, string str [, int limit])

The following example uses explode to count the number of words in a block of text:

<?php

$summary = <<< summary

In the latest installment of the ongoing Developer.com PHP series,

I discuss the many improvements and additions to

<a href="http://www.php.net">PHP 5's</a> object-oriented architecture.

summary;

$words = sizeof(explode(' ',strip\_tags($summary)));

echo "Total words in summary: $words";

?>

262

You can form a delimited string from an array with the implode function:

string implode(string delimiter, array pieces)

The strpos function finds the first position of the first occurance of one string within another:

int strpos(string str, string substr [, int offset])

The following finds the timestamp for the first occurrence of index.html:

<?php

$substr = "index.html";

$log = <<< logfile

192.168.1.11:/www/htdocs/index.html:[2006/02/10:20:36:50]

192.168.1.13:/www/htdocs/about.html:[2006/02/11:04:15:23]

192.168.1.15:/www/htdocs/index.html:[2006/02/15:17:25]

logfile;

// what is first occurrence of the time $substr in log?

$pos = strpos($log, $substr);

// Find the numerical position of the end of the line

$pos2 = strpos($log,"\n",$pos);

// Calculate the beginning of the timestamp

$pos = $pos + strlen($substr) + 1;

// Retrieve the timestamp

$timestamp = substr($log,$pos,$pos2-$pos);

echo "The file $substr was first accessed on: $timestamp";

?>

263

The strrpos function finds the last occurance of a string.264

The following function uses strrpos to truncate a lengthy article into a summary on the last word boundary at the desired length:

<?php

// Limit $summary to how many characters?

$limit = 100;

$summary = <<< summary

In the latest installment of the ongoing Developer.com PHP series,

I discuss the many improvements and additions to

<a href="http://www.php.net">PHP 5's</a> object-oriented

architecture.

summary;

if (strlen($summary) > $limit)

$summary = substr($summary, 0, strrpos(substr($summary, 0, $limit),

' ')) . '...';

echo $summary;

?>

The str\_replace function replaces all occurrences of one string with another. 265

If no match is found, the original string is returned, otherwise the modified string is returned.

The strstr function returns the remainder of a string beginning with the first occurrence of a predefined string.

The following example returns the domain name from an email address:

<?php

$url = "sales@example.com";

echo ltrim(strstr($url, "@"),"@");

?>

266

The substr function returns a portion of a string:

string substr(string str, int start [, int length])

If start is negative, the function will begin from the end of the string. If length is negative, the returned string will end length characters from the end of the string.

The substr\_count function returns the number of times one string occurs within another. 267

The following example counts the number of times certain buzzwords occur:

<?php

$buzzwords = array("mindshare", "synergy", "space");

$talk = <<< talk

I'm certain that we could dominate mindshare in this space with our new product,

establishing a true synergy between the marketing and product development teams.

We'll own this space in three months.

talk;

foreach($buzzwords as $bw) {

echo "The word $bw appears ".substr\_count($talk,$bw)." time(s).<br />";

}

?>

268

The substr\_replace function replaces occurances of one string with another:

string substr\_replace(string str, string replacement, int start [,int length])

The following example shows only the last four numbers of a credit card number:

<?php

$ccnumber = "1234567899991111";

echo substr\_replace($ccnumber,"\*\*\*\*\*\*\*\*\*\*\*\*",0,12);

?>

269

The ltrim function trims characters from the beginning of a string (including horizontal tab \t, \n,or vertical tab \x0b and NULL \0):

string ltrim(string str, [,string charlist])

rtrim trims from the right and trim trims from both ends. 270

The str\_pad function is used to pad a string.

The following will pad a string with designated characters at both ends:

$header="Log Report";

echo str\_pad($header,20,"=+",STR\_PAD\_BOTH);

The count\_chars function provides information on the number of characters in a string:

mixed count\_chars(string str [,mode])

271

<?php

$sentence = "The rain in Spain falls mainly on the plain";

// Retrieve located characters and their corresponding frequency.

$chart = count\_chars($sentence, 1);

foreach($chart as $letter=>$frequency)

echo "Character ".chr($letter)." appears $frequency times<br />";

?>

The str\_word\_count function returns information about the number of words:

mixed str\_word\_count(string str [, int format])

272

If format is not specified, the word count is retrurned. If format is 1, an array of all words found in the string is returned. If format is 2, an array is returned where key is the numerical postion of the word and value is the word itself.

You can use this function in conjunction with array\_count\_values to determine the frequency of each word within a string:\

<?php

$summary = <<< summary

In the latest installment of the ongoing Developer.com PHP series,

I discuss the many improvements and additions to PHP 5's

object-oriented architecture.

summary;

$words = str\_word\_count($summary,2);

$frequency = array\_count\_values($words);

print\_r($frequency);

?>

273

There is a PEAR package called Validate\_US that handles vaidation of many US formats like state abbreviations, zip codes, SSNs, etc. 274

At the time of writing, this was a beta release so you needed to pass the -f parameter to the PEAR install command. 275

<?php

include "Validate/US.php";

$validate = new Validate\_US();

echo $validate->phoneNumber("614-999-9999");

?>

# 10: Working with the File and Operating System

PHP is particularly adept at working with the underlying file system, so much so that it is gaining popularity as a command-line interpretor (a capability introducted in 4.2.0). This is beyond the scope of this book.

278

The basename function returns the filename component of a path.

If the optional suffix parameter is included, the suffix will be excluded if it matches (i.e., ".txt").

The dirname function returns the directory portion of a path. 279

The pathinfo function returns an associative array consisting of the path, base name, and extension.

<?php

$pathinfo = pathinfo("/home/www/htdocs/book/chapter10/index.html");

echo "Dir name: $pathinfo[dirname]<br />\n";

echo "Base name: $pathinfo[basename] <br />\n";

echo "Extension: $pathinfo[extension] <br />\n";

?>

280

The realpath function returns the absolute path with all relative and symbolic references converted.

The filesize function returns the size of a file:

<?php

$file = "/www/htdocs/book/chapter1.pdf";

$bytes = filesize("$file"); // Returns 91815

echo "File ".basename($file)." is $bytes bytes, or

".round($bytes / 1024, 2)." kilobytes.";

?>

281

The disk\_free\_space function returns the amount of free disk space:

<?php

$drive = "/usr";

echo round((disk\_free\_space($drive) / 1048576), 2);

?>

I used "." as the drive to get the free space of the current drive.

The disk\_total\_space function returns the size of the disk. 282

<?php

$systempartitions = array("/", "/home","/usr", "/www");

foreach ($systempartitions as $partition) {

$totalSpace = disk\_total\_space($partition) / 1048576;

$usedSpace = $totalSpace - disk\_free\_space($partition) / 1048576;

echo "Partition: $partition (Allocated: $totalSpace MB.

Used: $usedSpace MB.)";

}

?>

The following recursive function was created to get the size of a directory:

<?php

function directory\_size($directory) {

$directorySize=0;

/\* Open the directory and read its contents. \*/

if ($dh = @opendir($directory)) {

/\* Iterate through each directory entry. \*/

while (($filename = readdir ($dh))) {

/\* Filter out some of the unwanted directory entries. \*/

if ($filename != "." && $filename != "..")

{

// File, so determine size and add to total.

if (is\_file($directory."/".$filename))

$directorySize += filesize($directory."/".$filename);

// New directory, so initiate recursion. \*/

if (is\_dir($directory."/".$filename))

$directorySize += directory\_size($directory."/".$filename);

}

} #endWHILE

} #endIF

@closedir($dh);

return $directorySize;

} #end directory\_size()

$directory = "/usr/local/apache2/htdocs/book/chapter10/";

$totalSize = round((directory\_size($directory) / 1024), 2);

echo "Directory $directory: ".$totalSize. "kb.";

?>

The fileatime function returns the last time a file was accessed in UNIX timestamp format. 284

The filectime function returns the file's last modification date. 285

<?php

$file = "/usr/local/apache2/htdocs/book/chapter10/stat.php";

echo "File inode last changed: ".date("m-d-y g:i:sa", fileatime($file));

?>

Last changed time differs from last modified time. The former includes changes to permissions, etc.. The latter only refers to changes in the file's content.

The filemtime function returns the last time a file was modified. 286

The term resource is commonly used to refer to any entity from which an input or output stream can be initiated. Most of the "file" handling functions in this section are actually resource handling function because they can work with network sockets as well as files.

The newline character is \n on Linux and \r\n on Windows. 287

The feof function determines if the end of a resource has been reached.

<?php

$fh = fopen("/home/www/data/users.txt", "rt");

while (!feof($fh)) echo fgets($fh);

fclose($fh);

?>

Typically you will create a handle when you open a resource.

The fopen function binds a file to a handle:

resource fopen(string resource, string mode [, int use\_include\_path [, resource zcontext]])

288

The following modes are available:

|  |  |
| --- | --- |
| Mode | Description |
| r | Read only. Pointer placed at beginning of file. |
| r+ | Read and write. File point placed at beginning of file. |
| w | Write only. Before writing, delete file and place pointer at beginning. Create the file if it doesn't exist. |
| w+ | Read and write. Delete file contents and put pointer at the beginning. Create the file if it doesn't exist. |
| a | Write only, opening the file for append. File pointer placed at end. Create file if it doesn't exist. |
| a+ | Read and write. File pointer is placed at the end of the file. |
| b | Open the file in binary mode |
| t | Open the file in text mode |

Table 1: fopen Modes

288

In my own example, I open a web site and return the HTML code line by line:

<?php

$fh=fopen("http://localhost:8080/", "r");

while (!feof($fh)){

$t=fgets($fh);

print nl2br(htmlspecialchars($t));

}

?>

It is good programming practice to use fclose to close any resource you have opened once you are done with it. 289

The file function will read each line of a file into an array using newline as the delimiter (the newline is retained in each array element):

<?php

$users = file("users.txt");

foreach ($users as $user) {

list($name, $email) = explode(" ", $user);

// Remove newline from $email

$email = trim($email);

echo "<a href=\"mailto:$email\">$name</a> <br />\n";

}

?>

290

Although simple in nature, the importance of this function can't be overstated.

