



The PHP Company

# Function Junction

Mike Pavlak  
Solutions Consultant  
[mike.p@zend.com](mailto:mike.p@zend.com)  
(815) 722 3454



## PHP Sessions

Session 1:30

- PHP101

Lab 3:00

- Lab

Session 3:00

- Zend Framework

Lab 4:45

- Lab Repeated

Session 4:45

- Function Junction

Dinner 6:15

- Dinner Break

Session 7:15

- Build your Intranet on IBM i for Free

# Agenda

---

- What are functions in PHP?
- Base functions in PHP
- Home grown function
- Data access function
- ITK - Toolkit Library
- Q&A

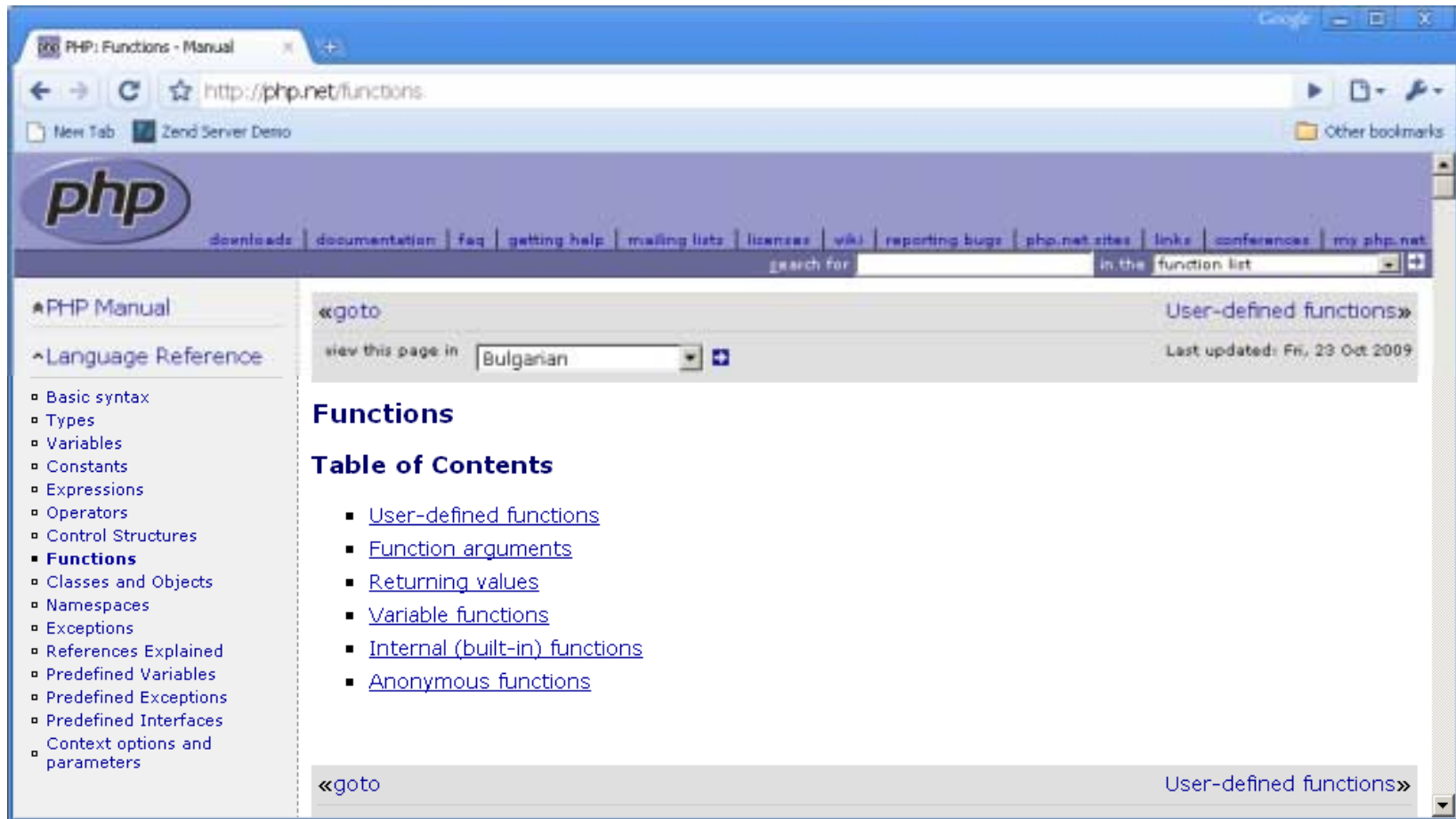


# What are functions in PHP?

---

- What's so great about functions?
- Remember life before subroutines?
- Need to organize code
- Before OOP, functions was all they had!
- Also the basis for language
- PHP supports more than 5000 functions
- And you can create them!
- [www.php.net/functions](http://www.php.net/functions)

