PHP for zOS

Share Tech Conference August 2010

Wayne Duquaine

Grandview Systems

Phone: 707-829-9633
E-mail: grandvu@sonic.net

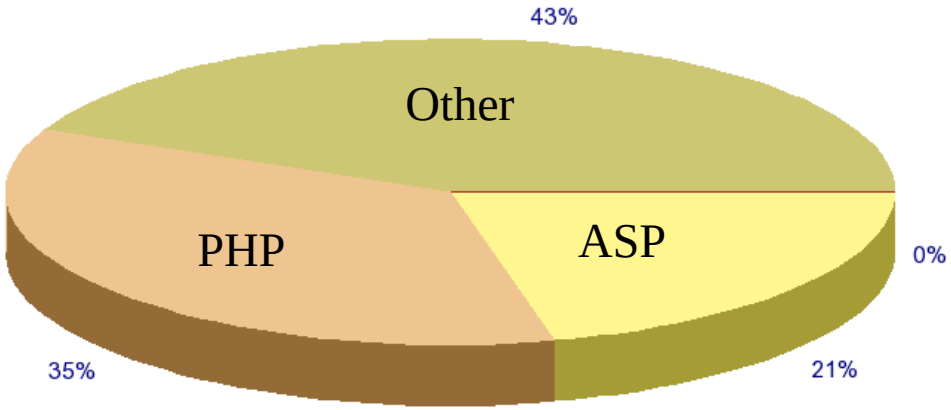
Outline

- Quick Overview of PHP
- Quick Overview of IBM's PHP Support on z/OS
- Language - Part 1 – Basic Syntax
- Language – Part 2 – Functions, Scripts
- Constructing a Simple PHP Application

PHP Dominates the Web

- PHP dominates the Server Side Web tools/apps market
  - » Is THE most popular web server language for creating Web sites
  - » 34+ % of all Web applications use PHP

- PHP runs on 22 million registered web domain servers
  - » Over 2.5 million PHP programmers



Source: www.nexen.net

What is PHP

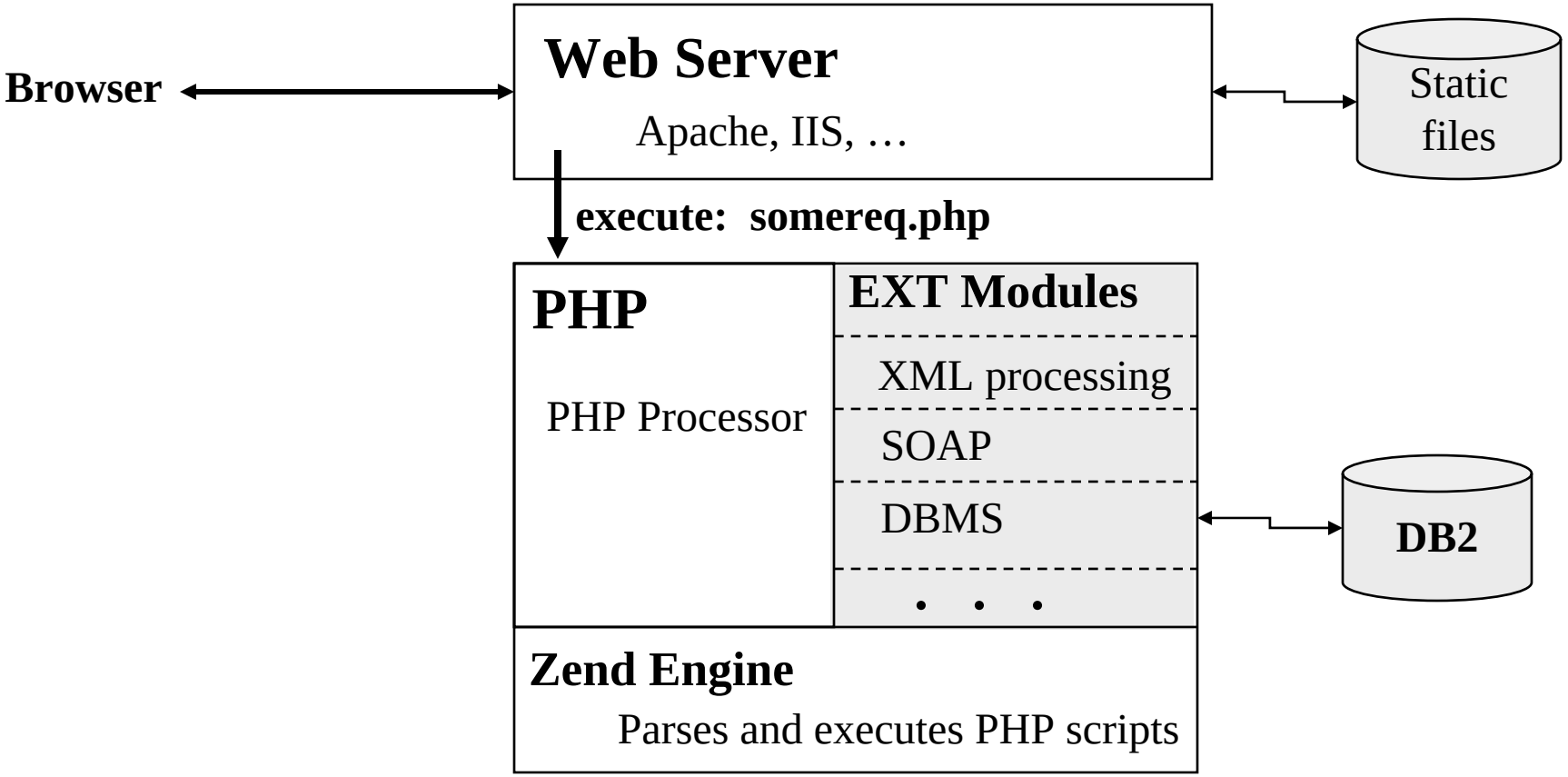
- PHP is a server-side language
  - » a **high-speed script processor** that is used to quickly create Web Pages, process XML, and access Databases
  - » Runs inside the Web Server