Note that file returns an array rather than a resouce handle.

The file\_get\_contents function reads the contents of a file into a string. 291

The following code uses this function to achieve the same results as the code above:

<?php

$userfile= file\_get\_contents("users.txt");

// Place each line of $userfile into array

$users = explode("\n",$userfile);

foreach ($users as $user) {

list($name, $email) = explode(" ", $user);

echo "<a href=\"mailto:$email\">$name/a> <br />";

}

?>

292

The fgetcsv function parses each line of a CVS file to an array:

array fgetcsv(resource handle [,int length [, string delimiter [, string enclosure]]])

Following is an example:

<?php

$fh = fopen("/home/www/data/subscribers.csv", "r");

while (list($name, $email, $phone) = fgetcsv($fh, 1024, ",")) {

echo "<p>$name ($email) Tel. $phone</p>";

}

?>

293

The fgets function reads everything up to a newline, EOF, or the specified length into a string. 293

If length is not specified, 1,024 characters is assumed as the maximum length. 294

The fgetss function is similar except that it also strips out any HTML or PHP tags it encounters.

<?php

/\* Build list of acceptable tags \*/

$tags = "<h2><h3><p><b><a><img>";

/\* Open the article, and read its contents. \*/

$fh = fopen("article.html", "rt");

while (!feof($fh)) {

$article .= fgetss($fh, 1024, $tags);

}

fclose($fh);

/\* Open the file up in write mode

and write $article contents. \*/

$fh = fopen("article.html", "wt");

fwrite($fh, $article);

fclose($fh);

?>

The fgetc function reads one character at a time from the specified resource. 295

You can use the fread function to read in characters up to the specified length or EOF. Newlines are not considered. You can use this in conjunction with filesize to determine to the number of bytes to read.

The readfile function reads an entire file and outputs it to the output buffer. It returns the number of bytes read.

The fscanf function offers a means for parsing a resource according to a predefined format. 297

mixed fscanf(resource handle, string format [, string var1])

Here is an example:

123-45-6789

234-56-7890

345-67-8901

?php

$fh = fopen("socsecurity.txt", "r");

/\* Parse each SSN in accordance with

integer-integer-integer format. \*/

while ($user = fscanf($fh, "%d-%d-%d")) {

list ($part1,$part2,$part3) = $user;

...

}

fclose($fh);

?>

The fwrite function outputs the contents of a string to the specified resource.

The fseek function will move the file pointer to the desired position. 298

int fseek(resource handle, int offset [, int whence])

The arguments for whence are:

* SEEK\_CUR
* SEEK\_END
* SEEK\_SET

299

The ftell function retrieves the current file pointer's offset.

The rewind function puts the file pointer back to the beginning of the file.

The opendir function opens a directory stream specified by a path:

resource opendir(string path)

The closedir function will close the directory stream.

The readdir function reads each element in a directory:

string readdir(in directory\_handle)

300

<?php

$dh = opendir('/usr/local/apache2/htdocs/');

while ($file = readdir($dh))

echo "$file <br>";

closedir($dh);

?>

The scandir function, introduced in PHP 5, reads the directory contents into an array:

array scandir(string directory, [,int sorting\_order [,resource context]])

Example:

<?php

print\_r(scandir("/usr/local/apache2/htdocs"));

?>

The rmdir function will remove a directory (but only if it is empty). 301

The rename function will rename a file or directory. 302

The touch function will set the last-modified and last-accessed time of a file (you can specify a time or use current by default). If the file does not exist it will be created.

Be sure to sanitize user input, especially before passing it along to a system level function like system or exec.

303

The HTMLDOC program (<http://www.htmldoc.org/>) converts html documents into indexed html, postscript, or PDF files. This was used as an example of where you want to sanitize input before passing it along to a shelled to program.

The escapeshellarg and escapeshellcmd functions are used to sanitize user input before passing them along to shell programs. 304

The former makes all input appear as a single argument, the latter escapes shell matacharacters.

The exec function is best-suited for executing an operating system-level application indeded to continue in the server background:

string exec(string command [, array output [, int return\_var]])

305

If the optional output parameter is given, you will get an array of each line of output (the system itself returns the last output line). If return\_var is given, you will get the exit status of the program.

The system function is useful when you want to output the executed command's results.306

The passthru function is similar to exec except that it returns binary output to the caller:

void passthru(string command [, int return\_var])

307

You can use backticks to execute a shell command and have the output returned. It is operationally identical to shell\_exec.

# 11: PEAR

Good programmers write solid code, white great programmers reuse the code of good programmers. 309

PEAR (PHP Extension and Application Repository).

PEAR, started in 1999, has more than 400 packages under 37 different topics. Packages are thoroughly reviewed by the community for code quality and adherence to standards before they are accepted into the repository.

When you run the PEAR installer (even if it is already installed), you will be asked to install seven additional packages. The author suggests accepting this. 310

The Mail package is capable of sending email three ways – the sendmail program, SMTP, or PHP's mail() function. 311. This is one of the seven packages. 311

You can browse PEAR solutions at <http://pear.php.net/>. 312

You should regularly look to the repository before attempting to resolve any significant programming task.

The author uses the Numbers\_Roman package as an example (converting years into Roman numerals) of looking to PEAR before trying to create a solution yourself.

The PEAR package manager has been installed by default wince version 4.3.0. The chapter also explains how to take advantage of PEAR without the package manager for those who are on a hosting plan that doesn't offer it. 313

PEAR is not installed by default on Windows. You need to run the go-pear.bat file.314

You will probably want to append the path to PEAR to your system path. 315

At the end of installation, a PEAR\_ENV.reg file is created that will do this and add other environment variables that will prove convenient. The author recommends running this.

You will want to check for updates to the PEAR package manager often. Just re-run the installation script for this.316

Package manager syntax:

pear [options] command [command-options] <parameters>

pear help <command>

To view packages installed on the machine:

pear list

To learn more about a particular package:

pear info packageName

317

Installing a PEAR packages is surprisingly automated:

pear install [options] package

318

Later versions of PEAR will install dependencies by default. However, you can also use the -a or –alldeps options:

pear install -a Auth\_HTTP

same as

pear install –alldeps Auth\_HTTP

319

If you can't use the package manager or wish to install an older version of the package, you can download it from the repository and install it manually. They are stored in a tarred Gzip format.

You will need to unzip this into a folder within the include\_path directive.

You use a package via include or, preferably, require. You can tell where package is located through it's name: Numbers\_Roman would be located under Numbers\Roman in the tree hierarchy.

You should regularly check for upgrades:

pear upgrade [package name]

320

The following would upgrade pear itself:

pear upgrade pear

You can upgrade all packages with the following:

pear upgrade-all

321

You can uninstall a package like this:

pear uninstall [options] packageName

pear uninstall package1 package2 package3

322

# 12: Date and Time

PHP 5.1 introduced vastly improved date and time manipulation functions. 323

The UNIX world opted to base time on Coordinated Universal Time (UTC). It represents temporal values based on the number of seconds since January 1, 1970. 00:00:00 UTC January 1, 1970 is also known as Unix epoch. 324

On Windows, due to an integer overlfow issue, you will get an error if you try to work with a time stamp earlier than the Unix epock with the functions in this chapter. This is not a problem on UNIX.

The checkdate function checks if a date is valid:

Boolean checkdate(int month, int day, int year)

325

The date function returns a string representation of the current date:

string date(string format [, int timestamp])

326

echo "Today is " .date("F d, Y");

// Today is August 22, 2007

327

echo "Today is ".date("l");

// Today is Wed

328

<?php

$weekday = date("l");

$daynumber = date("dS");

$monthyear = date("F Y");

printf("Today is %s the %s day of %s", $weekday,$daynumber,$monthyear);

?>

produces:

Today is Thursday the 29th day of May 2008

The following will display the current time:

echo "the time is ".date("h:i:sa");

//the time is 07:11:26am

329

The gettimeofday function returns an associative array with the elements of the current time.

* dsttime
* minuteswest
* sec
* usec

The getdate function returns an associative array. You can pass in a UNIX timestamp to use other than the current time: 330

* hours
* mday
* minutes
* mondaymonth
* seconds
* wday
* weekday
* yday
* year
* 0

The Windows operating system doesn't support negative timestamp values, so the earliest date you can represent with getdate is January 1, 1970.

331

PHP offers two functions for working with timestamps – time and mktime.

The time function returns the current timestamp as an integer. 332

my code:

<?php

$array=getdate(1187897100);

foreach($array as $item=>$value){

print "$item => $value . \n<br />";

}

?>

seconds => 0 .

minutes => 25 .

hours => 9 .

mday => 23 .

wday => 4 .

mon => 8 .

year => 2007 .

yday => 234 .

weekday => Thursday .

month => August .