# http://php.net/functions



# More on functions...

---

- Functions come from 1 of 3 places
  - Built-in
    - Part of base PHP
  - Extensions
    - Components of modules like image functions in GD
  - User defined
    - You will create these!
- You may have been using functions...
  - `db2_connect()`
  - `print_r`
  - `strtoupper`
  - Etc.

# Function fundamentals

---

- Can accept parameters
- Will always return a value
- Maintain scope of control within
  
- Echo is a construct, not a function
  - ▶ Does not return a value

# Function Junction

Base PHP Functions

# What do functions look like?

---

## Function call has three major parts:

Function Name - Required

Parameters - Optional

Return Value – Always returned, default null



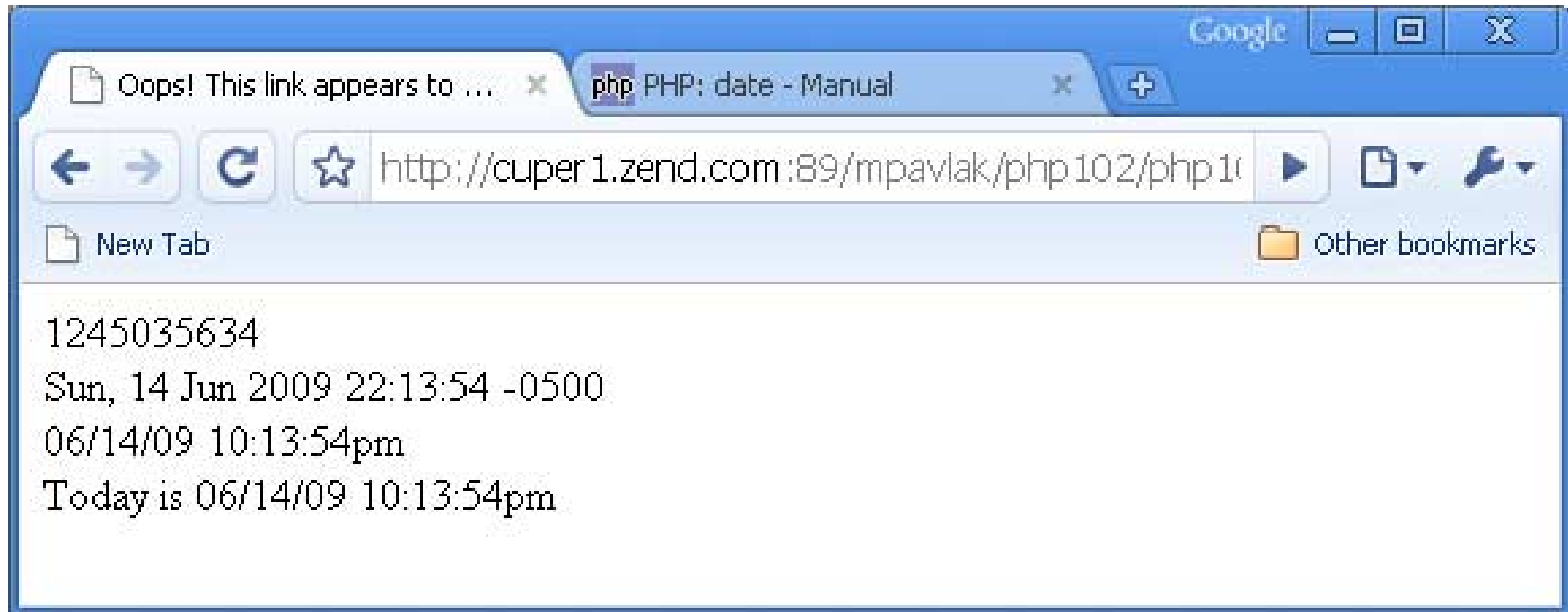
# Functions in action - Simple

```
$UserID = 'MPavlak';  
$Pwd = '*****';  
  
echo $UserID . "<br><br>";  
$UserID = strtoupper($UserID);  
echo $UserID;
```



