Lënda: Programimi në Internet

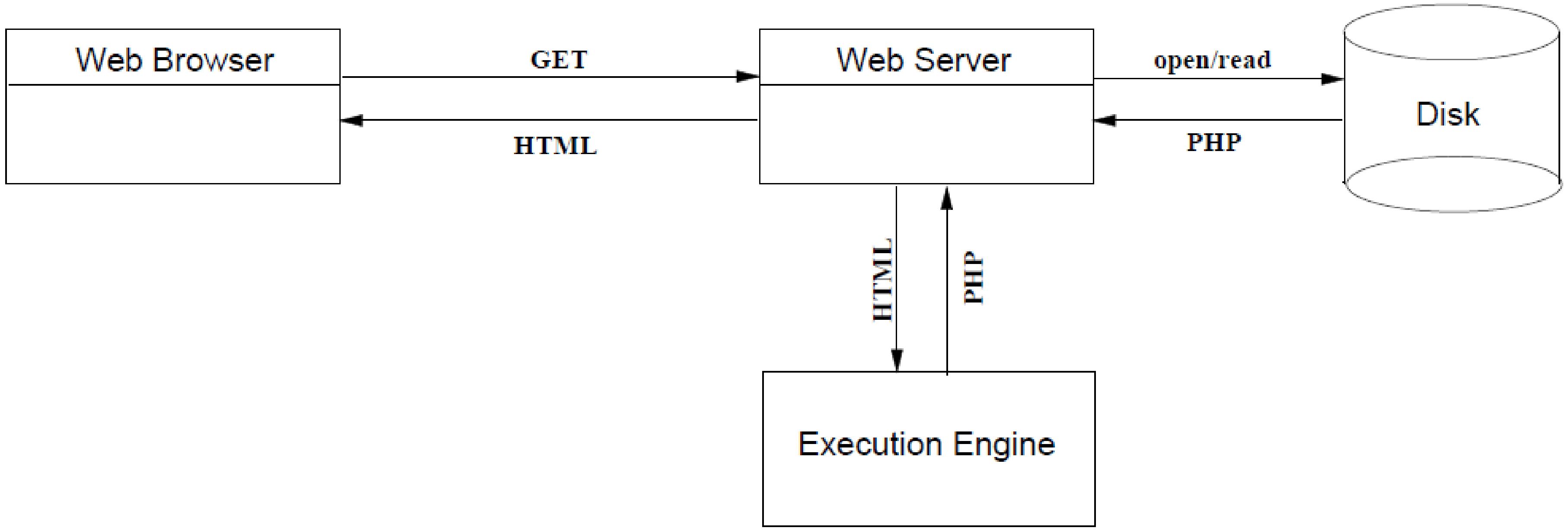
Universiteti i Prishtinës

PHP, Part I

Dr. Ing. Lule Ahmedi

Slides mostly adopted from Tufts University, Prof. Ming Chow

Architecture of the c/s PHP



Server-Side Programming

- Program that resides on the server
- Can:
  - Store state
  - Invoke system functions
  - Communicate with databases
  - Generate content
  - Perform computations
  - Enforce security
- Cannot:
  - Access information on the client (the browser) other than what is sent to server

Advantages of Server-Side Programming

- Portability: Everything we do will work on any browser
- Adaptability: We can even adapt output to whatever browser the user is using!
- Flexibility: We can change servers without modifying any scripts!

Disadvantages of Server-Side Programming

- Inefficiency: It takes time to execute
- Bandwidth: it takes time to send data from a program to a client
- Scalability: if you get hit 1000 times a second, complex programs are unworkable unless you have a really big server or server farm (see Google)
- Concurrency: it's difficult to write server-side scripts so that several copies of a script can run simultaneously
- Security: Very easy to write insecure programs
  - Easy to give access to data & files that users are now allowed to view
  - Many web developers are (still) clueless about basic web security

What is PHP?

Dynamically-typed server-side scripting language

- Used to make Web pages dynamic
  - Process form information
  - Provide various content (including PDF, GIF images)
  - Interface with e-mail, databases, XML, etc.
  - Connect to other network services (like LDAP)
- Command-line scripting: can run scripts from the command line, much like Perl, or the Unix shell
- Client-side GUI applications, see PHP-GTK for details

A Short History of PHP

- Conceived in 1994 by Rasmus Lerdorf, a real zenit in 2001
- Originally stood for "Personal Home Page”, but changed to PHP Hypertext Preprocessor
- Developed to:
  - Replace Perl
  - Display his résumé
  - Track the number of visitors to his page (web analytics)
- Written in C

Why PHP?