You can use the date function to make a timestamp readable:

echo date("F d, Y h:i:s", 1187897100);

August 23, 2007 09:25:00

The mktime function is useful for returning a timestamp based on components:

int mktime([int hour [, int minute [, int second [,int month [,int day [, int year [,int is\_dst]]]]]]])

This function is particularly useful for calculating the difference between two points in time:

<?php

$now = mktime();

$taxday = mktime(0,0,0,4,15,2006);

// Difference in seconds

$difference = $taxday - $now;

// Calculate total hours

$hours = round($difference / 60 / 60);

echo "Only $hours hours until tax day!";

?>

333

The setlocale function changes PHP's localization default. 334

Microsoft has devised its own set of language and country codes rather than following the standards. You can retrieve a list of them at [http://msdn.microsoft.com](http://msdn.microsoft.com/).

335

Americans represent large numbers in 1,400.00 format, Europeans in 1.400,00.

Europeans and much of the world represent dates in dd-mm-yyyy format.

The following code will format money according to the Italian format:

setlocale(LC\_MONETARY, "it\_IT");

echo money\_format("%i", 478.74);

EUR 478,54

336

(this did not work on my system. money\_format was not recognized as a function)

The getlastmod function returns the timestamp for the page's last modification date. 338

<?php

$lastmod = date("F d, Y h:i:sa", getlastmod());

echo "Page last modified on $lastmod";

?>

Page last modified on June 03, 2008 07:21:00am

339

Use the t parameter of date to determine the number of days in a month:

printf("There are %d days in %s.", date("t"), date("F"));

The following will determine the number of days in a desired month given the timestamp:

<?php

$lastday = mktime(0, 0, 0, 3, 0, 2006);

printf("There are %d days in February, 2006.", date("t",$lastday));

?>

You can use the strtotime function and GNU date syntax to determine a future date:

<?php

echo "45 days from now<br />";

$futuredate = strtotime("45 days");

echo date("F d, Y", $futuredate);

echo "<br />45 days ago<br />";

$pastdate = strtotime("-45 days");

echo date("F d, Y", $pastdate);

echo "<br />10 weeks and 2 days from now<br />";

$futuredate = strtotime("10 weeks 2 days");

echo date("F d, Y", $futuredate);

?>

340

The PEAR Calendar packages will:

* Render a calendar of any scope in a format of your choice (hourly, daily, weekly, monthly, yearly, etc.)
* Navigate calendars
* Validate any date (does April 1, 2019 fall on a Monday?)
* Extending Calendar's capabilities

341

To fully capitalize on Calendar, you also need to install Date:

pear install -a -f Date

Calendar was in beta at the time of writing, hence the -f flag.

The following will create a monthly grid calendar:

<?php

require\_once 'Calendar/Month/Weekdays.php';

$month = new Calendar\_Month\_Weekdays(2006, 4, 0);

$month->build();7

echo "<table cellspacing='5'>\n";

echo "<tr><td class='monthname' colspan='7'>April, 2006</td></tr>";

echo "<tr><td>Su</td><td>Mo</td><td>Tu</td><td>We</td>

<td>Th</td><td>Fr</td><td>Sa</td></tr>";

while ($day = $month->fetch()) {

if ($day->isFirst()) {

echo "<tr>";

}

if ($day->isEmpty()) {

echo "<td>&nbsp;</td>";

} else {

echo '<td>'.$day->thisDay()."</td>";

}

if ($day->isLast()) {

echo "</tr>";

}

}

echo "</table>";

?>

343

Date and time support was enhanced in PHP 5.1. An object-oriented interface was added and also the ability to manage dates and times in respect to various time zones. 345

You can use the DateTime class:

object DateTime([string $time [, DateTimeZone $timezone]])

You can set the time later and instantiate a plain DateTime object:

$date = new DateTime();

You can use any of the formats supported by strtotime:

$date = new DateTime("27 February 2007 12:55);

$date = new DateTime("21:55");

346

You can use the format method to retrieve a formatted portion of the date. It accepts the same format as date:

<?php

$date=new DateTime();

echo $date->format("Y-m-d h:i:sa");

?>

You can use the setDate method to set the date after instantiation:

$date->setDate(2007,2,27);

You can use the setTime method to set the time:

Boolean setTime(integer hour, integer minute [, integer second])

347

You can use the modify method to modify the the DateTime object:

$date->modify("+7 hours");

# 13: Forms

Two useful sites that offer HTML forms tutorials are:

<http://www.w3schools.com/>

<http://www.topxml.com/>

350

You should following the cardinal rule to never trust user input and filter it first through a function like htmlentities.

It is a quite common practice to have the form submit to itself and conditional logic then determines the action or what is displayed. 352

If you are submitting the form to the originating document, you can use the$\_SERVER['PHP\_SELF'] superglobal to avoid having to change the code if you change the page name:  
  
<form action="<?php echo $\_SERVER['PHP\_SELF']; ?>" method="post">

The following submits data to itself and validates an email address:

<?php

// Function used to check email syntax

function validate\_email($email)

{

// Create the syntactical validation regular expression

$regexp = "^([\_a-z0-9-]+)(\.[\_a-z0-9-]+)\*@([a-z0-9-]+)

(\.[a-z0-9-]+)\*(\.[a-z]{2,6})$";

// Validate the syntax

if (eregi($regexp, $email)) return 1;

else return 0;

}

// Has the form been submitted?

if (isset($\_POST['submit']))

{

echo "Hi ".$\_POST['name']."!<br />";

if (validate\_email($\_POST['email']))

echo "The address ".$\_POST['email']." is valid!";

else

echo "The address <strong>".$\_POST['email']."</strong> is invalid!";

}

?>

<form action="subscribe.php" method="post">

<p>

Name:<br />

<input type="text" name="name" size="20" maxlength="40" value="" />

</p>

<p>

Email Address:<br />

<input type="text" name="email" size="20" maxlength="40" value="" />

</p>

<input type="submit" name = "submit" value="Go!" />

</form>

In order to use multivalued form components, you need to append [] to the name so that PHP will create an array.:

<?php

if (isset($\_POST['submit']))

{

echo "You like the following languages:<br />";

foreach($\_POST['languages'] AS $language) echo "$language<br />";

}

?>

<form action="multiplevalueexample.php" method="post">

What's your favorite programming language?<br /> (check all that apply):<br />

<input type="checkbox" name="languages[]" value="csharp" />C#<br />

<input type="checkbox" name="languages[]" value="jscript" />JavaScript<br />

<input type="checkbox" name="languages[]" value="perl" />Perl<br />

<input type="checkbox" name="languages[]" value="php" />PHP<br />

<input type="submit" name="submit" value="Go!" />

</form>

355

The PEAR repository offers the HTML\_QuickForm package to handle form generation, validation, and processing. 355

It offers more than 20 XHTML-compliant form elements, client and server side validation, the ability to integrate with template engines like Smarty, and the ability to create your own custom elements.

356

The HTML\_QuickForm package also requires HTML\_Common, so your install command would be like this:

pear install –onlyreqdeps HTML\_QuickForm

HTML\_QuickForm has been superseded by HTML\_QuickForm2

To use it, instantiate an object, call the addElement method for the elements you want to add, then call the display method.

HTML\_QuickForm lets you pass input through filters, which are just PHP functions (you can use built-in functions like strtoupper or create your own). If your filter is going to alter data the user relies on, you should notifiy him of changes. 358

Rules impose restrictions on what will be accepted rather than actually changing the entry. 359

The following predefined validation rules are available:

* alphanumeric
* callback
* compare
* email
* lettersonly
* maxlength
* minlength
* nopunctuation
* nonzero
* numeric
* rangelength
* regex
* required

$form->addRule('zipcode','Please enter a zipcode', 'required',null,'client);

360

Filters are executed automatically. Rules are not executed unless the validate method is called. 361

The following methods are available to process the form:

* getSubmitValues (load the form data into an array)
* process (pass them to a function)
* exportvalue (retrieve an item by name)

362

HTML\_QuickForm also has an auto-completion feature. 363

# 14: Authenticating Your Users

$\_SERVER['PHP\_AUTH\_USER'] and $\_SERVER['PHP\_AUTH\_PW']

365

The HTTP protocol offers a fairly effective means for user authentication:

* Client requests restricted resource
* Server responds with 401 (Unauthorized access)
* Browser recognizes 401 and pops up authentication prompt
* User credentials are sent to server. Access is either allowed or denied.
* If the user is validated, the browser stored the authentication information in its authentication cache until the cache is cleared or another 401 response is received.

366

Passing the credentials is not encrypted unless you use a secure communications channel, typically accomplished using Secure Sockets Layer (SSL). 367

PHP offers two predefined variables to authenticate a user:

* $\_SERVER['PHP\_AUTH\_USER']
* $\_SERVER['PHP\_AUTH\_PW']

There are two caveats:

* Both variables must be verified at the start of a restricted page (typically with the authentication code in an include page)
* These variables don't function properly with the CGI version of PHP or with Microsoft IIS.

In IIS 6 or earlier, you have to parse this information from $\_SERVER['HTTP\_AUTHORZATION']. In IIS 7 or newer, forms authentication is no longer restricted to ASP.NET pages and works with PHP.

368

The header and isset functions are commonly used with authentcation.

void header(string string [, boolean replace [, int http\_reponse\_code]])

Example:

header('WWW-Authenticate: Basic Realm="Book Projects"');

header("HTTP/1.1 401 Unauthorized");

369

The following example checks to see if the credentials are set and echoes them if they are:

<?php

if (isset($\_SERVER['PHP\_AUTH\_USER']) and isset($\_SERVER['PHP\_AUTH\_PW'])) {

// execute additional authentication tasks

} else {

echo "<p>Please enter both a username and a password!</p>";

}

?>

This chapter explores 5 authentication methodologies:

* Hard coded login pair
* File based authentication
* Database based authentication
* IP based authentication
* PEAR HTTP authentication

370

When using file based authentication, it is crucial that the file be stored outside the server document root. 371

Using file based authentication involves several functions

* Read the file into an array with file()
* Using explode() to split a string into substrings
* Using md5() to calculate an MD5 hash

The following code illustrates this:

authenticate.txt:

jason:60d99e58d66a5e0f4f89ec3ddd1d9a80

donald:d5fc4b0e45c8f9a333c0056492c191cf

mickey:bc180dbc583491c00f8a1cd134f7517b

<?php

// Preset authentication status to false.