## How do you get a date?

```
$currentTime = time(); print_r($currentTime); echo "<br>";  
$currentTime = date(r); print_r($currentTime); echo "<br>";  
$currentTime = date("m/d/y g:i:sa"); print_r($currentTime); echo "<br>";  
echo "Today is $currentTime";
```



## More built in functions...

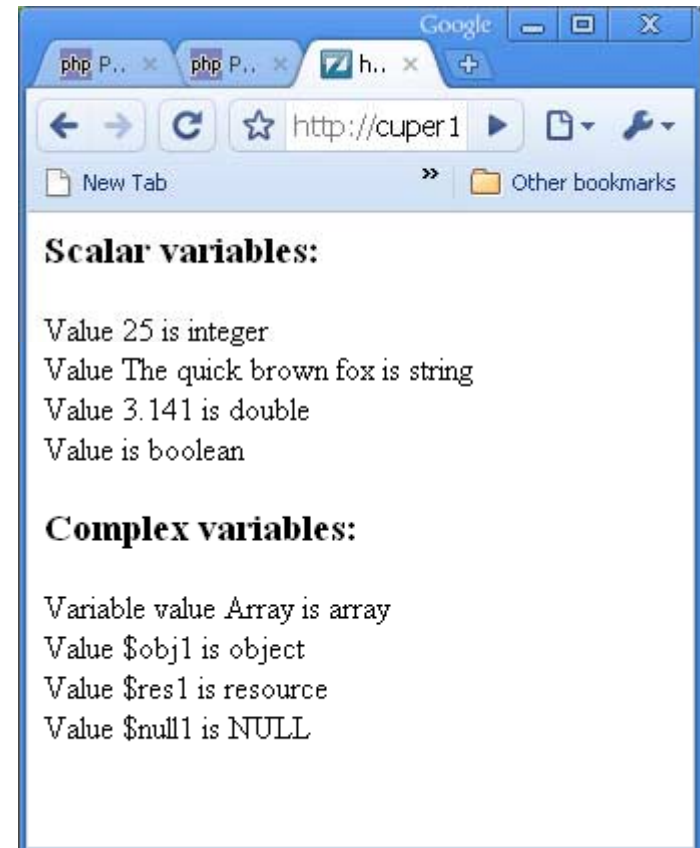
---

- About 5,000 in base and extensions
- Gettype - Can return the variable type but...
  - ▶ Can give you incorrect info like String 0 returns Bool
- Use `is_*` functions

<code>is_array()</code>	<code>is_object()</code>
<code>is_bool()</code>	<code>is_resource()</code>
<code>is_int()</code>	<code>is_scalar()</code>
<code>is_numeric()</code>	<code>is_string()</code>

# Gettype...

```
3 // Scalar
4 $Int1 = 25; $str1 = 'The quick brown fox';
5 $flt1 = 3.141; $bool1=false;
6
7 // Complex
8 $arr1 = array('Dog','Cat'); $obj1 = new stdClass();
9 $res1 = i5_connect("", "PHPUSER","phpuser"); $null1 = null;
10
11 echo "<h3>Scalar variables:</h3>";
12 echo "Value $Int1 is " . gettype($Int1);
13 echo "<br>Value $str1 is " . gettype($str1);
14 echo "<br>Value $flt1 is " . gettype($flt1);
15 echo "<br>Value $bool1 is " . gettype($bool1);
16
17 echo "<h3>Complex variables:</h3>";
18 echo "Variable value $arr1 is " . gettype($arr1);
19 echo '<br>Value $obj1 is ' . gettype($obj1);
20 echo '<br>Value $res1 is ' . gettype($res1);
21 echo '<br>Value $null1 is ' . gettype($null1);
```



# Function Junction

User Defined Functions

# User Defined Functions

---

- PHP version of sub-procedure (subroutine)
- Programmer defines action, input and results
- This is how applications are built
- Modular components, pulled together

# Rules for creating functions

---

- See [php.net](http://php.net) for more details
  - Must have name
  - Parameters are optional, you test inside function
  - Function always returns a value, even if you don't
  - Function variables have scope
  - Builds suspense for OOP, but not yet...

