



The PHP Company

PHP Arrays for RPG Programmers

Mike Pavlak
Solutions Consultant
mike.p@zend.com
(815) 722 3454



PHP Sessions

Session 1-9:00

- PHP101

Session 2-10:30

- PHP Arrays for the RPG Programmer

11:45

- Lunch

Session 3-12:30

- PHP Function Junction

Session 4-1:45

- PHP Toolkit Functions

Session 5-3:00

- Build your Intranet on IBM i for Free

4:00

- Goodies!

Agenda

- Introduce arrays in PHP
- Review RPG arrays
- Compare RPG and PHP array concepts
- More functions for arrays in PHP
- Q&A



Why are we talking about arrays?

- Fastest method for manipulating ordered sets
- Highly leveraged in PHP development
- PHP developers take them for granted
- Available in RPG but long neglected
- Gap that needs to be closed
- Array defined:

...a data structure consisting of a group of elements that are accessed by indexing

PHP Array Examples

Data Type Review: 8 Data Types

- *Scalar*
 - String "the quick brown fox...", '123456'
 - Integer 860, -9, 57009
 - Floating point 19.99, 29.99, 3.1412
 - Boolean true, false
- *Compound*
 - Array [0] => 0 [1] => 1 [2] => 1 [3] => 2 [4] => 3...
 - Object OOP
- *Special*
 - Resource Handle
 - Null Something that not nothing (empty set)

Three types of arrays

- Enumerated
 - Simple list

```
$arrayone = array("Scooby", "Shaggy", "Daphne",  
                 "Fred", "Velma");
```

- Associative
 - Custom key

```
$arraytwo = array(    Cartoon1=>'Scooby',  
                   Cartoon2=>'Shaggy',  
                   Cartoon3=>'Daphne',  
                   Cartoon4=>'Fred',  
                   Cartoon5=>'Velma'   );
```

- Multidimensional
 - Array of arrays

```
$arraythree = array(  
    array('Scooby', 'Shaggy', 'Daphne',  
         'Fred', 'Velma'),  
    array('Bugs', 'Daffy', 'Tweety',  
         'Elmer', 'Foghorn') );
```

Enumerated array

Code:

```
$arrayone = array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma');  
echo "<BR> <BR> Array one: "; print_r($arrayone);
```

Output:

```
Array one: Array ( [0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma )
```

Associative array

Code:

```
$arraytwo = array(Cartoon1=>'Scooby',  
                  Cartoon2=>'Shaggy',  
                  Cartoon3=>'Daphne',  
                  Cartoon4=>'Fred',  
                  Cartoon5=>'Velma' );  
echo "<BR> <BR> Array two: ";print_r($arraytwo);
```

Output:

```
Array two: Array ( [Cartoon1] => Scooby [Cartoon2] => Shaggy  
[Cartoon3] => Daphne [Cartoon4] => Fred [Cartoon5] => Velma )
```

If you have trouble, think CL command parameters: Keyword & Values!!!

Multidimensional array

Code:

```
$arraythree = array(  
    array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma'),  
    array('Bugs', 'Daffy', 'Tweety', 'Elmer', 'Foghorn')  
);  
  
echo "<BR> <BR> Array three: ";print_r($arraythree);
```

Output:

```
Array three: Array ( [0] => Array ( [0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma ) [1] => Array ( [0] => Bugs [1] => Daffy [2] => Tweety [3] => Elmer [4] => Foghorn ) )
```

Adding elements & growing the array

- PHP Arrays are dynamic
- Can be sized on the fly, no need to recompile
- Example adding element:

```
$arrayone = array('Anne', 'Mark', 'Sabrina', 'Katie', 'Rick');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
$arrayone[] = 'Joe';  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
|
```

```
Array one: Array ( [0] => Anne [1] => Mark [2] => Sabrina [3] => Katie [4] => Rick )
```

```
Array one: Array ( [0] => Anne [1] => Mark [2] => Sabrina [3] => Katie [4] => Rick [5] => Joe )
```