$authorized = FALSE;

if (isset($\_SERVER['PHP\_AUTH\_USER']) && isset($\_SERVER['PHP\_AUTH\_PW'])) {

// Read the authentication file into an array

$authFile = file("/usr/local/lib/php/site/authenticate.txt");

// Cycle through each line in file, searching for authentication match.

foreach ($authFile as $login) {

list($username, $password) = explode(":", $login);

// Remove the newline from the password

$password = trim($password);

if (($username == $\_SERVER['PHP\_AUTH\_USER']) &&

($password == md5($\_SERVER['PHP\_AUTH\_PW']))) {

$authorized = TRUE;

break;

}

}

}

// If not authorized, display authentication prompt or 401 error

if (! $authorized) {

header('WWW-Authenticate: Basic Realm="Secret Stash"');

header('HTTP/1.0 401 Unauthorized');

print('You must provide the proper credentials!');

exit;

}

// restricted material goes here...

?>

372

Of all the authentication methodologies discussed here, database authentication is the most powerful. 373

The following example authenticates against a MySQL database:

<?php

/\* Because the authentication prompt needs to be invoked twice,

embed it within a function.

\*/

function authenticate\_user() {

header('WWW-Authenticate: Basic realm="Secret Stash"');

header("HTTP/1.0 401 Unauthorized");

exit;

}

/\* If $\_SERVER['PHP\_AUTH\_USER'] is blank, the user has not yet been prompted for

the authentication information.

\*/

if (! isset($\_SERVER['PHP\_AUTH\_USER'])) {

authenticate\_user();

} else {

// Connect to the MySQL database

$conn = mysql\_connect("localhost", "jason", "secret");

$db = mysql\_select\_db("corporate");

// Create and execute the selection query.

$query = "SELECT username, pswd FROM userauth

WHERE username='$\_SERVER[PHP\_AUTH\_USER]' AND

pswd=md5('$\_SERVER[PHP\_AUTH\_PW]')";

$result = mysql\_query($conn, $query);

// If nothing was found, reprompt the user for the login information.

if (mysql\_num\_rows($result) == 0) {

authenticate\_user();

}

else {

echo "Welcome to the secret archive!";

}

}

?>

374

The $\_SERVER['REMOTE\_ADDR'] variable returns the user's IP address.

The PEAR Auth\_HTTP package hides many of the authentication implementation details within a class. 377

Because it inherits from the Auth class (another package), it allows you to use DB, LDAP, POP3, IMAP, RADIUS, and SAMBA as well as flat file.

pear install -o auth\_http

The following authenticates against a MySQL database:

<?php

require\_once("Auth/HTTP.php");

// Designate authentication credentials, table name,

// username and password columns, password encryption type,

// and query parameters for retrieving other fields.

$dblogin = array (

'dsn' => "mysql://corpweb:secret@localhost/corporate",

'table' => "userauth",

'usernamecol' => "username",

'passwordcol' => "pswd",

'cryptType' => "md5"

'db\_fields' => "\*"

);

// Instantiate Auth\_HTTP

$auth = new Auth\_HTTP("DB", $dblogin) or die("blah");

// Begin the authentication process

$auth->start();

// Message to provide in case of authentication failure

$auth->setCancelText('Authentication credentials not accepted!');

// Check for credentials. If not available, prompt for them.

if($auth->getAuth())

{

echo "Welcome, $auth->commonname<br />";

}

?>

378

The DSN for the DB authentication must be in the following format:

datasource:username:password@hostname/database

mysql://corpweb:secret@localhost/corporate

379

For a complete list of datasource values, see <http://pear.php.net/package/DB>. 380

The CrackLib library (<http://www.crypticide.com/dropdafe/>) is used to test password strength. 381

The CrackLib library is located at <http://sourceforge.net/projects/cracklib>.

CrackLib is often pre-installed on UNIX/Linux systems.

You need to get the PHP extension for CrackLib from PECL (PHP Extension Community Library). 382

CrackLib comes with a dictionary but you can use various other dictionaries. 383

The University of Oxford has a lot of dictionaries, including specialty dictionaries. [ftp.ox.ac.uk](ftp://ftp.ox.ac.uk/).

You should have a simple, automated process for a user retrieving or resetting their password if forgotten.

You can use a one-time URL with an ?id=myOneTimeString. You can generate it with code like this:

$id=md5(uniqid(rand(),1));

384

One example is tagging a newsletter link with a such an id so you can determine who is following the link.

# 15: Handling File Uploads

In this chapter we look at:

* PHP's file-upload configuration directives.
* The $\_FILES superglobal array
* The built-in file-upload functions:
  + is\_uploaded\_file()
  + move\_uploaded\_file()
* Possible error messages
* Overview of HTTP\_Upload PEAR package

387

The multipartt/form-data Internet media type was designated for things like file uploads. 388

File upload code sample:

<form action="uploadmanager.php" enctype="multipart/form-data" method="post">

Last Name:<br /> <input type="text" name="name" value="" /><br />

Class Notes:<br /> <input type="file" name="classnotes" value="" /><br />

<p><input type="submit" name="submit" value="Submit Notes" /></p>

</form>

<?php

/\* Set a few constants \*/

define ("FILEREPOSITORY","/home/www/htdocs/class/classnotes/");

/\* Make sure that the file was POSTed. \*/

if (is\_uploaded\_file($\_FILES['classnotes']['tmp\_name'])) {

/\* Was the file a PDF? \*/

if ($\_FILES['classnotes']['type'] != "application/pdf") {

echo "<p>Class notes must be uploaded in PDF format.</p>";

} else {

/\* move uploaded file to final destination. \*/

$name = $\_POST['name'];

$result = move\_uploaded\_file($\_FILES['classnotes']['tmp\_name'],

FILEREPOSITORY."/$name.pdf");

if ($result == 1) echo "<p>File successfully uploaded.</p>";

else echo "<p>There was a problem uploading the file.</p>";

} #endIF

} #endIF

?>

Successful file uploads in PHP involves coordination between configuration directives, the $\_FILES array, and a properly coded Web form. 388

Directives:

* file\_uploads = On/Off (default=1)
* max\_execution\_time=integer (default 30)
* memory\_limit=integerM (default 8M)
* upload\_max\_filesize=integerM (default 2M) (should be smaller than pos\_max\_size)
* upload\_tmp\_dir=string (default NULL)
* post\_max\_size=integerM (default 8M)

389

The $\_FILES array is the only EGCPFS (environment, get, cookie, put, files,server) superglobal array that is two-dimensional. 390

* $\_FILES['userfile']['error']
* $\_FILES['userfile']['name']
* $\_FILES['userfile']['size']
* $\_FILES['userfile']['tmp\_name']
* $\_FILES['userfile']['type']

391

The is\_uploaded\_file function is intended to detect if a hacker is trying to send a server file (like /etc/passwd) to the temporary upload bin to have it copied to the publicly available area. This function determines if the file was indeed uploaded from the form. 392

The move\_uploaded\_file function works similar to copy except that it ensures that the file being moved was uploaded via HTTP POST. If you use this function, you can forgoe is\_uploaded\_file.

move\_uploaded\_file($\_FILES['classnotes']['tmp\_name'],

"/www/htdocs/classnotes/" .$\_FILES['classnotes']['name']);

393

The $\_FILES['userfile']['error'] code will inform you if an error has occurred in the upload process.

The following code upload class notes and renames them for the last name of the student uploading them. It only allows PDF files:

<form action="uploadmanager.php" enctype="multipart/form-data" method="post">

Last Name:<br /> <input type="text" name="name" value="" /><br />

Class Notes:<br /> <input type="file" name="classnotes" value="" /><br />

<p><input type="submit" name="submit" value="Submit Notes" /></p>

</form>

<?php

/\* Set a few constants \*/

define ("FILEREPOSITORY","/home/www/htdocs/class/classnotes/");

/\* Make sure that the file was POSTed. \*/

if (is\_uploaded\_file($\_FILES['classnotes']['tmp\_name'])) {

/\* Was the file a PDF? \*/

if ($\_FILES['classnotes']['type'] != "application/pdf") {

echo "<p>Class notes must be uploaded in PDF format.</p>";

} else {

/\* move uploaded file to final destination. \*/

$name = $\_POST['name'];

$result = move\_uploaded\_file($\_FILES['classnotes']['tmp\_name'],

FILEREPOSITORY."/$name.pdf");

if ($result == 1) echo "<p>File successfully uploaded.</p>";

else echo "<p>There was a problem uploading the file.</p>";

} #endIF

} #endIF

?>

Remember that files uploaded and moved are under the guise of the Web server daemon owner. This user must have sufficient privileges in both the temporary and destination directories.

395

While it's easy enough to write your own upload functions, the PEAR HTTP\_Upload package renders this a truly trivial task.

Pear install HTTP\_Upload

396

To use this class, you pass the name of the form field to the getFiles method. If it passes the isValid method call, you then pass it along to moveTo.

Transferring files via the Web eliminates a great many inconveniences otherwise posed by firewalls and FTP servers and clients.