# Let's create a function

---

- Create a pricing function
  - Called with a price & quantity
  - Will add standard discount based on quantity
    - Qty 1-10 gives 10%
    - Qty 11-50 gives 15%
    - Qty > 50 gives 20%
  - Return adjusted price

## Code for the function

```
function customer_price( $price, $quantity) {  
    switch ($quantity) {  
        case ($quantity > 50):  
            $discount = .2;  
            break;  
        case ($quantity > 10):  
            $discount = .15;  
            break;  
        default:  
            $discount = .1;  
    }  
    $newPrice = $price * (1-$discount);  
    return $newPrice;  
}
```

## Test the function

```
$p1 = 5.00; $q1=15; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 0.00; $q1=15; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 12.00; $q1=9; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 10.00; $q1=800; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";
```



# Change the spec...

---

- What if I do not pass a quantity?
  - Ask the BA...
  - BA states that you should default to retail
  - OK, let's go change the code!

# Modified code

```
function customer_price( $price, $quantity="zero") {
    switch ($quantity) {
        case ($quantity == "zero"):
            $discount = 0;
            break;
        case ($quantity > 50):
            $discount = .2;
            break;
        case ($quantity > 10):
            $discount = .15;
            break;
        default:
            $discount = .1;
    }
    $newPrice = $price * (1-$discount);
    return $newPrice;
}
```

## Test the mod...

```
$p1 = 5.00; $q1=15; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 0.00; $q1=15; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 12.00; $q1=9; $output=customer_price($p1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";  
$p1 = 10.00; $q1=800; $output=customer_price($p1,$q1);  
echo "Retail is $p1 and quantity is $q1 with discount price at $output<br>";
```



# Scope

---

- **Variables have scope, much like RPG ILE**
  - Function keeps its own set of variables
  - Global keyword can change all that (RPG III)
  - Better to pass, globals can't be trusted
- **Static variables**
  - Persistence from call to call

# Parameters

---

- **Default**
  - Assign value in interface
- **Pass by value**
  - This is the default behavior
- **Pass by reference**
  - This is done with & before the variable in the interface.
- **Variable number**
  - `func_get_args()`
  - `func_num_args()`
  - `func_get_arg(argument_#)`

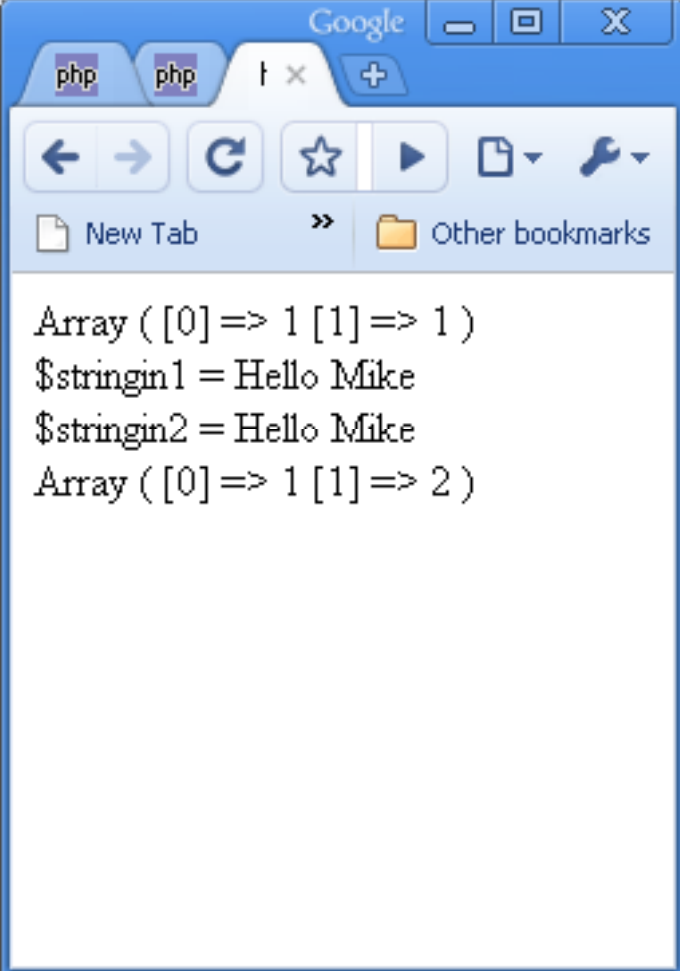
# Return

---

- Can only return one value
- If you want to return more than one, try array!
  