- Unlike Java or other interpretive language, PHP's focus is to use the script to invoke "native" C subroutines that do the actual work at full speed, i.e.
  - » PHP script provides If/Then/Else/While control logic to invoke native C functions that do the actual work.
    (versus Java where everything is interpreted). The script invokes the appropriate C subroutine to handle the request
  - » Most PHP implementations provides "caching" support for scripts, so that they are parsed only once.
    - – Result is near native speeds, e.g.
    - – Can scale to 130 million PHP requests per day (CommunityConnect)

Why use PHP

- Short Learning Curve
  - » PHP is faster and easier to learn than any other Web programming language
- Quick Development Time
  - » Scripts are 5 – 10 times more productive than hand coding a Server application (Java or C) from scratch.
- Fast Performance
- Easily integrates DBMS processing with HTML/XML processing

High Level View: PHP as Web PlugIn



The EXT modules are all "native" C subroutines, that are called by the PHP processor as it executes the .php script. These are the things that do the actual work (read from a database or file, execute a SOAP request, …)

PHP Operation

- PHP operates in one of two modes:

  » As an “embedded” plug-in that runs inside a Server
  - **Apache**
  - **IIS**
  - **CICS**

  » As a “stand-alone” utility that can be executed under USS command line
  (like a REXX or PERL script)

PHP Operation on z/OS

- IBM provides two versions of PHP on the mainframe:
  - » A standard “C” based version that is built using the standard PHP C source code. It is complaint with PHP 5.2.
  - » It runs as a “Command Line” (CLI) version under USS.
    Provides same level of support as any of the CLI versions of PHP that run on PCs or Linux.
  - » Is available free from IBM.
  - » A “Java” based version of PHP, that IBM created, that re-implements PHP using Java instead of C. It is compatible with PHP 5.2
  - » It runs as a plug-in for both CICS and for WebSphere.
  - » Is the basis for IBM’s new sMASH technology.
    - – Development has been spun off as a separate organization: “projectzero,org”

PHP Operation on CICS - 1

- Uses the Java Based PHP plug-in
- Supports PHP 5.2 and most of the major PHP libraries
- Comes with LOTs of tools, including DoJo and sMash framework
- All of the PHP language facilities covered in this presentation are supported by CICS's PHP support
- DBMS support to DB2 is provided via standard PHP PDO objects