399

# 16: Networking

With the introduction of the command-line interface (CLI) in PHP 4.2.0, PHP is nor increasingly used for system administration. 401

Several of the functions in this chapter don't work on Windows. Check out the PEAR package Net\_DNS to emulate their capabilities.

402

DNS (Domain Name System)

The checkdnsrr function checks for the existence of DNS records: (not implemented on Windows)

int checkdnsrr(string host [, string type])

Possible record types include the following:

* A (Ipv4 Address Record)
* AAAA (Ipv6 Address Record)
* A6
* ANY (look for any records)
* CNAME (Canonical Name Records. Maps an alias to the real domain name)
* MX (Mail Exchange Record)
* NAPTR (naming Authority Pointer. Allows for non-DNS-compliant names
* NS (name Server Record. Determines the name of the server for the host)
* PTR (Pointer Record. Maps an IP address to a host)
* SOA (Start of Authority Record. Sets global parameters for the host)
* SRV (Services Record. Denotes the location of various services for the supplied domain)

403

<?php

$recordexists = checkdnsrr("example.com", "ANY");

if ($recordexists) echo "The domain name has been taken. Sorry!";

else echo "The domain name is available!";

?>

The dns\_get\_record function returns an array of DNS resource records. 404

HTTP's default port is 80 and SSH's is 22. 407

the getservbyname and getservbyport functions are available for learning about the services and their corresponding ports.

An example of getservbyname from the PHP help file:

<?php

$services = array('http', 'ftp', 'ssh', 'telnet', 'imap',

'smtp', 'nicname', 'gopher', 'finger', 'pop3', 'www');

foreach ($services as $service) {

$port = getservbyname($service, 'tcp');

echo $service . ": " . $port . "<br />\n";

}

?>

http: 80

ftp: 21

ssh:

telnet: 23

imap: 143

smtp: 25

nicname: 43

gopher: 70

finger: 79

pop3: 110

www: 80

On Linux the service names can be found in the /etc/services file. 408

The getservbyport function will return the default service for the port and protocol:

echo getservbyport(80,'tcp');

or

echo getservbyport(80,'');

408

The fsockopen function allows you to query both local and remote services:

resource fsockopen(string target, int port [, int errno [,string errstring [float timeout]]])

An example:

<?php

// Establish a port 80 connection with www.example.com

$http = fsockopen("www.example.com",80);

// Send a request to the server

$req = "GET / HTTP/1.1\r\n";

$req .= "Host: www.example.com\r\n";

$req .= "Connection: Close\r\n\r\n";

fputs($http, $req);

// Output the request results

while(!feof($http))

{

echo fgets($http, 1024);

}

// Close the connection

fclose($http);

?>

The following is a primitive port scanner (when I tried it, it quit when it encountered the first failure):

<?php

// Give the script enough time to complete the task

ini\_set("max\_execution\_time", 120);

// Define scan range

$rangeStart = 0;

$rangeStop = 1024;

// Which server to scan?

$target = "www.example.com";

// Build an array of port values

$range =range($rangeStart, $rangeStop);

echo "<p>Scan results for $target</p>";

// Execute the scan

foreach ($range as $port) {

$result = @fsockopen($target, $port,$errno,$errstr,1);

if ($result) echo "<p>Socket open at port $port</p>";

}

?>

410

The mail function of PHP is so useful and needed in so many Web applications that it might prove to be the most popular section in this book. 411

The following PHP configuration directives apply. Pay attention because they are platform specific:

* SMTP=string. Default:localhost. This is for the Windows platform. The MTA (Mail Transfer Agent) is set here. This can be either a local or remote source.
* sendmail\_from=string. Only useful on Winodws. On Unix, you must set this within the addl\_headers parameter of the mail function
* sendmail\_path=string. Default value:sendmail path. (sendmail -t -i). Applies only to Unix.
* smtp\_port=integer. Default: 25
* mail.force\_extra\_parameters=string. Default NULL. Applies only to Unix

411

Email is sent via the mail function:

boolean mail(string to, string subject,string message

[, string addl\_headers [, string addl\_params]])

412

On Unix, the mail function is dependent on the sendmail MTA. If you use an alternative MTA (e.g., qmail) you need to use that MTA's sendmail wrappers. 413

The simplest text email can be sent as follows:

<?php

mail("test@example.com", "This is a subject", "This is the mail body");

?>

The Mail and Mail\_Mime PEAR packages make sending HTML mail simple. The former sends the mail while the latter composes it.

Pear install Mail Mail\_Mime

The following example allows you to ping a server from PHP:

<?php

// Which server to ping?

$server = "www.example.com";

// Ping the server how many times?

$count = 3;

// Perform the task

echo "<pre>";

system("/bin/ping -c $count $server");

echo "</pre>";

// Kill the task

system("killall -q ping");

?>

418

# 17: PHP and LDAP

There was a great deal of home-made administrative solutions and multiple non-integrated systems until directory services came along. 425

Some of the more popular directory service products:

* Fedora Directory Server: [http://directory.fedoraproject.org](http://directory.fedoraproject.org/)
* Microsoft Active Directory: http:/www.microsoft.com/activedirectory/
* Novell eDirectory: <http://www.movell.com/products/edirectory/>
* Oracle Collaboration Suite: <http://www.oracle.com/collabsuite/>

All of these directory services rely on an open specification known as LDAP – Lightweight Directory Access Protocol. 426

Here are some resources for learning LDAP:

* LDAP v3 specification: <http://www.ietf.org/rfc/rfc3377.txt>. This is the official LDAP specification.
* The Official OpenLDAP Web site: <http://www.openldap.org/>
* IBM LDAP Redbooks: <http://www.redbooks.ibm.com/>. IBM's free 700+ page introduction to LDAP

The examples in this chapter use OpenLDAP's free read only online server. For Windows OpenLDAP binaries, the Lucas Bergman binaries (<http://www.bergmans.us/>) seem to be popular. 427

The ldap\_connect function connects to an LDAP server:

resource ldap\_connect([string hostname [, int port]])

427

My note: According to the PHP manual, you need to have libeay32.dll and ssleay32.dll somewhare in the system path before the LDAP functions will work on Win32. On Linux, support must be compiled in.

Skipping this chapter for now. Will return to it when I have LDAP support

# 18: Session Handlers

Session handling has been available since PHP 4.0 and is one of its coolest and most discussed features. 445

HTTP is a stateless protocol, meaning it is handled without knowledge of any past or future requests.

Session handling assigns each visitor a unique ID known as the Session ID (SID). 446

The SID is kept track of in one of two ways:

* Cookies
* URL rewriting

The cookie based method is recommended for security reasons. 447

You can use $\_GET['sid'] or $\_COOKIE['sid'] to retrieve the session id you have set. The parameter corresponds to the session name. You could also use $\_REQUEST['sid']. 448

There are nearly 30 configuration directives associated with session handling.

The session.save\_handler directive determines how session date will be saved:

session.save\_handler = files|mm|sqlite|user

The user defined function option can be used to store session date in a MySQL database. 449

The session.save\_path directive determines where the session files will be saved.

This is set to /tmp by default. It should not be set to a directory within the server document root.

Setting session.auto\_start to 1 will automatically have sessions started by default. However, using this method disables the ability to store objects within sessions. If you don't set this directive, you need to use session\_start to start a session. 450

PHPSESSID is the default session name. You can change this with the session.name directive.

The session.use\_cookies directive determines if cookies are used. If disabled, you can set session.use\_trans\_sid to 1 to have the session id automatically added to URLs. 451

The session.cookie\_lifetime directive determines in seconds how long a session id should last. The default is 0, which means it is lost when the user closes the browser.

You can set the session.cookie\_path variable to determine the valid path for a session cookie. If set to / it is valid for the whole site. If set to /books it is valid only for folders under the books directory.

The session.cookie\_domain directive is set to the valid domain and prevents other domains from reading your cookies.

The session\_start function both starts and resumes a session.453

The session\_unset function resets all session variables that have been set. The session\_destroy function ends the session. You can use these functions to allow a user to manually log out. 444

The session\_id function can retrieve or manually set the session id.

The $\_SESSION superglobal is used to set or retrieve session variables. 455

You can use the unset function to reset a session variable.

The session\_register and session\_unregister functions used to be the preferred method, but this is no longer the case because they rely on the register\_globals directive.

456

PHP handles all session data as a single string in the following format:

name/s:length:"value";

If you want to store all session data in a database for later retrieval, you can use the session\_encode and session\_decode functions to do this.

Using custom session handlers is a more efficient methods of storing session data in a database.

Normally, the session.cookie\_lifetime variable is set to 0 so that sessions only last until the browser is closed. If you want to make it so users can be automatically authenticated, you need to change this value.

459

The following code uses auto-login. The download code is in error, having include a POSTGRE SQL script instead of MySQL, but the logic seems to be the same as in the book:

<?php

session\_start();

// Has a session been initiated previously?

if (! isset($\_SESSION['name'])) {

// If no previous session, has the user submitted the form?

if (isset($\_POST['username']))

{

$username = $\_POST['username'];

$pswd = $\_POST['pswd'];

// Connect to the PGSQL database

$conn=pg\_connect("host=localhost dbname=corporate

user=website password=secret")

or die(pg\_last\_error($conn));

// Look for the user in the users table.

