COMP284 Scripting Languages
Lecture 9: PHP (Part 1)
Handouts

Department of Computer
School of Electrical Engineering, Electronics, an
University of Liverpo

Contents

Common Gateway Interface — CGI

The Common Gateway Interface (CGI) is a standard method for web servers to use external applications, a CGI program, to dynamically generate web pages

1. A web client generates a client request, for example, from a HTML form, and sends it to a web server
2. The w
   conv
3. The C
   the program's response back to the client

Disadvantages of CGI/Perl

- A distinction is made between static web pages and dynamic web pages created by an external program
- Using Perl scripting it is difficult to add 'a little bit' of dynamic content to a web page
  – can be a
- Use of a
  - star
  - exchanging data between web server and external p

  ⇝ resource-intensive

If our main interest is the creation of dynamic web pages, then the scripting language we use

- should integrate well with HTML
- should not require a web server to execute an external program

PHP

- PHP is (now) a recursive acronym for PHP: Hypertext Preprocessor
- Development started in 1994 by Rasmus Lerdorf
- Originally designed as a tool for tracking visitors at Lerdorf's website
- Devel
  server
- Inheri
- Easy-to-use interface to databases
- Free, open-source
- Probably the most widely used server-side web pro
- Negatives: Inconsistent, muddled API; no scalar objects

The departmental web server uses PHP 5.6.25 (released August 2014)
PHP 7 was released in December 2015 (PHP 6 was never released)

PHP processing

- Server plug-ins exist for various web servers
  ⇝ avoids the need to execute an external program
- PHP code is embedded into HTML pages using tags
  ⇝ static web pages can easily be turned into dynamic ones

PHP satis

Processi

1. The web server receives a client request
2. The web server recognizes that the client request i
   a HTML page containing PHP code
3. The server executes the PHP code, substitutes output
   into the HTML page, the resulting page is then send to the client

As in the case of Perl, the client never sees the PHP code,
only the HTML web page that is produced

PHP: Applications

- Applications written using PHP
  - activeCollab – Project Collaboration Software
    `http://www.activecollab.com/`
  - Drupal – Content Management System (CMS)
    `http://drupal.org/home`
  - Ma – eCo
    `htt`
  - Me
    `http://www.mediawiki.org/wiki/Me`
  - Moodle – Virtual Learning Environment (VLE)
    `http://moodle.org/`
  - Sugar – Customer Relationship Management (CRM) platform
    `http://www.sugarcrm.com/crm/`
  - WordPress – Blogging tool and CMS
    `http://wordpress.org/`

PHP: Websites

- Websites using PHP:
  - Delicious – social bookmarking
    `http://delicious.com/`
  - Digg – social news website
    `http://digg.com`
  - Fac – so
    `htt`
  - Flic – photo shari
    `http://www.flickr.com`
  - Friendster – social gaming
    `http://www.friendster.com`
  - SourceForge – web-based source code repository
    `http://sourceforge.net/`
  - Wikipedia – collaboratively built encyclopedia
    `http://www.wikipedia.org`

Recommended texts

- R. Nixon:
  Learning PHP, MySQL, and JavaScript.
  O'Reilly, 2009.

  Har
  (or lat

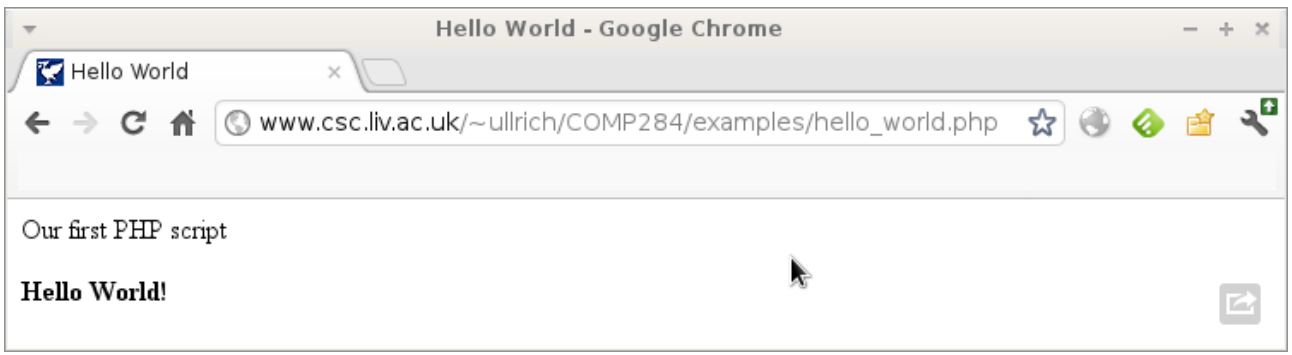
- M. Achour, F. Betz, A. Dovgal, N. Lopes,
  H. Magnusson, G. Richter, D. Seguy, J.
  PHP Manual.
  PHP Documentation Group, 2018.