- Let's look at an example that shows all of these...

## A little of everything...

```
3 function example ($str1, &$str2) {
4     $a = 0;     static $b;
5     $a++; $b++;
6     $str2 = $str1;
7     $c = array($a, $b);
8     return $c;
9 }
10
11 $stringin1 = 'Hello Mike'; $stringin2 = 'Hello Scott';
12
13 $result = example($stringin1, $stringin2);
14
15 print_r($result);
16 echo '<br>$stringin1 = ' . $stringin1;
17 echo '<br>$stringin2 = ' . $stringin2;
18
19 $stringin1 = 'Hello Jane'; $stringin2 = 'Hello Laura';
20
21 echo '<br>';
22 $result = example($stringin1, $stringin2);
23 print_r($result);
```



The screenshot shows a web browser window with two tabs labeled 'php'. The address bar contains 'Google'. The browser's navigation bar includes back, forward, refresh, star, and play buttons. Below the navigation bar, there are two bookmarks: 'New Tab' and 'Other bookmarks'. The main content area of the browser displays the output of the PHP code from the left panel, showing the results of the 'example' function and the state of the variables.

```
Array ( [0] => 1 [1] => 1 )
$stringin1 = Hello Mike
$stringin2 = Hello Mike
Array ( [0] => 1 [1] => 2 )
```

# Function Junction

Data Access Function

- 
- Build PHP script using functions
  - Pass values rather than use global variables
  - Analyze function activity with Code Profiler

# Data Access in Functions

---

- Build PHP script using functions
- Pass values rather than use global variables
- Analyze function activity with Code Profiler