$query = "SELECT name FROM users

WHERE username='$username' AND pswd='$pswd'";

$result = pg\_query($conn, $query);

// If the user was found, assign some session variables.

if (pg\_num\_rows($result) == 1)

{

$\_SESSION['name'] = pg\_fetch\_result($result,0,'name');

$\_SESSION['username'] = pg\_fetch\_result($result,0,'username');

echo "You're logged in. Feel free to return at a later time.";

}

// If the user has not previously logged in, show the login form

} else {

include "login.html";

}

// The user has returned. Offer a welcoming note.

} else {

$name = $\_SESSION['name'];

echo "Welcome back, $name!";

}

?>

460

Of the four session storage method, user-defined custom handlers offer the greatest degree of flexibility. 462

You use the session\_set\_save\_handler function to register your custom handler. Your session handler needs to tailer six tasks with parameters:

* session\_open($session\_save\_path,$session\_name)
* session\_close()
* session\_read($sessionID)
* session\_write($sessionID,$value)
* session\_destroy($sessionID)
* session\_garbage\_collect($lifetime)

463

You would register your session handlers as follows:

session\_set\_save\_handler("session\_open","session\_close","session\_read",

"session\_write", "session\_destroy","session\_garbage\_collect");

464

We created the following table to store session information:

CREATE TABLE sessioninfo(

SID CHAR(32) NOT NULL,

expiration INT NOT NULL,

value TEXT NOT NULL,

PRIMARY KEY(SID)

);

The following code then implements our session handlers. Note that we need to define the requisite parameters whether or not we actually use them in out code:

<?php

/\*

\* mysql\_session\_open()

\* Opens a persistent server connection and selects the database.

\*/

function mysql\_session\_open($session\_path, $session\_name) {

$conn = mysql\_connect("localhost", "website", "secret");

$db = mysql\_select\_db("corporate");

} // end mysql\_session\_open()

/\*

\* mysql\_session\_close()

\* Doesn't actually do anything since the server connection is

\* persistent. Keep in mind that although this function

\* doesn't do anything in my particular implementation, it

\* must nonetheless be defined.

\*/

function mysql\_session\_close() {

return 1;

} // end mysql\_session\_close()

/\*

\* mysql\_session\_select()

\* Reads the session data from the database

\*/

function mysql\_session\_select($SID) {

$query = "SELECT value FROM sessioninfo

WHERE SID = '$SID' AND

expiration > ". time();

$result = mysql\_query($query);

if (mysql\_num\_rows($result)) {

$row = mysql\_fetch\_assoc($result);

$value = $row['value'];

return $value;

} else {

return "";

}

} // end mysql\_session\_select()

/\*

\* mysql\_session\_write()

\* This function writes the session data to the database.

\* If that SID already exists, then the existing data will be updated.

\*/

function mysql\_session\_write($SID, $value) {

$lifetime = get\_cfg\_var("session.gc\_maxlifetime");

$expiration = time() + $lifetime;

$query = "INSERT INTO sessioninfo

VALUES('$SID', '$expiration', '$value')";

$result = mysql\_query($query);

if (! $result) {

$query = "UPDATE sessioninfo SET

expiration = '$expiration',

value = '$value' WHERE

SID = '$SID' AND expiration >". time();

$result = mysql\_query($query);

}

} // end mysql\_session\_write()

/\*

\* mysql\_session\_destroy()

\* Deletes all session information having input SID (only one row)

\*/

function mysql\_session\_destroy($SID) {

$query = "DELETE FROM sessioninfo

WHERE SID = '$SID'";

$result = mysql\_query($conn, $query);

} // end mysql\_session\_destroy()

/\*

\* mysql\_session\_garbage\_collect()

\* Deletes all sessions that have expired.

\*/

function mysql\_session\_garbage\_collect($lifetime) {

$query = "DELETE FROM sessioninfo

WHERE sess\_expiration < ". time() - $lifetime;

$result = mysql\_query($query);

return mysql\_affected\_rows($result);

} // end mysql\_session\_garbage\_collect()

?>

465

# 19: Templating with Smarty

The separation of presentation from logic is a basic tenet of programming. 471

Solutions that separate web design from logic are called templating engines.

All the popular templating engines follow an approach where they have a presentational language that allow you to placeholders that use delimiters. 472

An example Smarty template and code that uses it:

.tpl file:

<html>

<head>

<title>{$pagetitle}</title>

</head>

<body>

{if $name eq "Kirk"}

<p>Welcome back Captain!</p>

{else}

<p>Swab the decks, mate!</p>

{/if}

</body>

</html>

.php file:

<?php

// Reference the Smarty class library.

require("smarty/Smarty.class.php");

// Create a new instance of the Smarty class.

$smarty = new Smarty;

// Assign a few page variables.

$smarty->assign("pagetitle","Welcome to the Starship.");

$smarty->assign("name","Kirk");

// Render and display the template.

$smarty->display("index.tpl");

?>

Smarty ([http://smarty.php.net](http://smarty.php.net/))is PHP's "unofficial official" templating engine. 476

Smarty compiles its templates in comparable PHP code so that subsequent calls are faster. It is smart enough to recompile when the template changes.

Smarty also offers a caching feature.

To install Smarty:

* Untar and unarchive Smarty to a location outside your Web document root (c:\php\includes\smarty).
* Add the location where the Smarty.class.php class file is located to your include\_path configuration. This will be the libs directory where you installed Smarty (i.e., .;php/includes/smarty/libs). As an alternative, you can specify the full path to the class or use a predefined SMARTY\_DIR constant.
* Create four directories where Smarty's template and configuration files will be stored:
  + templates
  + configs
  + templates\_c (for compiled templates)
  + cache

475

Smarty assumes these will be in the same folder as the script calling the class, but it is recommended that you place them outside the Web root folder by modifying the following class members:

* $template\_dir
* $compile\_dir
* $config\_dir
* $cache\_dir

476

<?php

// Reference the Smarty class library.

require("Smarty.class.php");

// Create a new instance of the Smarty class.

$smarty = new Smarty;

$smarty->template\_dir="/usr/local/lib/php5/smarty/template\_dir/";

$smarty->compile\_dir="/usr/local/lib/php5/smarty/compile\_dir/";

$smarty->config\_dir="/usr/local/lib/php5/smarty/config\_dir/";

$smarty->cache\_dir="/usr/local/lib/php5/smarty/cache\_dir/";

?>

477

Curly brackets are Smarty's default delimiters.

Here is an example Smarty template (index.tpl) and the code that would use it:

<html>

<head>

<title>{$title}</title>

</head>

<body bgcolor="#ffffff" text="#000000" link="#0000ff"

vlink="#800080" alink="#ff0000">

<p>

Hi, {$name}. Welcome to the wonderful world of Smarty.

</p>

</body>

</html>

<?php

require("Smarty.class.php");

$smarty = new Smarty;

// Assign two Smarty variables

$smarty->assign("name", "Jason Gilmore");

$smarty->assign("title", "Smarty Rocks!");

// Retrieve and output the template

$smarty->display("index.tpl");

?>

478

The prototype for the display method:

void display(string template [,string cache\_id [, string compile\_id]])

479

Smarty comments are enclosed within the delimiters and asterisks. They can be single or multiple lines:

{\* This is just a comment \*}

480

Smary allows modifiers (such as capitalization) to be applied to variables and uses an unusual syntax – the vertical bar:

{$var|modifier}

The capitalize function capitalizes the first letter of all words found in a variable.

{$title|capicalize}

481

The count\_words modifier counts the number of words in a variable.

The date\_format function is a wrapper to PHP's strftime function:

Submitted on: {$filed|date\_format:"%B %e, %Y"}

$smarty->assign("filed","1172345525");

482

The default function allows you to assign a default value if the application layer doesn't return one:

Author: {$author|default:"Anonymous"}

The strip\_tags function removes any markup tags from a variable:

<strong>{$title|strip\_tags}</strong>

483

The truncate function will truncate a string. You can specify the character limit (default is 80), a string to add at the end (such as "..."), or whether truncation should be immediate (TRUE) or at a word boundary.

{foreach from=$summaries item=summary}

{$summary|truncate:35:"..."}<br />

{/foreach}

Smarty offers several control structures for conditionally and iteratively evaluating passed in data:

* if
* foreach
* foreachelse
* section
* sectionelse

An example of if:

{if $month < 4}

Summer is coming!

{elseif $month ge 4 && $month <=9}

It's hot out today!

{else}

Brr... It's cold!

{/if}

485

The foreach function is similar in function to PHP's but the syntax is quite different. There are four parameters (two of which are required):

* from
* item
* key
* name

The following example displays the days of the week:

{foreach from=$daysofweek item=day}

{$day}<br />

{/foreach}

<?php

require\_once("Smarty.class.php");

$smarty = new Smarty;

$daysofweek=array("Mon.","Tues.","Weds.","Thurs.","Fri.","Sat.","Sun");

$smarty->assign("daysofweek",$daysofweek);

$smarty->display("daysofweek.tpl");

?>

You can use the key parameter for associative arrays.

The section structure acts as an enhanced foreach structure. It is much more powerful. It has several parameters, the first two of which are required:

* name:Name of the section. This is arbitrary.
* loop:Sets the number of times the loop iterates. Should be same name as array variable
* start:Index position at which iteration begins.
* step:Stepping value to traverse the array
* max:Maximum number of times the loop iteration will occur.
* show:Determines if the section will actually display

487

The following example uses section with an associative array:

{section name=book loop=$titles}

<p>

Title: {$titles[book].title}<br />

Author: {$titles[book].author}<br />

Published: {$titles[book].published}

</p>

{/section}

<?php

require\_once("Smarty.class.php");

$smarty=new Smarty;

$titles[]=array(

"title" => "Pro PHP",

"author" => "Kevin McArthur",

"published" => "2008"

);

$titles[]=array(

"title" => "Beginning Python",

"author" => "Lie Hetland",

"published" => "2005"

);

$smarty->assign("titles",$titles);

$smarty->display("titles.tpl");

?>

The include statement allows you to include other templates:

{include file="/user/local/lib/test/header.tpl"}

{\* some other stuff here \*}

{include file="test.tpl" myAttrib="misc attribute"}

{include file="/user/local/lib/test/footer.tpl" assign="footerVar"}

490

The fetch statement can accomplish the same thing as the include statement except that it allows you to use HTTP and FTP protocols. Unlike include, it can't assign attributes at retrieval time.

