Advanced PHP Programming

International Standard Book Number: 0-672-32561-6

Warning and Disclaimer

When reviewing corrections, always check the print number of your book. Corrections are made to printed books with each subsequent printing. To determine the printing of your book, view the copyright page. The print number is right-most number on the line below the "First Printing" line. For example, the following indicates the 4th printing of a title.

First Printing: March 2004

06 05 04 03 10 9 8 7 6 5 4

Misprint	Correction
Page 14, next to last line of code before "Control Flow Constructs" WHERE **u**.dept id = d.dept id	WHERE **e**.dept id = d.dept id
Page 15, first three code blocks Change **echo** to **print**	
Page 16, third code block **Function** hello ($name)	**function** hello ($name)
Page 17, first full code block **If**(($number % 2) != **0**) }	**if**(($number % 2) != **0**) }
Page 17, second code block for($I=**0**; $i < $number; $i++){	For($I=**3**; $i < $number; $i++){
Page 23, first code block function clean cache($expiration time)	function clean cache($expiration time) **{**
Page 23, second code block, fifth line my function($parent[$parent+index][$child index]);	my function($parent[$parent index][$child index]);
Page 24, code block **echo** "Hello $name";	**print** "Hello $name";
Page 26, last line on page **Mysql** query($second query)	**mysql** query($second query)
Page 27, second head Avoiding Using Open Tags	Avoiding Using **Short** Open Tags

Page 27, second and sixth lines of code **echo** "Hello $username";	**print** "Hello $username";
Page 28, fourth line of code <tr><td><? echo $employee['name']	<tr><td><?**php** echo $employee['name']
Page 28, second line in third code block if(($year % 4 == **0**) &&	if(($year % 4 == **0**) &&
Page 30, second code block // Use the bitwise "AND" **operatorest** to see if the first bit in $i is set	// Use the bitwise "AND" **operator** to see if the first bit in $i is set
Page 32, ninth line in second code block * **echo** "$i is prime/n";	* **print** "$I is prime/n";
Page 38, first code block **echo** hello($name); **echo** "You are ".age($bday)." years old.\n"; **echo** goodbye ($name);	**print** hello($name); **print** "You are ".age($bday)." years old.\n"; **print** goodbye ($name);
Page 38, second paragraph in "Introduction to OO Programming" A class constructor in PHP5 should be named **contstructor()** so that the engine knows how to identify it.	A class constructor in PHP5 should be named **contstruct()** so that the engine knows how to identify it.
Page 39, end of first code block **echo** $user->hello(); **echo** "You are ".$user->age()." years old.\n"; **echo** $user->goodbye();	**print** $user->hello(); **print** "You are ".$user->age()." years old.\n"; **print** $user->goodbye();

Page 43, middle of page …one in the constructor, one during the assignment of the return value from the constructor to `$copy`, and one when you assign **`$copy`** to **`$obj`**.	…one in the constructor, one during the assignment of the return value from the constructor to `$copy`, and one when you assign **`$obj`** to **`$copy`**.
Page 43, last paragraph Inside the `clone()` method, you not only have `$this`, which represents the new object, but also `$that`, which is the object being cloned.	Inside the `clone()` method, you have `$this`, which is the new object with all the original object's properties already copied.
Page 44, code block remove `$this->other = $that->other;` line	
Page 53, last code block `class B {`	`class B { extends A {`
Page 58, first code block `echo '<a href=\"mailto:$result->email\">$result->name</a>";`	`print '<a href=\"mailto:$result->email\">$result->name</a>";`
Page 62, second line of first code block `$a->counter = 0;`	`$a->counter = 1;`
Page 63 SPL and **Interators**	SPL and **Iterators**
Page 65, sixth line of code `function hasMore();`	`function valid();`
Page 65, second line of second code block `$iter->hasMore();`	`$iter->valid();`
Page 66, twelfth line of code `function hasMore() {`	`function valid() {`

Page 67, second code block function **offsetGet**($name)	function **offsetGet**($name)
Page 67, next to last line of code function **offsetSet**($name, $value)	function **offsetSet**($name, $value)

Page 68, first six lines of code
function offsetUnset ($name)
{
return dba delete($name, $this->dbm);
}
return dba replace($name, serialize($value), $this->dbm);
}
	return dba replace($name, serialize($value), $this->dbm);
	}
	function offsetUnset ($name)
	{
	return dba delete($name, $this->dbm);
	}

Page 68, first code block function current() **{** function next() **{** function has More() **{** function key() **{**	function current() **{** function next() **{** function has More() **{** function key() **{**
Page 74, third code block <$	<$**php**