# Raw Data in DB2

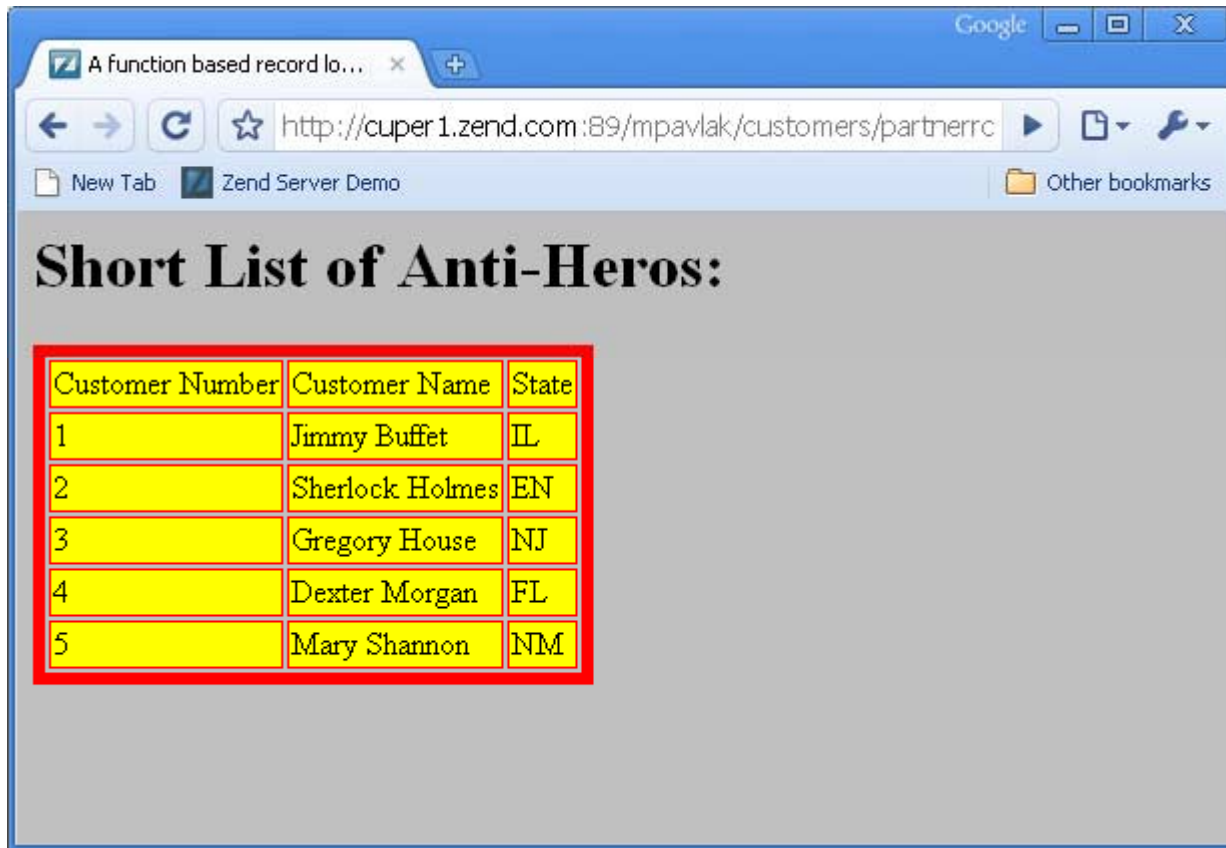


The screenshot shows a terminal window titled "cuper1.zend.com - Mocha W32 TNS250". The window contains a "Display Data" command output. The output shows a table with three columns: "Customer Number", "Customer Name", and "Customer State". The data is as follows:

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

The output ends with "\*\*\*\*\* End of data \*\*\*\*\*". At the bottom of the terminal, there are function key shortcuts: F3=Exit, F12=Cancel, F19=Left, F20=Right, and F24=More keys. The status bar at the bottom right shows "ONLINE", "3,32", and "M".

# Web Page



# Function 1 - Database connect

```
1 <html>
2 <head>
3 <TITLE>A function based record lookup</TITLE></head><body BGCOLOR=SILVER></body>
4 <h1>Short List of Anti-Heros:</h1>
5
6 <?php
7
8 function Connect() {
9     // Standard DB connection to DB2...
10
11     $conn = "LOCAL"; $name = ""; $pwd = "";
12     $i5link = db2_connect($conn, $name, $pwd);
13     if (!$i5link)
14         echo 'Connection failed: '.db2_stmt_error().': '.db2_stmt_errormsg();
15
16     return $i5link;
17 }
```

## Function 2 - Query execute

---

```
19 function GetData($i5link) {
20
21
22     $sql = "SELECT customer_number, customer_name, customer_state from zenddata.customer";
23     $stmt = db2_exec($i5link,$sql)
24         or die("Failed query:".db2_stmt_error().":".db2_stmt_errormsg());
25
26     return $stmt;
27 }
28
```

## Function 3 - Read and load data

---

```
29 function listData($stmt) {
30     // loop di loop through customer recs...
31     while($row=db2_fetch_array($stmt)) {
32         //print_r($row); echo "<BR>";
33         list( $CUSTOMER_NUMBER, $CUSTOMER_NAME, $CUSTOMER_STATE)= $row;
34         //include "somefilethatdoesnotexist.inc";
35         echo("<TR><TD> $CUSTOMER_NUMBER</TD> <TD>$CUSTOMER_NAME </TD>
36             <TD> $CUSTOMER_STATE</TD>"); }
37 }
```

# Main line component

---

```
40 $connection = Connect();
41 $resultset = GetData($connection);
42 // Setup table header...
43 echo '<TABLE BORDER="6" BORDERCOLOR=RED><TBODY BGCOLOR="yellow">';
44 echo '<TR><TD>Customer Number</TD><TD>Customer Name</TD><TD>State</TD></TR>';
45 listData($resultset);
46
47
48 echo '</table>';
49 db2_close($connection);
50 ?>
51 </html>
```

# Function Junction

ITK Class Library - Kevin Schroeder

# A bit about me

---

- **Kevin Schroeder**
  - ▶ Technology Evangelist - Zend Technologies
  - ▶ Blog: <http://www.eschrade.com>
  - ▶ Email: [kevin@zend.com](mailto:kevin@zend.com)

# ITK

---

- **What is it?**
  - ▶ An easy to use interface into the i5 PHP Toolkit
  - ▶ Fully unit tested
- **Why use it?**
  - ▶ It drastically simplifies accessing RPG functionality in PHP
- **What's it cost?**
  - ▶ Nothing
- **Where do I get it?**
  - ▶ <http://www.github.com/...>

---

# How does one currently access RPG functionality in PHP?

---

# How can one now access RPG functionality in PHP?

# Concepts

---

- **Adapters**
  - ▶ Represents the interface to the functionality be accessed
- **Programs**
  - ▶ Represents the program definition
- **Unit Tests**
  - ▶ Mechanisms for testing whether or not program functions are properly handling RPG-based data