491

The insert statement is similar to include except that it is meant for content that is never cached. It accepts several parameters, the first of which is required:

* name: Determines the name of the insert function. The function insert\_name() will be called.
* assign:Optional to assign output to a variable rather than display.
* script: Points to a script that will be executed before the insert is made.
* var: Used to pass in other variables.

As an example, the following code would invoke a function called insert\_banner and pass it height and width parameters:

<img src="ads/images/{insert name="banner" height=468 width=60}.gif" />

492

The literal statement is used to mark off a section that Smarty should not try to intepret (mostly used to include things like JavaScript or CSS that would conflict with Smarty's {} delimiters).

{literal}

stuff not to interpret like {..}

{/literal}

The php statement is used to embed code that will be handled by the PHP engine:

{php}echo date("F j, Y"){/php}

You can also use the include\_php statement to include an external PHP script, which probably does a cleaner job of separating presentation from logic.

You can use configuration files with Smarty. 493

#Global variables – define before any sections

appName = "Example Application"

copyright = "Copyright 2008 by Howard Morrow"

[Aggregation]

title="Recent News"

warning="""Copyright protected...

Note the use of triple quotes for multiline stuff. """

[Detail]

title="A closer look..."

Items surrounded by brackets are called sections. Any thing outside of a section is global. Globals should be defined before any sections.

You would then use the config\_load function to load the configuration file. Configuration files are stored within the configs diretory.

{config\_load file="app.config"}

{config\_load file="app.config" section="Aggregation"}

494

There are several ways to access variables loaded from the configuration file:

hash mark:

{#myConfigVar}

{$smarty.config.myConfigVar}

get\_config\_vars("myConfigVar")

495

There are three alternatives for using CSS within Smarty (because of the {} conflict):

* Use a link tag in the head section(<link rel="stylesheet" type="text/css" href="default.css" />)
* Use a Smarty literal tag
* Redefine the Smarty delimiters

496

The first method is the preferred method.

While compilation is enabled by default, caching must be specifically enabled by the developer. 497

Cached pages remain valid for the $cache\_lifetime attribute. The default is 3600 seconds (1 hour). 498

By setting caching to 2 rather than 1, a template can override the globally set caching value.

# 20: Web Services

SimpleXML is new to PHP 5 and is a highly practical method for parsing XML. The SOAP extension was also made available in version 5. 504

Web Services technology is today's most promising solution to the interoperability problem. It's definition can be found at <http://w3.org/TR/ws-arch/>

Web Services offer platform-agnostic interfaces to expose application methods. 505

Because port 80 and 443 (HTTPS) are normally allowed, Web Services offer a convenient method for penetrating the corporate firewall.

Amazon, Google, and Microsoft have all offered Web Services APIs to some of their data:

* <http://www.amazon.com/webservices/>
* <http://code.google.com/more/>
* <http://msdn.microsoft.com/mappoint/>

506

RSS (Really Simple Syndication)

With RSS you can:

* Read feeds with a standalone RSS reader
* Use an online aggregator like <http://www.google.com/reader/>
* Retrieve and republish feeds as part of a third-party Web application or service.

507

Below is a sample RSS feed:

<?xml version="1.0" encoding="iso-8859-1"?>

<rss version="2.0">

<channel>

<title>Jason Gilmore</title>

<link>http://blogs.apress.com/</link>

<item>

<title>Ohio LinuxFest 2005</title>

<link>http://blogs.apress.com/archives/000639.html</link>

<description>The annual Ohio LinuxFest 2005 conference is rapidly

approaching, taking place at the Columbus Convention Center on October 1,

2005...</description>

</item>

<item>

<title>Retrieving Map Location Coordinates</title>

<link>http://blogs.apress.com/archives/000634.html</link>

<description>In the first installment of a three-part series for

Developer.com, you learned how to take advantage of Google&apos;s amazing

mapping API...</description>

</item>

<item>

<title>Pro PHP Security Published</title>

<link>http://blogs.apress.com/archives/000626.html</link>

<description>The Web&apos;s greatest advantage, accessibility, has also

proved to be its greatest detriment in terms of security...</description>

</item>

</channel>

</rss>

(not the same as in the book)

509

This example doesn't contain all the possible RSS elements like update interval, language, creator, etc. 510

MagpieRSS (Magpie for short) is a poerfull RSS parser written in PHP. It is available for download at <http://magpierss.sourceforge.net/>

If offers an amazingly practical and easy means for retrieving and parsing RSS feeds. 510

You can just copy the unpacked magpie folder to somewhere available in your include path. 511

The following retrieves the blog.xml feed and outputs it with print\_r:

<?php

require("magpie/rss\_fetch.inc");

$url = "http://localhost/book/20/blog.xml";

$rss = fetch\_rss($url);

print\_r($rss);

?>

An object named Magpie\_Feed is returned. 515

The following renders the feed in a browser:

<?php

require("magpie/rss\_fetch.inc");

// RSS feed location?

$url = "http://localhost/book/20/blog.xml";

// Retrieve the feed

$rss = fetch\_rss($url);

// Format the feed for the browser

$feedTitle = $rss->channel['title'];

echo "Latest News from <strong>$feedTitle</strong>";

foreach ($rss->items as $item) {

$link = $item['link'];

$title = $item['title'];

// Not all items necessarily have a description, so test for one.

$description = isset($item['description']) ? $item['description'] : "";

echo "<p><a href=\"$link\">$title</a><br />$description</p>";

}

?>

skipped 515-519 retrieving and rendering RSS feed because my version of magpie different

While XML represents an enormous leap forward in data management capabilities, using DOM, SAX, and XLT presents a reasonably steep learning curve. 519

SimpleXML offers a practical and intuitive method for processing XML structures. It is enabled by default as of PHP 5.

The following XML is used as for the SimpleXML examples here:

<?xml version="1.0" standalone="yes"?>

<library>

<book>

<title>Pride and Prejudice</title>

<author gender="female">Jane Austen</author>

<description>Jane Austen's most popular work.</description>

</book>

<book>

<title>The Conformist</title>

<author gender="male">Alberto Moravia</author>

<description>Alberto Moravia's classic psychological novel.</description>

</book>

<book>

<title>The Sun Also Rises</title>

<author gender="male">Ernest Hemingway</author>

<description>The masterpiece that launched Hemingway's

career.</description>

</book>

</library>

520

To use SimpleXML in PHP versions before 6, you need to disable the PHP directive zend.ze1\_compatability\_mode.

The simplexml\_load\_file function loads an XML file into an object:

object simplexml\_load\_file(string filename [, string class\_name])

521

<?php

$xml=simplexml\_load\_file("books.xml");

var\_dump($xml);

?>

If the class\_name parameter is used, an object of that type will be returned (it should extend SimpleXMLElement). FALSE is returned if there is a problem.

If the XML document is stored in a variable, you can load it with simplexml\_load\_string. 522

The PHP DOM XML extension allows you to manage an XML document using the DOM standard. You can use simplexml\_import\_dom to convert a DOM node into a SimpleXML node. 522

Use the attributes method to retrieve attributes:

object simplexml\_element->attributes()

An example:

<?php

$xml = simplexml\_load\_file("books.xml");

foreach($xml->book as $book) {

echo $book->author." is ".$book->author->attributes().".<br />";

}

?>

523

The following would get you the attributes of the second book:

echo $xml->book[2]->author->attributes();

The following code would allow you to display multiple attributes:

foreach($xml->book[0]->author->attributes() AS $a => $b){

printf("%s = %s <br />", $a, $b);

}

gender = female

age = 20

524

The asXML method returns a well-formed XML 1.0 string based on the SimpleXML object:

string simplexml\_element->asXML()

The following example echos the XML document with newline charqacters removed and special characters converted to XML emtities (so that you will see the tags in the browser):

<?php

$xml = simplexml\_load\_file("books.xml");

foreach($xml->book as $book) {

echo $book->author." is ".$book->author->attributes().".<br />";

}

?>

The children method returns the children of a node.

<?php

$xml = simplexml\_load\_file("books.xml");

foreach($xml->book[2]->cast->children() AS $character){

echo "$character<br />";

}

?>

525

The xpath method allows you to identify nodes using XPath.

Array simplexml\_element->xpath(string path)

The following lets us use the /library/book/author xpath expression to list all the authors:

<?php

$xml = simplexml\_load\_file("books.xml");

$authors = $xml->xpath("/library/book/author");

foreach($authors AS $author){

echo "$author<br />";

}

?>

526

# 24 MVC and the Zend Framework

Although Zend Framework is the newest of the frameworks discussed here, it is rapidly becoming the most popular of the framework solutions. 601

The MVC approach makes development more efficient by breaking the application into three distinct components:

* Model
* View
* Controller

603

The model defines the rules for the world or the process an application is intended to represent. 604

The view is responsible for formatting the data returned by the model and presenting it to the user.

The controller is responsible for determining how the application should respond based on the events occurring inside the application space.

A special controller called a front controller is responsible for routing all requests to the appropriate controller and returning the response.