- Free
- Simple; familiar syntax
- Lots of built-in functions
- Supported on most web hosting providers and servers
- Can access data and information on server
- Fewer security restrictions compared to other languages

Sample Application

```
<form action="processororder.php" method=post>
<table border=0>
<tr bgcolor=#cccccc>
   <td width=150>Item</td>
   <td width=15>Quantity</td></tr>
<tr>
   <td>Tires</td>
   <td align="center"><input type="text"
   name="tireqty" size="3" maxlength="3"></td></tr>
<tr>
   <td>Oil</td>
   <td align="center"><input type="text"
   name="oilqty" size="3" maxlength="3"></td></tr>
<tr>
   <td>Spark Plugs</td>
   <td align="center"><input type="text"
   name="sparkqty" size="3" maxlength="3"></td></tr>
<tr>
   <td colspan="2" align="center"><input
        type="submit" value="Submit Order"></td></tr>
</table></form>
```

Sample Application (cont.)

- The name of the PHP script that will process the order, *not* the URL where the user data will be sent

```
action="processorder.php"
```

- Keep in mind the names of the form fields for later call within a PHP code

Processing the Form: Embedding PHP in HTML

The `processorder.php` file

```
<html>
<head>
  <title>Bob’s Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob’s Auto Parts</h1>
<h2>Order Results</h2>
<?php
  echo ’<p>Order processed.<p/>’;
?>
</body>
</html>
```

Preliminaries

- File suffix: .php. Server runs PHP on the file.
- PHP files can contain HTML and special PHP directives. PHP directives are parsed by the server and never seen by the client.
- Interpreted language
- Relaxed syntax and rules. Variables do not need to be declared
- Built-in regular expressions (very powerful)
- A simple embedded PHP program:

```
<?php echo ("hello there"); ?>
```

where `<?php` starts PHP extension,
whereas `?>` ends PHP extension

Preliminaries (cont.)

- `echo ("hello there");` - put this into current document
- `phpinfo();` - displays system information
  - Example: `<?php phpinfo(); ?>`
  - Useful to test PHP installation
  - Lists the modules that are enabled (e.g., MySQL, GD2, XML)
  - Lists the built-in / system variables (e.g., `$_SERVER["HTTP_USER_AGENT"])`
- Comments:
  - `#` - single line
  - `//` - single line
  - `/*`
    `*/` - multi lines

Similarities and Differences Between PHP and Other Languages

PHP	JavaScript	C/C++
dynamically typed	dynamically typed	statically typed
$x	var x	int x
$this	this	this
global $x	x	main::x
$x = array('foo', 'bar');	x = new Array('foo', 'bar');	no analogue
$x[0]	x[0]	no analogue
$x = array('foo'=>'bar');	x = new Object('foo':'bar');	no analogue
$x['foo']	x['foo']	no analogue
"SOME STRING" evaluates variables	"SOME STRING" and 'SOME STRING' same	"SOME STRING" is string, '.' is character
function foo() {...}	function foo() {...}	int foo() {...}
class Foo	no "class" keyword	class Foo
class Foo extends Bar	no inheritance	class Foo: public Bar

Similarities and Differences Between PHP and Other Languages (cont.)

- `if`, `else`, `while`, `do`, `for` work as usual
- Usual gang of arithmetic and logical operators (`+`, `-`, … `==`, `>=`, `>`, `++`, `--`, `&&`, `||`, …)
- PHP uses `elseif` keyword in `if...then` statements
- Use a semicolon at the end of each statement
- Whitespaces ignored - use them for readability only

Accessing Form Variables

Basically, access a form field using a PHP variable whose name relates to the name of the form field

- Variables in PHP start with `$`

Method 1: The same name preceded with `$`, like `$tireqty`

- The form variables are all passed into your script (like arguments are to functions)
- Convenient, but error-prone: could be easily mixed-up with user defined global variables
  - To avoid it, initialize your own variables in time

Accessing Form Variables (cont.)

Method 2: The name of a variable as a member identifier of array

- Form variables are stored in one of the arrays _GET, _POST, or _REQUEST, depending on the transfer method
  - `$_POST['tireqty']` , or
  - `$HTTP_POST_VARS['tireqty']`

Accessing Form Variables: The Running Example

```
<?php
  //create short variable names
  $tireqty = $HTTP_POST_VARS['tireqty'];
  $oilqty = $HTTP_POST_VARS['oilqty'];
  $sparkqty = $HTTP_POST_VARS['sparkqty'];

echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br/>';
echo $oilqty.' bottles of oil<br/>';
echo $sparkqty.' spark plugs<br/>';
?>
```

- '.' is a string concatenation operator

Variable Types

PHP is a very weakly typed language

- No need to declare a variable before using it
- The type of a variable is determined by the value assigned to it (on-the-fly change of type)

  `$totalqty = 0;` => of type integer

  `$totalamount = 0.0;` => of type double

  `$totalqty = 'Hi';` => turned into a string

Built-in data types:

- Integer, Double, String, Boolean, Array, Object, NULL, etc.