PHP Operation on CICS - 2

- For debug and testing, they also provide a CLI tool that you can use to invoke PHP scripts under CICS
  - » Uses EXCI to communicate to the PHP script processor under CICS
- All PHP scripts are kept in zFS on USS
- Can invoke COBOL/PL1/C/ASM CICS Apps via PHP LINK call.
  - » PHP provides a “wrapper class” that is used to access/update fields in the COMMAREA.
  - » The PHP wrapper class invokes a set of Java “accessor” classes that were generated from the COMMAREA ADATA descriptors
  - » Tooling for generating the Java “accessor” classes is provided by JZOS
- Can extend CICS PHP support via IBM supplied “Java/PHP Bridge” support

PHP Language – Part 1

- Language Overview
  - » Syntax is very similar to C / Java / Javascript languages
    - – If / Then / Else
    - – While and Do While
    - – For loops
    - – Switch / Case statements
    - – Semi-colon ; is used to separate statements
    - – Can use # or // or /* */ for comments
    - – { } are used to group statements into a block
    - – [ ] are used for array subscripts

PHP Language Basics

- Variables
- Operators
- Expressions
- Flow of Control Constructs
- Comments

PHP Language - Variables

- Variable names always start with $ followed by 1-n alphanumerics, e.g. $abc456

- Variable names are case sensitive.
  $abc is not the same variable as $ABC

- Is "loosely" typed language
  - » No explicit int/float/string/boolean data type notation.
  - » PHP automatically derives the data type based on what it parses:
    - – Strings with quotes (‘ or “) = char string
    - – Number without decimal point = integer
    - – Number with decimal point = floating point number

- Has conversion and casting functions to convert
  e.g. convert a char string to int/float or vice versa:
  $b = (int) $strNum;

Variables – Data Types

- String
- Integer
- Double (floating point)
- Boolean
- Array
- Resource (file or database connection)
- Object (PHP class – new in PHP 5)

Variables - contd

- During execution, uses Java style garbage collection to automatically free up de-referenced, un-used variables.

- Always automatically cleans up and frees any variables and open file/database connections (resources) when a script ends or is terminated.

Literals

- Can have string literals and numeric literals, including hexadecimal literals

- $strval = “abcdefg”;

- $intnum = 123;
- $floatnum = 45.66;
- $float2 = 0.314E1;

- $booleanval = true;

- $hexnum = 0x1234;

- $val = null; // variable’s value is gone

Strings – Additional Details

- Strings come in two forms:
  - » Single quote: any embedded $xxx names are ignored
  - » Double quote: any embedded $xxx names will have the associated variable substituted in

  - » print ‘this will literally print $xyz’;
  - » print “this will substitute the value for variable $xyz”;
  - » print ‘mix quotes name=“abc” in a string’;
  - » print “use \\ escape characters \n”;

- Are a whole group of string operators and functions:
  - » Access a specific character in a string: $strVal[5]; // get 5th character
  - » Concatenate two strings: $st = ‘Add this ‘ . $val . ‘ together’;
  - » Trim spaces off a string: $x = trim($strval);
  - » Force to uppercase or lowercase: strtoupper($val); strtolower($val);
  - » Many more string functions – see PHP manual

Operators

- PHP operator syntax is similar to the operators used in C, Java, Javascript

- Arithmetic operators
  - » + - * / %
  - » Use parenthesis to specify precedence order: $x + ($y / 100)

- AutoIncrement/AutoDecrement operators
  - » ++ ---
  - » ++$x adds 1 to $x and then uses the result in the expression
  - » $x++ uses $x in the expression, then adds 1 to it

- Shift and Bit operators
  - » >> << & | ~
  - » $x >> 4 will shift $x contents right by 4 bits
  - » $x | $y will logically OR the contents of $x and $y

Operators - contd

- Assignment operators (syntax is similar to C, Java)
  - = += -= *= /= %=
  - $x = $y assigns the value of $y to $x
  - $x += $y is equivalent to $x = $x + $y
  - $x *= $y is equivalent to $x = $x * $y
  - $x %= $y is equivalent to $x = $x % $y (modulus operator)

Operators - contd

- Comparison operators
  - » == < > <= >= !=
  - » Caution: If ($x == $y) will test for equality
    $x = $y will set $x equal to $y