Removing elements & reducing the array

- `array_pop` removes element from the end
- `unset` removes an element you specify (or entire array!)

```
$arrayone = array('Anne', 'Mark', 'Sabrina', 'Katie', 'Rick');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
array_pop($arrayone);  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
unset($arrayone[2]);  
echo "<BR> <BR> Array one: "; print_r($arrayone);
```

```
Array one: Array ( [0] => Anne [1] => Mark [2] => Sabrina [3] => Katie [4] => Rick )
```

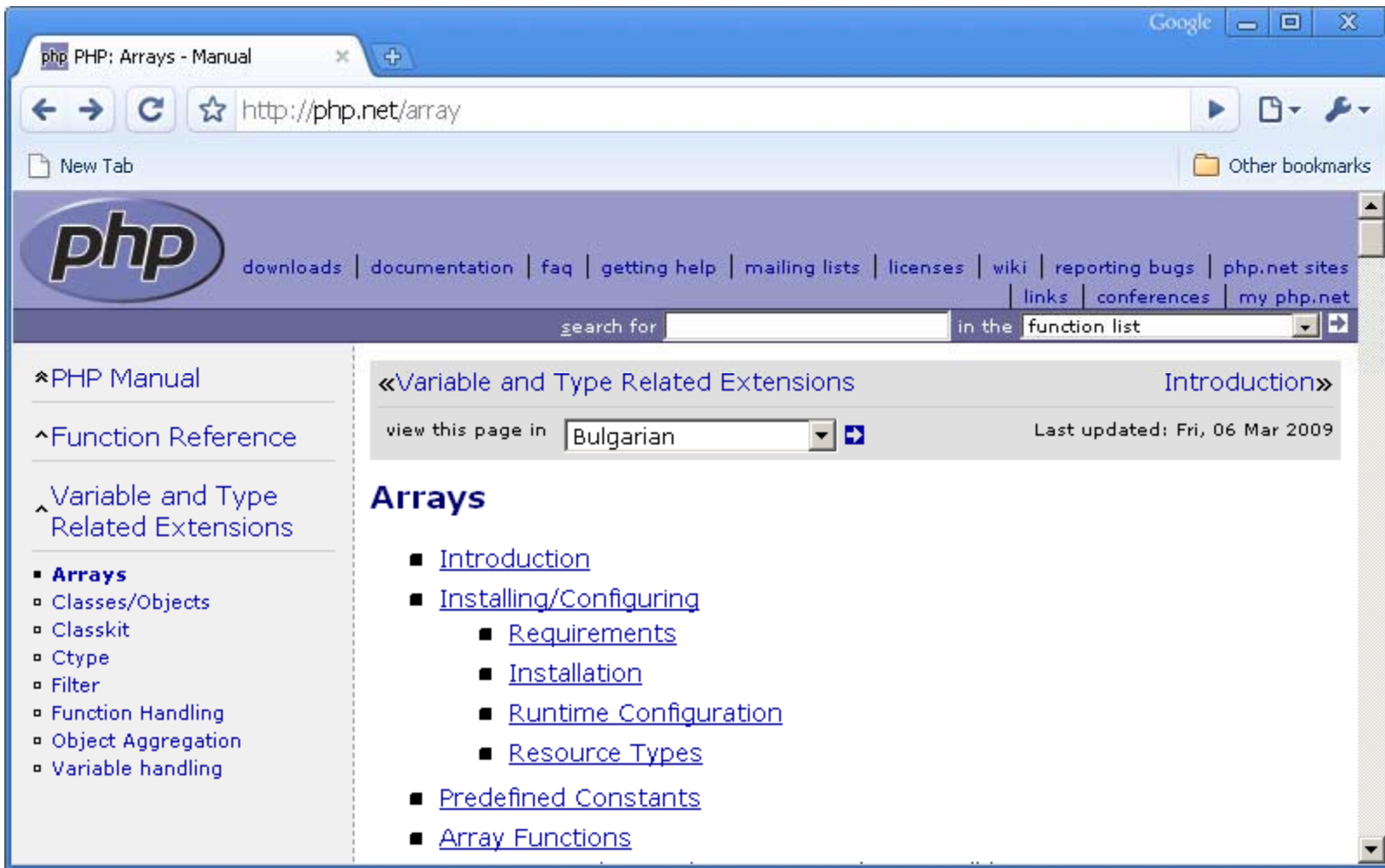
```
Array one: Array ( [0] => Anne [1] => Mark [2] => Sabrina [3] => Katie )
```

```
Array one: Array ( [0] => Anne [1] => Mark [3] => Katie )
```