More on Variables

- Variable names are cAsE sEnSiTiVe
- Type casting
  - `$totalqty = 0;`

    `$totalamount = (double)$totalqty;`
  - The type of `$totalqty` remains integer
- Variable variables
  - Allow to change the name of variables dynamically
  - `$varname = 'totalqty';`

  then `$$varname = 5` same as if `$totalqty = 5`

Constants - Examples

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

Printing and Interpreting Strings

- `print("some text\n");` or
  `echo("some text\n");`
- Strings inside " " are interpreted or evaluated --replaced with its meaning
- Strings inside ' ' are literal
- Example:

```
<?php
    $foo = 3;
    echo "foo has the value of $foo";
        // Result: foo has the value of 3
    echo 'foo has the value of $foo';
        // Result: foo has the value of $foo
?>
```

Printing and Interpreting Strings

- . is used for string concatenation
- + in PHP always mean numeric addition
- If something is a string, and you need a number, it is automatically converted to a number (an integer) by looking at the first few characters. Examples:
  - The value of "hi" is 0
  - The value of "24.5e7hohohoho" is 245000000
- Zero-based indexing using bracket notation. Example:
  - `$someString = 'incoherent`;`
  - `$c = $someString[3]; // $c = 'o'`

Arrays

- Array constructor `array()`
- Use []'s
- Indices and elements can be anything
- The idea of associate arrays is supported:
  - Example: `$names['Clinton'] = 'Bill';`

```
<?php
      $a = array();
      $a[1] = "hi";
      $a['ho'] = 1;
      echo '$a[1] is ' . $a[1] . '<br>';
      echo '$a["ho"] is ' . $a["ho"] . '<br>';
      echo '$a[$a["ho"]] is ' . $a[$a["ho"]] . '<br>';
?>
```

- This produces:

```
      $a[1] is hi
      $a["ho"] is 1
      $a[$a["ho"]] is hi
```

Arrays (cont.)

- You can construct whole arrays with one subroutine call. Example:

```
<?php
    $b = array(
            1 => 2,
            3 => "hi",
            'ho' => 'hoho′ );
            echo '$b[1] is ' . $b[1] . '<br>';
            echo '$b[3] is ' . $b[3] . '<br>';
            echo '$b["ho"] is ' . $b["ho"] . '<br>';
?>
```

- This produces:

```
    $b[1] is 2
    $b[3] is hi
    $b["ho"] is hoho
```

Writing Functions in PHP

- Format:

```
function name (parameters) {
     statements;
     ...
     ...
     ...
}
```

- No parameter or return types

  (e.g., in C/C++ `int fibonacci (int n)`)
- All variables declared inside functions are local

Writing Functions in PHP - Example

```
function areaCircle ($radius)
{
   if (isset($radius)) {
        return pi() * pow($radius, 2);
        // Notice the use of built-in functions pi()
        // and pow(base, power)
   }
   return 0;
}

echo "Area of circle with radius 5 = " . areaCircle(5) . "<br>\n";
echo "Area of circle with radius 10 = " . areaCircle(10) . "<br>\n";
echo "Area of circle with radius NULL = " . areaCircle() . "<br>\n";
   // Is this legal?
```

PHP Built-In Functions: Strings

- `strlen($str)` - String length
- `strcmp($str2, $str2)` - String compare; Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal
- `strpos($str, $char)` - Return position of character; 0 to strlen(#str) - 1, or FALSE if not found
- `substr($str, $pos1, $pos2)` - Substring given positions
- `strtoupper($str)`, `strtolower($str)` - Uppercase or lowercase entire string
- `explode($delimeter, $str)` - Split a string by string; returns an array
- `implode($glue, $pieces)` - Join array elements with a string; returns a string
- Complete list of string functions

PHP Built-In Functions: Arrays

- `count`
- `print_r`
- `array_pop()`
- `array_push()`
- `array_shift()`
- `array_unshift()`
- `array_reverse()`
- `in_array()`
- `rsort()`
- `shuffle()`
- `sort()`
- `array_merge()`
- `array_slice()`
- `array_keys()`

Complete list of string functions

PHP Built-In Functions: Arithmetics

- `intval()` Returns < - Converts a string to an integer
- `(int)$someDouble` - Type casting
- The usual gang: `abs, ceil, floor, max, min, rand, round, srand`
- `pi`() or `M_PI – Pi`
- `pow($base, $power)`
- Complete list of math functions