- Conditional operators (and/or/not condition x)
  - » && || !
  - » Use these to group comparison conditions together, e.g.
    if ($x == $y && $x < $y)

- Many other operators not listed here (see the manual)

Expressions

- An expression is a series of variables, operators, and method calls that evaluate to a single result value

- Use parenthesis to specify precedence order

```
$z = $a + ( ($x * 100) / ($y * 100) );
```

Flow of Control Constructs

- IF / THEN / ELSE
- WHILE and DO / WHILE
- FOR loops
- SWITCH / CASE statements

IF / THEN / ELSE

- ```
  if  (boolean_expression)
      next_statetment;
      else   someother statement;
  ```

  - » if the condition is true it executes the next statement, otherwise it execute the else clause.
  - » Curly braces are used to delimit blocks of code { }

- ```
  if  ($x <  $y)
      {  $hours = $hours + 1;
         $a       = $y;
      }
   else  $a  =  $x;
  ```

WHILE loops

- while (boolean_expression)
  { statements;
  }
  - » boolean_expression evaluated at the top of each loop
  - » Body is executed if boolean_expression evaluates to true
  - » Can use break; to force exit from the loop
  - » Can use continue; to skip processing rest of statements and go back to top of while loop

- while ($x < $y)
  { $x++;
  $a = $b + $x;
  if ($a < $p)
  continue; // skip back to top of while()
  if ($a > $y)
  break; // exit from while loop
  }

DO / WHILE loops

- ```
  do {
        statements;
  } while (boolean_expression)
  ```

  - » boolean_expression evaluated at the bottom of each loop
  - » Body is always executed at least once
  - » Can use break; to force exit from the loop
  - » Can use continue; to skip processing rest of statements and go back to top of loop

- ```
  do {  $x++;
        $a = $b + $x;
        if ($a < $p)
           continue;          // skip back to top of loop
        if ($a > $y)
           break;             // exit from loop
  } while ($x  <  $y)
  ```

FOR loops

- for (initialization ; continuation_expr ; increment_expr)
  {
  statements;
  }

  - » initialization is executed once at the beginning
  - » increment_expr is executed each time, at the bottom of the loop
  - » continuation_expr is executed at the top of every iteration
    the loop terminates when continuation_expr is false

- for ($i = 0; $i < 3; $i++)
  { $a = $i * $z;
  print (“<td> $a </td>”);
  }

SWITCH / CASE

```
switch (expression)
  { case x:
       statements for case x;
       break;
    case y:
       statements for case y;
       break;
    default:
       statements for default case;
   }
```

- Used to make a choice between multiple, alternative execution paths
- Choice (expression, x, y) can be either an integer or a string
- The break statement is option. If omitted, execution drops through to the next case statement.
- The default clause catches all other conditions

Switch/Case statements - contd

- switch ($int_var)
  { case 1:
  print $int_var; // then drops through
  case 2:
  $a = $x + $y
  break;
  }

- switch ($string_var)
  { case "ABC":
  print ("choice was ABC");
  break;
  case "XYZ":
  print ("choice was XYZ");
  break;
  default:
  print ("choice was none of the above");
  }

Comments

- Provides several different options
  - » Comment only the current line
    ```
    #   sign
    //
    ```
  - » Comment across one or more lines
    ```
    /* some comments
       more comments
    */
    ```
- Examples
    ```
    # comments here
    $x = $y;                    // comments here
    /* a comment
       across several lines
    */
    ```

PHP Language – Part II

- Functions
- Classes
- Arrays
- Basics of setting up a <?php xxx> script

Functions

- Calling functions is part of the heart and soul of PHP

- Are two types of functions:
  - » Built-in functions – pre-compiled functions compiled into PHP.
    For C-based PHP, these are C sub-routines that have been compiled and linked into PHP, and are part of the *EXT Modules* of PHP
    - – Static functions – compiled directly into PHP
    - – Dynamic functions – DLLs that are loaded as needed (third party add-ons)

    For Java-based PHP, these are implemented as jar files.