  `http://www.php.net/manual/en/index.php`

PHP: Hello World!

```
1 <html>
2 <head><title>Hello World</title></head>
3 <body>
4 <p>Our first PHP script</p>
5 <?php
6   print ("<p><b>Hello␣World!</b></p>\n");
7 ?>
8 </body>
```


- PHP co
- File must be stored in a directory accessible by the web
  example $HOME/public_html, and be reada
- File name must have the extension .ph p



PHP: Hello World!

Since version 4.3.0, PHP also has a command line interface

```
1 #!/usr/bin/php
2 <?ph
3   /* Author: Ulrich Hustadt
4        A "Hello World" PHP script. */
5   prin
6   // A sing
7 ?>
```


- PHP code still needs to be enclosed between <?php and ?>
- Code must be stored in an executable file
- File name does not need to have any particular forma

↝ PHP can be used as scripting language outside a web programming context

Output:

```
Hello World!
```

PHP: Hello World!

```
<html>
<head><title>Hello World</title></head>
<body><p>Our first PHP script</p>
<?php
  print("<p><b>Hello World!</b></p>\n");
?>
</body></
```


- Can als

  ```
  php filename
  ```

- File does not need to exectuable, only readable for th

Output:

```
<html>
<head><title>Hello World</title></head>
<body><p>Our first PHP script</p>
<p><b>Hello World!</b></p>
</body></html>
```

PHP scripts

- PHP scripts are typically embedded into HTML documents and are enclosed between <?php and ?> tags
  - A PHP script consists of one or more statements and comments
    ⇝ there is no need for a main function (or classes)
  - Stat
  - Whi
    (Th
  - One // #
  - Multi-line comments are enclosed in /* a ?>

Types

PHP has eight primitive types

- Four scalar types:
  - bool – booleans
  - int – integers
  - flo
  - str
- Two compound types:
  - array – arrays
  - object – objects

NULL

- Integers, floating-point numbers, and strings do n from the corresponding Perl scalars, including th single-quoted versus double-quoted strings
- In contrast to Perl, PHP does distinguish between different types including between the four scalar types

Variables

- All PHP variable names start with $ followed by a PHP identifier
- A PHP identifier consists of letters, digits, and underscores, but cannot start with a digit
  PHP identifiers are case sensitive
- In PHP
- A varia
  although initialisation is a good idea
- Uninitialized variables have a default value of their t
  the context in which they are used

Type	Default	Type	Default
bool	FALSE	string	empty string
int/float	0	array	empty array

If there is no context, then the default value is `NULL`

Assignments

- Just like Java and Perl, PHP uses the equality sign = for assignments

```
$student_id = 200846369;
```


- As in Perl, this is an assignment expression
- The value of an assignment expression is the value assigned

```
$b = ($a = 0) + 1;
// $a is 0
// $b is 1
```

Binary assignments

PHP also supports the standard binary assignment operators:

Binary assignment	Equivalent assignment
`$a += $b`	`$a = $a + $b`
`$a -= $b`	`$a = $a - $b`
`$a **= $b`	`$a = $a ** $`
`$a .= $b`	`$a = $a . $b`

Example:

```
// Convert Fahrenheit to Celsius:
// Subtract 32, then multiply by 5, then divide by 9
$temperature = 105;              // temperature in Fahrenheit
$temperature -= 32;
$temperature *= 5/9;             // converted to Celsius
```

Constants

- `bool define(string, expr [, case_insensitive])`
  - defines a constant that is globally accessible within a script
  - *string* should be a string consisting of a PHP identifier (preferably all upper-case)
    The PHP identifier is the name of the constant
  - *exp*
  - *cas*
    whe
  - returns TRUE on success or FALSE on failure

```
define("PI",3.14159);
define("SPEED_OF_LIGHT",299792458,true);
```