The CakePHP framework ([http://cakeforge.org](http://cakeforge.org/)) is the framework that most closely resembles Ruby on Rails. 605

CakePHP was developed in 2005. It has lead to a the Cake Software Foundation ([http://www.cakefoundation.org](http://www.cakefoundation.org/)) and community repository ([http://cakeforge.org](http://cakeforge.org/)).

Unlike the other frameworks discussed here, CakePHP can run on both PHP 4 and 5

606

Below is a partial listing the some of the feature-specific components:

##### Table Partial Listing of Zend's Feature-Specific Components

| Component | Purpose |
| --- | --- |
| Zend\_Amazon | Facilitates interaction with Amazon E-Commerce Service. |
| Zend\_Cache | Caches data into speedy backend adapters such as RAM, SQLite, and APC (Alternative PHP Cache). |
| Zend\_Config | Facilitates the management of application configuration parameters. |
| Zend\_Controller | Manages the framework's controller component. |
| Zend\_Db | Drives the framework's PDO-based database API abstraction layer. |
| Zend\_Feed | Consumes RSS and Atom feeds. |
| Zend\_Filter | Facilitates the filtering and validation of data, including the ability to validate proper syntax for commonplace values such as e-mail addresses, credit card numbers, dates (according to ISO 8601 format), and phone numbers. |
| Zend\_Filter\_Input | Relies upon the methods provided by Zend\_Filter to filter input. |
| Zend\_Gdata | Provides an interface to several of Google's services, including, among others, Google Blogger, Google Calendar, and Google Notebook. |
| Zend\_HTTP\_Client | Performs HTTP requests. Presently capable of executing GET, POST, PUT, and DELETE requests. |
| Zend\_Json | Facilitates interaction between JavaScript and PHP by serializing PHP data to JSON (JavaScript Object Notation) and vice versa. See <http://www.json.org/> for more information about JSON. |
| Zend\_Log | Facilitates application logging. |
| Zend\_Mail | Sends text and MIME-compliant e-mail. |
| Zend\_Mime | Parses MIME messages. |
| Zend\_Pdf | Creates PDF documents. |
| Zend\_Search\_Lucene | Facilitates search engine development using the Lucene library. |
| Zend\_Service\_Amazon | Facilitates interaction with the Amazon Web Services API. |
| Zend\_Service\_Flickr | Facilitates interaction with the Flickr Web Services API. |
| Zend\_Service\_Yahoo | Facilitates interaction with the Yahoo! Web Services API. |
| Zend\_View | Manages the framework's view component. |
| Zend\_XmlRpc | Provides support for consuming and serving XML-RPC implementations. |

Table 2: Partial listing of feature specific components

The Zend Framework is fostered by Zend Technologies (<http://www.zend.com/>). 607

Zend\_Db drives the frameworks PDO-based database API abstraction layer. 608

Zend\_Filter facilitates the filtering and validation of data (such as email addresses, credit card numbers, etc.).

You can download the framework from <http://framework.zend.com/download>.

When you unpack the archive, only the /library directory is relevant. You might want to change the name to something easy to reference like zfw. 609

We will need to use mod\_rewire, modifying .htaccess to something like this:

RewriteEngine on

RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php

On Windows, mod\_rewire is disabled by default. You need to open httpd.conf and uncomment LoadModule require\_module modules/mod\_require.so. You also need to update the appropriate AllowOverride directive to allow .htaccess files to be used, then restart Apache.

610

You may also want to add the /library directiory to the include\_path variable in php.ini.

By default, the Zend Framework relies on a highly organized application directory structure known as the conventional modular directory structure. Its most basic structure looks like this: 611

* web server document root
  + index.php
  + application
    - modules
      * default
        + controllers
        + views

scripts

This structure opens the possibility to manage multiple hosted MVC applications within the same location.

Note in the following simple structure that there are three controllers, each matching up to a corresponding view directory:

* web server document root
  + index.php
  + application/
    - modules/
      * default/
        + controllers/

IndexController.php

BookController.php

AboutController.php

* + - * + views/

footer.phtml

header.phtml

scripts/

about/

contact.phtml

index.phtml

book/

index.phtml

toc.phtml

index/

index.phtml

Although not covered here, you can change the default directory structure if it doesn't fit your needs. 612

To avoid editing php.ini, I added the following code to the beginning of the index.php file:

ini\_set('include\_path',ini\_get('include\_path') . ':' . '/var/www/library/');

The following is the index.php file used in the book example:

<?php

// Load the Front Controller class

require\_once('Zend/Controller/Front.php');

// Instantiate an instance of the Front Controller Class

$frontController = Zend\_Controller\_Front::getInstance();

// Point to the module directory

$frontController->addModuleDirectory('./application/modules');

// Throw exceptions (useful during debugging)

$frontController->throwExceptions(TRUE );

// Start the Front Controller

$frontController->dispatch();

?>

613

It is assumed that the application will reside in the root directory of the site. If this isn't the case and you need to override the default behavior, see the setBaseUrl method.

The default controller class (IndexController.php) is as follows:

<?php

// Load the Zend\_Controller\_Action class

require\_once('Zend/Controller/Action.php');

class IndexController extends Zend\_Controller\_Action

{

// Accessed through http://www.example.com/

public function indexAction()

{

$this->view->title = "Welcome to Our Chess Club Web Site!";

}

}

?>

We then created AboutController.php:

<?php

// Load the Zend\_Controller\_Action class

require\_once('Zend/Controller/Action.php');

class AboutController extends Zend\_Controller\_Action

{

// Accessed through http://www.example.com/about/

public function indexAction()

{

$this->view->title = "About Our Chess Club";

}

// Accessed through http://www.example.com/about/you/

public function youAction()

{

// Page title

$this->view->title = "About You!";

// Retrieve the user's IP address

$this->view->ip = $\_SERVER['REMOTE\_ADDR'];

// Retrieve browser information

$this->view->browser = $\_SERVER['HTTP\_USER\_AGENT'];

}

}

?>

The following index.phtml view file was places in application/modules/default/views/scripts/index/:

<?php

echo $this->render('header.phtml');

?>

<div id="header">Next Chess Club Meeting: April 12</div>

<p>

Welcome to our Chess Club's Web site! We're a bunch of chess enthusiasts

who travel the globe in search of worthy oppon615ents. Join us at our next

meeting, held at the coffee shop on the corner of Third and Neil

each Tuesday at 6 p.m.

</p>

<?php

echo $this->render('footer.phtml');

?>

615

The about.phtml file was placed in the scripts/about folder (I had to name this index.phtml):

<?php

echo $this->render('header.phtml');

?>

<div id="header">About Our Chess Club</div>

<p>

Founded: 1997<br />

City: Columbus, Ohio<br />

Where we meet: Cup of Love, corner of Third and Neil<br />

When we meet: Each Tuesday at 6 p.m.<br />

Notes: Bring your board and pieces if you have them!

</p>

<?php

echo $this->render('footer.phtml');

?>

The you.phtml file was also placed in scripts/about/:

<?php

echo $this->render('header.phtml');

?>

<div id="header">About You!</div>

<p>

Your IP Address: <?php echo $this->escape($this->ip); ?><br />

Your Browser: <?php echo $this->escape($this->browser); ?><br />

</p>

<?php

echo $this->render('footer.phtml');

?>

616

You should pass all data originating in the controller through the escape() method, which will filter using the htmlspecialchars() function.

The Zend Framework has the ability to handle more sophisticated templating solutions than those used in this chapter, including Smarty.

The author chose Zend\_Service\_Yahoo as the special component to showcase in order to demonstrate using Web Services. 617

You need to register for a free Yahoo developer id to do this example ([http://developer.yahoo.com](http://developer.yahoo.com/)).

We place NewsController.php in the application/controllers directory:

<?php

// Load the Zend\_Controller\_Action class

require\_once('Zend/Controller/Action.php');

// Load the Yahoo! Service class

require\_once('Zend/Service/Yahoo.php');

class NewsController extends Zend\_Controller\_Action

{

public function indexAction()

{

// Invoke the Yahoo! Sservice

$yahoo = new Zend\_Service\_Yahoo("INSERT\_YAHOO\_ID");

// Execute the search

$results = $yahoo->newsSearch("chess");

// Send the search results to the view

$view->results = $results;

}

}

?>

We copy the following as application/modules/default/views/scripts/news/index.phtml:

<?php

echo $this->render('header.phtml');

?>

<h4>The Latest Chess News</h4>

<?php

foreach ($this->results as $result) {

printf("<p><a href='%s'>%s</a> | %s <br />",

$this->escape($result->NewsSourceUrl),

$this->escape($result->NewsSource),

date("F j, Y", $this->escape($result->PublishDate))

);

printf("%s </p>", $this->escape($result->Summary));

}

?>

<?php

echo $this->render('footer.phtml');

?>

619

NOTE: This example did not work with the version of Zend I was using (1.7.8)

## My Ubuntu 8.10 Apache Notes

On Ubuntu 8.10, mod\_rewrite was not on by default and neither was AllowOverrides for the root directory. To corrent this I copied:

/etc/apache2/mods-available/rewrite.load

to

/etc/apache2/mods-enabled/reqire.load

Note that I could have used ln to create a symbolic link instead.

I also changed the following:

/etc/apache2/sites-available/default:

DocumentRoot /var/www/

<Directory />

Options FollowSymLinks

AllowOverride None

</Directory>

<Directory /var/www/>

Options Indexes FollowSymLinks MultiViews

**# AllowOverride None**

**AllowOverride All**

Order allow,deny

allow from all

</Directory>