  - » User-defined functions – PHP routines defined by the customer
    - – User-defined functions are always interpreted (can be cached)
    - – If maximum speed is required, can re-write into a PHP "extension" instead

- Both types of functions use the same calling conventions:
  - » Uses C/Java style calling syntax
    $x = function_abc ($a, $b);
  - » Can return simple data types (integer, string, float) or complex data types (arrays, resources, class objects)

Typical Built-In Functions

- Sample list of the most popular PHP functions:
  - » HTML processing: header/strip_tags/htmlentities/ …
  - » File access: fopen/fwrite/fread/fclose/mkdir/opendir/copy/ …
  - » Database access: PDO objects
  - » XML processing: DOM parse/read/update/write/ …
  - » URL processing: parse_url/base64_decode/base64_encode/ …
  - » Session processing: setcookie/session_start/session_register/ …
  - » Date/Time manipulation: getdate/checkdate/ …
  - » Array processing: read/write/create/sort/search/re-order array data
  - » Math: abs/acos/atan/exp/max/min/rand/sqrt/ …
  - » Logging: write to log, trigger error handlers, …
  - » Echo/print/printf: write out text or variables to a file or HTTP connection
- Are over 200 built-in functions

User Defined Functions

- Declaring a user function
  - » Use function keyword, followed by name of function, followed by parms.
  - » Body of the function is then encapsulated in { } block.
  - » Function names must start with alphabetic char or underscore.

```
function  std_page_header ($title,$color)
     { print "<html><head><title>$title</title></head>";
       print '<body bgcolor=" '#.$color.' ">';
     }                                  //     ^        ^  concatenate strings
```

- Parameters can be <u>assigned default values</u>

```
function  std_page_header ( $title='XYZ Corp', $color='cc00cc')
   { same as above }
```

- Functions can return simple or complex values

User Functions – Returning Simple Values

- Return simple value:

```
function   compute_sales_tax ($meal)
    {  $tax_amount  =  $meal  *  (8.5 / 100);
       return  $tax_amount;          // returns a simple number
    }
```

- Usage:

```
$s_tax  =  compute_sales_tax ($meal_amount);
```

User Functions – Returning Arrays

- Return complex value (array, …)

```
function   compute_tax_and_tip ($meal)
    {  $tax_amount  =  $meal  *  (8.5 / 100);
       $tip_amount   =  $meal  *  (15 / 100);
       return  array ($tax_amount, $tip_amount);       // returns an array
    }
```

- Usage:

```
$totals = compute_tax_and_tip ($meal_amount);
print  “meal=$meal_amount    tax=$totals[0]    tip=$totals[1]”;
```

Incorporating User Defined Functions

- Can include inline or via include statement

- Inline

```
<?php
        function  abc ($parm1)
           {  $val2 = parm1 / 50;
              return $val2 * 1000;
           }

         $ret_val = abc (123);
         print $ret_val;
 ?>
```

- Include from a separate file (eg. a library of common functions)

```
<?php
        include ("abc.inc");

        $ret_val = abc (123);
        print $ret_val;
?>
```

Classes / Objects

- New in PHP5
  - » Allows programmer to use object oriented approach similar to Java
  - » CICS PHP uses classes extensively

- Declaring a class:

```
class  classnameXYZ
    {   var  $property1 = value;

        function  methodname1 (args)
            { statements;
              return $resultVal;
            }
       . . .
     }
```

- Invoking a class:

```
$aclass  = new classnameXYZ();
$result    = $aclass->methoname1 ($somearg);
$propval = $aclass->property1;                    // note no $ on property1
```

PDO Objects – DBMS Access

- <?php
  try {
  $db = new PDO (“ibm:DRIVER={IBM DB2 ODBC DRIVER};
  DATABASE=accounts; HOSTNAME=1.2.3,4;PORT=56789;
  PROTOCOL=TCPIP;", "username", "password");

  $count = $db->exec ("INSERT INTO animals(animal_type, animal_name)
  VALUES (‘dogs', ‘penelope')");
  print ”num rows affected = $count \n”; // tell number of rows affected

  $sqlstmt = "SELECT * FROM animals";
  print “Query results:\n”;
  foreach ($db->query($sqlstmt) as $row)
  {
  print $row['animal_type'] . ' - ‘ . $row['animal_name'] . '
';
  }