# Adapters

---

- **Two adapters are included**
  - ▶ Live
    - Represents an individual connection to the PHP Toolkit implementation
    - Used in production scenarios
  - ▶ Mock
    - Used for Unit Testing
    - Will return specific results for specific calls
    - Allows you to test that your class is properly handling data
- **A Program class can be called regardless of the adapter being used**
- **Additional adapters can easily be written w**

# Programs

---

- Based off of the `Itk_PgmAbstract` class
- Definitions required

- ▶ Program Name

- The name of the RPG program being called

```
protected $_programName = 'ZENDSVR/COMMONPGM';
```

- ▶ Description

- Describes the input/output parameters

```
protected $_description = array(  
    'CODE'=> array(  
        self::DESC_IO=> I5_INOUT,  
        self::DESC_TYPE=> I5_TYPE_CHAR,  
        self::DESC_LENGTH=> "10"  
    )  
);
```

# Unit Testing

---

- **The entire library is designed to be easily testable**
  - ▶ Repeatability in testing is KEY to not losing your hair
  - ▶ Well written tests ensure that changes to not break compatibility
- **Unit Tests can be run using**
  - ▶ Live adapter on i Series machines
  - ▶ Mock adapter on either
    - i Series machines (if you don't want to change data)
    - Linux/Windows workstations
- **Unit Tests are run using PHPUnit**

# Recommendations

---

- Use a bootstrap to define the adapter at the front of the request
- Keep as much code out of your HTML as possible
- If you are not familiar with OOP this is a good place to start
  - ▶ Small example
  - ▶ Follows good practices
  - ▶ Demonstrates the benefits on a small scale

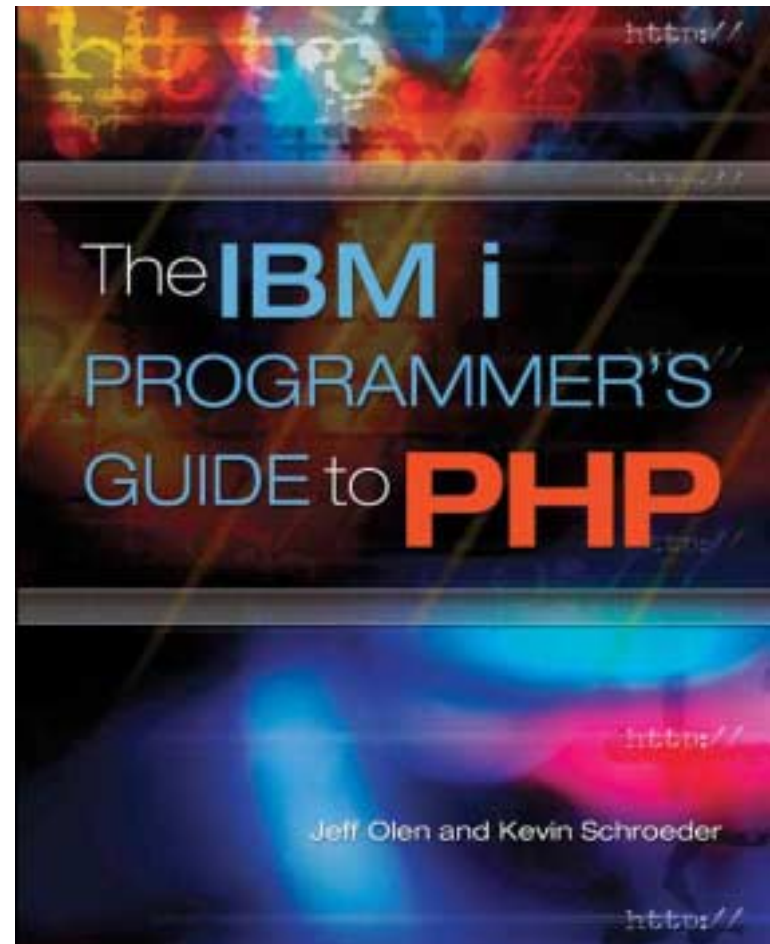
# New book, new printing, same great stuff!

Kevin Schroeder from Zend's  
Global Services Group

with

Jeff Olen, co-author of...

Get yours at MCPressonline  
or at fine bookstores everywhere



# Questions?

Thank you!!

[Mike.p@zend.com](mailto:Mike.p@zend.com)