Trivia points

- Really only one type of array, associative
- Data content is non-restrictive, any data types
- Each element can be different
- Array sizes change dynamically
- Supports no known limit of dimensions
 - ▶ Memory
 - ▶ Humans like 2 or 3 (Think spreadsheet and workbook)
- Used heavily in i/o
- Both keys and content can be dynamic
- Index starts at zero while RPG starts at one

Got Doc? php.net/array



Review RPG Arrays

In the beginning...

- Indicators were the only ordered set
 - Original RPG and RPG II

Name	Indicators	Notes
Numbered	*IN01-*IN99	Gen purpose
Command Key	*INKA - *INKY	No "O"
Halt	H1-H9	Error recovery
Matching	M1-M9, MR	Matching records
Control	L1-L9	Level Breaks
External	U1-U8	Switches
Cycle	1P, LR, OA-OG, OV	Printing

And then...

- RPG II - Then came simple arrays.
 - Predefined length
 - Single variable data type
 - Built in E-specs
- Op Codes
 - XFOOT - Summing array
 - MOVEA - Move data (Still most extremely powerful)
 - LOKUP - Search the array
 - SORTA - Gee, I wonder what this does?
- Seems like things paused here for a while

Today...

- Compile time tables
 - Great for static content
 - Defined below “O” specs
 - Two dimensional in nature
- RPG III - Multiple Occurrence Data Structure (MODS)
 - Two dimensional feel
 - Still a little clunky
- RPG IV - More Power!
 - **V5R1** - BIF's : %LOOKUP, %LOOKUPGT, etc.
 - **V5R2** - DIM for Data Structures; MODS on Steroids!
 - **V5R3** - %SUBARR is an attempt at dynamic sizing
 - **V5R4** - XML processing
 - **i6.1** - DIM up to 16,773,104

How PHP matches up to RPG

Array shootout

- Base functions
 - RPG has about a dozen op-codes and BIF's (Variations on BIF's)
 - Many op-codes can manipulate array content
 - PHP has 75 functions www.php.net/array
- Size
 - RPG has limits, 16,773,104 as if i6.1
 - PHP has no practical limits, No "array index overflow" error
 - RPG array must be defined, PHP grows dynamically
- Type
 - RPG uses static typing (one type, one length)
 - PHP is dynamically typed (Each element can be different)

Simple Array Search (Lookup)

```
$arrayone = array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma');
```

RPG

```
D ARRAYONE      S           8      DIM(5)
D              CTDATA PERRCD(1)
D X             S           1  0  INZ
/FREE
                X=LOOKUP (' DAPHNE' :ARRAYONE) ;
/END-FREE
```

```
I found her in position==> 3
```

PHP

```
$x=array_search('Daphne', $arrayone);
echo "<BR> <BR> I found her in position==> $x";
```

```
I found her in position==> 2
```

Simple traverse

```
$arrayone = array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma');
```

RPG

```
for X=1 to (%ELEM(ARRAYONE));  
    except REC1;  
endfor;
```

```
SCOOPY is the index value 1  
SHAGGY is the index value 2  
DAPHNE is the index value 3  
FRED is the index value 4  
VELMA is the index value 5
```

PHP

```
foreach ($arrayone as $key => $x) {  
    echo "<BR>$x is the index value " . $key;  
}
```

```
Scooby is the index value 0  
Shaggy is the index value 1  
Daphne is the index value 2  
Fred is the index value 3  
Velma is the index value 4
```