  $db = null; // closes the database connection
  }
  catch (PDOException $e) { echo $e->getMessage(); }
  ?>

PHP Arrays

- Array = primary vehicle for passing data around:
  - » HTTP Server variables (query string, remote host, browser name)
  - » HTTP POSTed Forms variables
  - » Database Row Results (each row = an array);

- Array data can be accessed via either name or number
  - » all arrays are associative arrays using hashed lookup, e.g.

```
$browserName = $_SERVER ['HTTP_USER_AGENT'];
$firstValue = $someArray [0];
```

- To "walk" through all elements in an array, use the foreach() function:

```
$meal = array ('breakfast' => 'cereal', lunch => 'sandwich');
foreach ($meal as $key => $value)
    print "meal is $key and we serve $value\n";
```

PHP Arrays

- Arrays can be single dimensional or multiple dimensional:

- Single Dimension:

```
$simpArray = array (‘name’ => ‘Vic’,
                    ‘age’    => 40);
```

- Multiple dimension:

```
$multDim = array (‘name’ => ‘Vic’,
                  ‘residences’ => array (‘home’  => ‘Sebastopol’,
                                         ‘dacha’ => ‘Green Bay’),
                  ‘age’   => 40);

$vacationspot =  $multDim [‘residences’] [‘dacha’];
```

- Anonymous array (no associative names):

```
$anonArray = (‘ABC’, ‘DEF’, ‘GHK’);
$element2   = $anonArray[1];
```

First element of an array is always [0]

Arrays used for Passing Web Data In

Incoming HTTP Requests

Method: GET

xyz.php?name=Jim&age=40

Associated PHP Array

```
$cust_name = $_GET ['name'];
$cust_age  = $_GET ['age'];
```

Method: POST

abc.php
action=sell
num=100
price=20.00

```
$action_id = $_POST ['action'];
$num_shares = $_POST ['num'];
$share_price  = $_POST ['price'];
```

Note: CICS PHP supports all of this stuff !

Using key PHP Arrays in Web Apps

- Server information is kept in $_SERVER array, e.g.
  $_SERVER ['REQUEST_METHOD'] // GET or POST
  $_SERVER ['REMOTE_HOST']
  $_SERVER ['SCRIPT_NAME']
  many more – see manual

- Input parameters are extracted by name from appropriate array ($_GET or $_POST) based upon the type of request:
  - » GET requests: $abc = $_GET ['parm_name1'];
  - » POST requests: $xyz = $_POST ['parm_name'];

- Session/Cookie related information is extracted from $_COOKIE array

Building a <?php > script

- A PHP-enabled Web Server detects PHP scripts based on the .php suffix, and passes the request to the PHP processor for execution

- PHP commands are encoded within <?php (start) and ?> (end) markers
  - » Any valid PHP function or PHP class can be invoked from within a PHP script