Constants

- To use a constant we simply use its name

```
define("PI",3.14159);
define("SPEED_OF_LIGHT",299792458,true);
$circumfence = PI * $diameter;
$distance    = speed_of_light * $time;
```

- Cavea


```
print "1 - Value of PI: PI\n";
print "2 - Value of PI: ".PI."\n";
```

```
1 - Value of PI: PI
2 - Value of PI: 3.14159
```

Values, Variables and Types

PHP provides several functions that explore the type of an expression:

string gettype(*expr*)	returns the type of *expr* as string
bool is_*type*(*expr*)	checks whether *expr* is of type *type*
void var_dump(*expr*)	displays structured information about *expr*


```
<?php print 'T

      print "Type of \"23\": ".gettype("23")."\n";

      if (is_int(23)) { echo "23 is an integer\n"; }
        else { echo "23 is not an integer\n"; }
?>
```

```
Type of 23:    integer
Type of 23.0: double
Type of "23": string
23 is an integer
```

Type juggling and Type casting

- PHP automatically converts a value to the appropriate type as required by the operation applied to the value (type juggling)

```
2 . "␣worlds"      ⇝  "2␣worlds"
"2" * 3            ⇝  6
"1.2
"hel
"10h
```


- PHP also supports explicit type casting via

```
(int) "12"                ⇝  12
(int) "1.23e2"            ⇝  1
(int) ("1.23e2" + 0)      ⇝  123
(int) "10hello5"          ⇝  10
(int) 10.5                ⇝  10
(array) "foo"             ⇝  array(0 => "foo")
```

```
(                         FALSE
(                         TRUE
(float) "1.23e2"          ⇝  123
```

Comparison operators

Type juggling also plays a role in the way PHP comparison operators work:

expr1 == *expr2*	Equal	TRUE iff *expr1* is equal to *expr2* after type juggling
expr1 != *expr2*	Not equal	TRUE iff *expr1* is not equal to *expr2* after type juggling
expr1 <> *expr2*	Not equal	TRUE iff *expr1* is not equal to *expr2*
expr1 *expr1*		*pr2*,

Note: For ==, !=, and <>, numerical strings are converted and compared numerically

```
"123" == 123              ⇝  TRUE
"123" != 123              ⇝  FALSE
"1.23e2" == 123           ⇝  TRUE
"1.23e2" == "12.3e1"      ⇝  TRUE
5 == TRUE                 ⇝  TRUE
```

```
"123" === 123             ⇝  FALSE
"123" !== 123             ⇝  TRUE
1.23e2 === 123            ⇝  FALSE
"1.23e2" === "12.3e1"     ⇝  FALSE
5 === TRUE                ⇝  FALSE
```

Comparison operators

Type juggling also plays a role in the way PHP comparison operators work:

expr1 < *expr2*	Less than	TRUE iff *expr1* is strictly less than *expr2* after type juggling
expr1 > *expr2*	Greater than	TRUE iff *expr1* is strictly greater than *expr2* after type juggling
expr1 <= *expr2*	Less than	TRUE iff *expr1* is less than or equal to *expr2*
expr1		*expr2*


```
'35.5' > 35              ⇝  TRUE
'ABD' > 'ABC'            ⇝  TRUE
'1.23e2' > '12.3e1'      ⇝  FALSE
"F1" < "G0"              ⇝  TRUE
TRUE > FALSE             ⇝  TRUE
5 > TRUE                 ⇝  FALSE
```

```
'35                      TRUE
'AB                      TRUE
'1.                      TRUE
"F1                      TRUE
TRUE >= FALSE            ⇝  TRUE
5 >= TRUE                ⇝  TRUE
```

Revision

Read

- Chapter 3: Introduction to PHP

of

R. Nixo

Learni

O'Reilly, 2009.

Also read

- `http://uk.php.net/manual/en/language.types.intro.php`
- `http://uk.php.net/manual/en/language.types.type-juggling.php`
- `http://uk.php.net/manual/en/language.operators.comparison.php`
- `http://uk.php.net/manual/en/types.comparisons.php`