Page 79, first code block `echo "FATAL error $msg at $filename:$linenum ";` `echo "unknown error at $filename:$linenumb ";`	 `print "FATAL error $msg at $filename:$linenum ";` `print "unknown error at $filename:$linenumb ";`

Page 82, last code block
`echo "Your Web server user probably shouldn't be shelled.\m";`
 	
`echo "Great!\n";`
 	
`echo "An error occurred checking the user\n";`
	
	`print "Your Web server user probably shouldn't be shelled.\m";`
	
	`print "Great!\n";`
	
	`print "An error occurred checking the user\n";`

Page 87, both code blocks change all occurrences of **echo** to **print**	
Page 99, last line on page `header("Location: /login/php");`	 `header("Location: /login.php");`

Page 110, second block of code
`echo $ Get['name'];`
 	
`echo "Stranger";`
	
	`print $ Get['name'];`
	
	`print "Stranger";`

Page 111, middle of second code block `Hello <$php echo ?this-> tpl vars['name']; ?>`	 `Hello <$php print ?this-> tpl vars['name']; ?>`

Page 120, last code block
`<title><?php echo $template->title ?></title>`
 	
`Hello <?php echo $template->name ?>!`
	
	`<title><?php print $template->title ?></title>`
	
	`Hello <?php print $template->name ?>!`

Page 121, third code block `<head><title<>?php **echo** ?template->title ?></title>`	`<head><title<>?php **print** ?template->title ?></title>`
Page 144, code block two instances of **Reflection Class** should be **ReflectionClass**	
Page 149, Note The constructor uses the **Reflection Class** class to introspect the service and logger classes before you try to instantiate them.	The constructor uses the **ReflectionClass** class to introspect the service and logger classes before you try to instantiate them.
Page 156, second line of first full code block `require once 'PHPUnit/Framework/Test**Class**.php';`	`require once 'PHPUnit/Framework/Test**Case**.php';`
Page 156, first line of second full code block `require **omce** "PHPUnit/Framework/TestSuite";`	`require **once** "PHPUnit/Framework/TestSuite**.php**";`
Page 156, first code line after "After you have done this, you run the test:" `require once "PHPUnit/TextUI/TestRunner";`	`require once "PHPUnit/TextUI/TestRunner**.php**";`
Page 161, sixth code line from bottom `foreach**( get declared classes()** as $class) {`	`foreach**(get declared classes()** as $class) {`
Page 162, first line of code `foreach**( get declared classes()** as $class) {`	`foreach**(get declared classes()** as $class) {`
Page 165, immediately before "Using the seUp() and tearDown() Methods The following **examples** generates a success:	The following **example** generates a success:
Page 209, next to last line of code `return Commerce calculate**Tax**($user->state, $price);`	`return Commerce **calculateStateTax**($user->state, $price);`

Page 235, first line of code `Content-Encoding: gzip,defalte`	`Content-Encoding: gzip,deflate`
Page 243, code in Note `echo "Hello World";`	`print "Hello World";`
Page 244, code in Note at top of page `echo "Hello World";`	`print "Hello World";`
Page 264, next to last line of code `protected $expriration;`	`protected $expiration;`
Page 331 Checking That `$ Server[REMOTE IP]` Stays the Same	Checking That `$ Server['REMOTE IP']` Stays the Same
Page 335, fourth line of code `set cookie(self::$cookiename, $cookie);`	`setcookie(self::$cookiename, $cookie);`

Page 335, middle of page
`$cookie = implode($glue, $buffer);`
 	
`explode($glue, $buffer);`
	`$cookie = implode(self::$glue, $buffer);`
	
	`explode(self::$glue, $buffer);`

Page 395, first code block `echo "Error\n";`	`print "Error\n";`
Page 423, first paragraph Here is a sample run against my Web log, in which I've specified **10,000** requests with a concurrency of 100 requests.	Here is a sample run against my Web log, in which I've specified **1,000** requests with a concurrency of 100 requests.

Page 431, second paragraph in "Installing and Using APD" After **ADP** is installed, you should enable it by setting the following in your `php.ini` file:	After **APD** is installed, you should enable it by setting the following in your `php.ini` file:

Page 433, code in "A Tracing Example" section
`echo "Hello $name\n";`
 	
`echo "Goodbye $name\n";`
	`print "Hello $name\n";`
	
	`print "Goodbye $name\n";`

This errata sheet is intended to provide updated technical information. Spelling and grammar misprints are updated during the reprint process, but are not listed on this errata sheet.