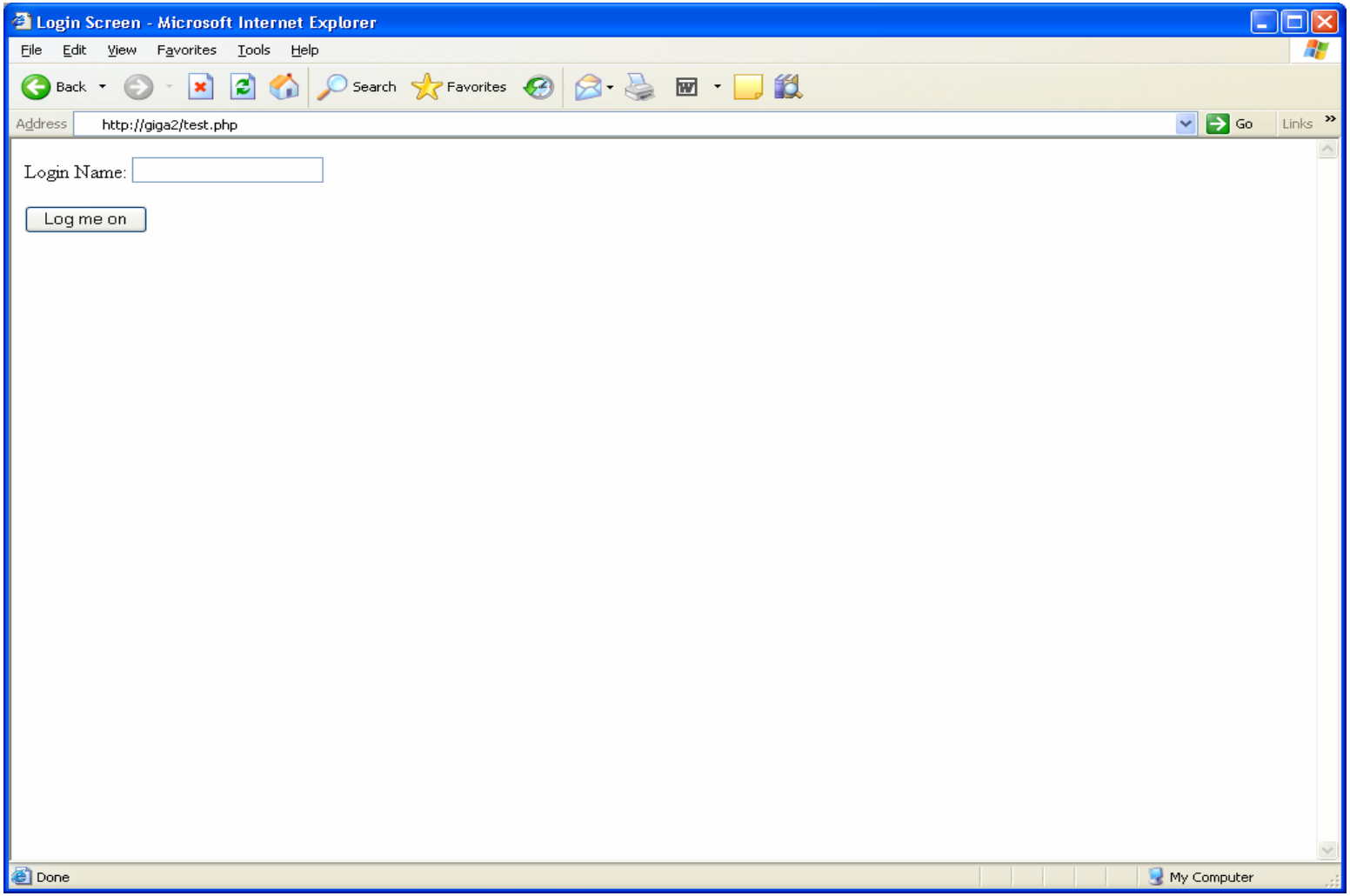
- PHP has similarities with Microsoft's ASP and Java's JSP technology:
  - » For Web apps, PHP commands can be embedded within HTML and XML pages.
  - » Anything not within the PHP command markers is sent back to the Web client as is (e.g. inline HTML or XML)

PHP – Generate Simple HTML Form

```
<html>
<head>
<title>Login Screen</title>
</head>

<body>
<?php
   if (array_key_exists('user_name',$_POST))                  // see if input parm is in array
      print "Hello, " . $_POST['user_name'];                  // yes, echo the user's name
      else                                                    // no, then request user's name
                                                              // and send it back to this same page
          print  "<form method='post'  action='$_SERVER['SCRIPT_NAME']'>
    Login Name: <input type='text'  name='user_name'>
    <br/>
    <br/>
    <input  type='submit'  value='Log me on'>
    </form>"                                                  // end of multi-line literal
?>
</body>
</html>
```

Sample Browser screen shot - Simple Form



Sites running PHP usually have their initial default HTML page redirect to a .php page, which will then hand off to subsequent .php pages

Where to get more information

- PHP Code download
  http://www.php.net/downloads.php
- PECL (PHP Extensions)
  http://www.pecl.php.net/
- PEAR (Libraries of PHP Routines)
  http://www.pear.php.net/
- Zend Engine
  http://www.zend.com/
- Learning PHP 5, Sklar, Oreilly & Associates
- Programming PHP, Lerdorf, Tatroe, O'Reilly & Associates, ISBN1-56592-610-2
- PHP Cookbook, Sklar, Trachtenberg, Oreilly & Associates
- Advanced PHP Programming, Schlossnagle, Sams Publishing

EOJ