RPG to PHP function map

Function	RPG	PHP	Notes
Search	%LOOKUP	array_search	
Sum	%XFOOT	array_sum	Array_prod can multiply
Get portion	%SUBARR	array_slice	Substring an array by chunks
Sort	SORTA	asort, arsort	PHP sequence dynamic
Move	MOVEA	array_slice	Substring by character
Count	%ELEM	count	Get number of elements

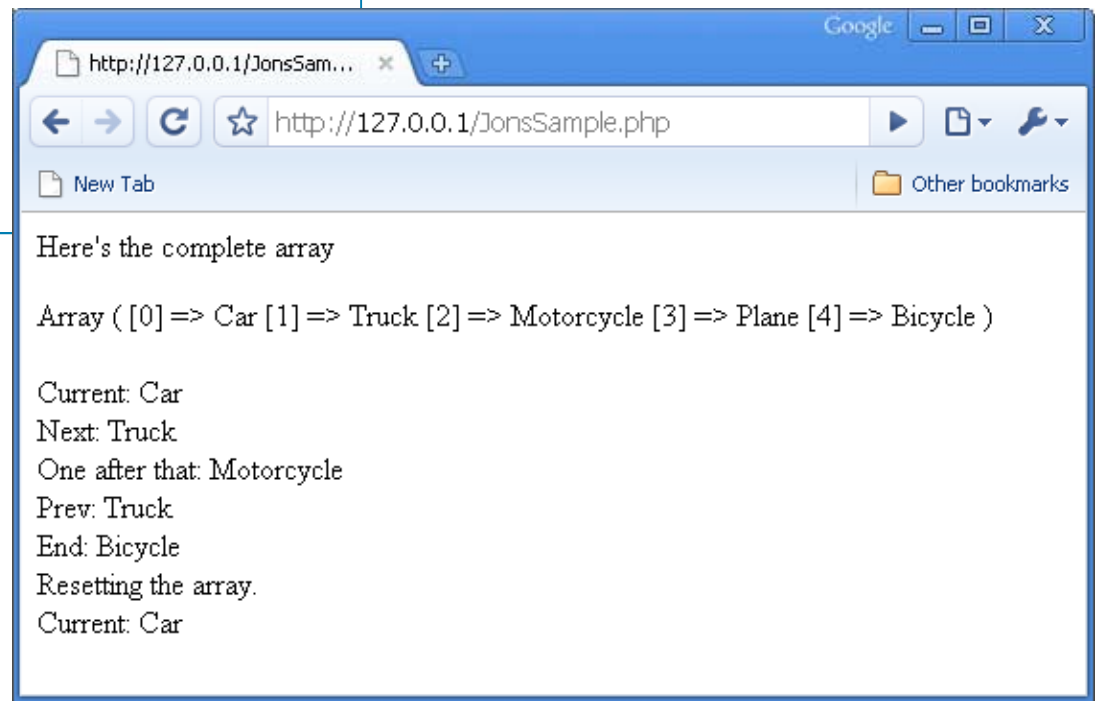
More functions in PHP

Interesting functions

- How to move around the array
- Randomize contents
- Array housekeeping
- Move array elements to variables
- Sort two or more arrays at once
- Execute a function on each element with no loop!
- Data file example

Navigate the array...Thanks Jon!

```
$myArray = array("Car", "Truck", "Motorcycle", "Plane", "Bicycle");  
echo "<p>Here's the complete array </p>";  
print_r($myArray);  
echo "<br><br>";  
echo "Current: ", current($myArray), "<br>";  
echo "Next: ", next($myArray), "<br>";  
echo "One after that: ", next($myArray), "<br>";  
echo "Prev: ", prev($myArray), "<br>";  
echo "End: ", end($myArray), "<br>";  
echo "Resetting the array.<br>";  
reset($myArray);  
echo "Current: ", current($myArray), "<br>";
```



Mix it up with a shuffle

```
$arrayone = array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
shuffle($arrayone);  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
shuffle($arrayone);  
echo "<BR> <BR> Array one: "; print_r($arrayone);
```

```
Array one: Array ( [0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma )  
  
Array one: Array ( [0] => Shaggy [1] => Daphne [2] => Velma [3] => Scooby [4] => Fred )  
  
Array one: Array ( [0] => Fred [1] => Shaggy [2] => Daphne [3] => Scooby [4] => Velma )
```

Consolidate, clean and sort arrays

```
$arrayone = array('Scooby', 'Shaggy', 'Daphne', 'Fred', 'Velma', 'Jerry');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
$arraytwo = array('Bugs', 'Daffy', 'Tweety', 'Elmer', 'Foghorn', 'Jerry');  
echo "<BR> <BR> Array two: "; print_r($arraytwo);  
$both = array_merge($arrayone, $arraytwo);  
echo "<BR> <BR> Array both: "; print_r($both);  
$both = array_unique($both);  
echo "<BR> <BR> Array unique: "; print_r($both);  
asort($both);  
echo "<BR> <BR> Array sort: "; print_r($both);
```

Array one: Array ([0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma [5] => Jerry)

Array two: Array ([0] => Bugs [1] => Daffy [2] => Tweety [3] => Elmer [4] => Foghorn [5] => Jerry)

Array both: Array ([0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma [5] => Jerry [6] => Bugs [7] => Daffy [8] => Tweety [9] => Elmer [10] => Foghorn [11] => Jerry)

Array unique: Array ([0] => Scooby [1] => Shaggy [2] => Daphne [3] => Fred [4] => Velma [5] => Jerry [6] => Bugs [7] => Daffy [8] => Tweety [9] => Elmer [10] => Foghorn)

Array sort: Array ([6] => Bugs [7] => Daffy [2] => Daphne [9] => Elmer [10] => Foghorn [3] => Fred [5] => Jerry [0] => Scooby [1] => Shaggy [8] => Tweety [4] => Velma)

Sort Multiple Arrays at once!

```
$arrayone = array('Dog', 'Cat', 'Chicken', 'Dolphin');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
$arraytwo = array('Puppy', 'Kitten', 'Chick', 'Pup');  
echo "<BR> <BR> Array two: "; print_r($arraytwo);  
$arraythree = array('Litter', 'Kindle', 'Peep', 'Pod');  
echo "<BR> <BR> Array two: "; print_r($arraythree);  
array_multisort($arrayone,$arraytwo,$arraythree);  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
echo "<BR> <BR> Array one: "; print_r($arraytwo);  
echo "<BR> <BR> Array one: "; print_r($arraythree);
```

```
Array one: Array ( [0] => Dog [1] => Cat [2] => Chicken [3] => Dolphin )
```

```
Array two: Array ( [0] => Puppy [1] => Kitten [2] => Chick [3] => Pup )
```

```
Array two: Array ( [0] => Litter [1] => Kindle [2] => Peep [3] => Pod )
```

```
Array one: Array ( [0] => Cat [1] => Chicken [2] => Dog [3] => Dolphin )
```

```
Array one: Array ( [0] => Kitten [1] => Chick [2] => Puppy [3] => Pup )
```

```
Array one: Array ( [0] => Kindle [1] => Peep [2] => Litter [3] => Pod )
```

Manipulate all elements of an array

```
$arrayone = array('Dog', 'Cat', 'Chicken', 'Dolphin');  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
strtoupper($arrayone);  
echo "<BR> <BR> Array one: "; print_r($arrayone);  
  
function upperCase(&$animal, $key) {  
    $animal=strtoupper($animal);  
}  
  
array_walk($arrayone, upperCase);  
echo "<BR> <BR> Array one: "; print_r($arrayone);
```

```
Array one: Array ( [0] => Dog [1] => Cat [2] => Chicken [3] => Dolphin )
```

```
Array one: Array ( [0] => Dog [1] => Cat [2] => Chicken [3] => Dolphin )
```

```
Array one: Array ( [0] => DOG [1] => CAT [2] => CHICKEN [3] => DOLPHIN )
```

Get data from a file

```
while($row=db2_fetch_array($stmt)) {  
    print_r($row); echo "<BR>";  
    list( $CUSTOMER_NUMBER, $CUSTOMER_NAME, $CUSTOMER_STATE)= $row;  
  
    echo("<TR><TD> $CUSTOMER_NUMBER</TD> <TD>$CUSTOMER_NAME </TD>  
        <TD> $CUSTOMER_STATE</TD>"); }  
}
```

- Loop through data
- List function copies to variables
- Implicit copy, be careful
- Arrays in PHP like Data Structures in RPG: The workhorse of data manipulation!

```
Array ( [0] => 1 [1] => Jimmy Buffet [2] => IL )  
Array ( [0] => 2 [1] => Sherlock Holmes [2] => EN )  
Array ( [0] => 3 [1] => Gregory House [2] => NJ )  
Array ( [0] => 4 [1] => Dexter Morgan [2] => FL )  
Array ( [0] => 5 [1] => Mary Shannon [2] => NM )
```

Customer Number	Customer Name	State
1	Jimmy Buffet	IL
2	Sherlock Holmes	EN
3	Gregory House	NJ
4	Dexter Morgan	FL
5	Mary Shannon	NM

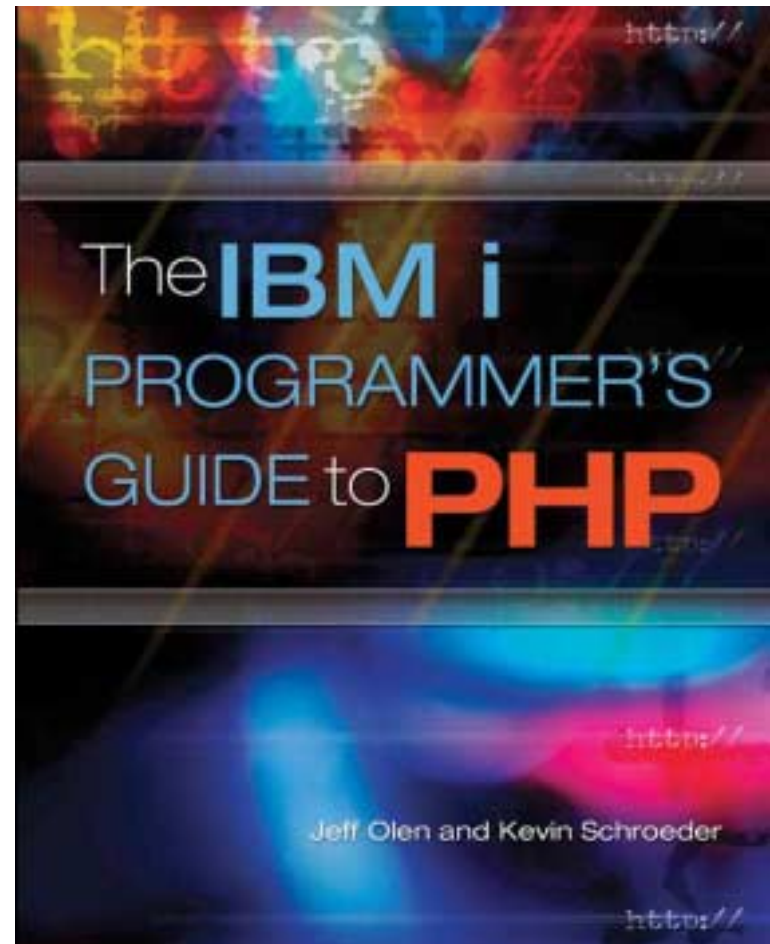
New book, new printing, same great stuff!

Kevin Schroeder from Zend's
Global Services Group

with

Jeff Olen, co-author of...

Get yours at MCPressonline
or at fine bookstores everywhere



Q&A

www.zend.com

mike.p@zend.com