COMP519 Web Programming
Lecture 19: PHP (Part 1)
Handouts (8 on 1)

Ullrich Hustadt
Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

1 PHP
   - Motivation
2 Overview
   - Features
   - Applications
3 Types and Variables
   - Types
     - Integers and Floating-point numbers
     - Booleans
     - Strings
   - Variables

PHP Motivation

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 web server selects a CGI program to handle the request, converts the client request to a CGI request, executes the program
3 The CGI program then processes the CGI request and the server passes the program’s response back to the client

Disadvantages of CGI

- A distinction is made between static web pages and dynamic web pages created by external CGI programs
- Using CGI programs it is difficult to add ‘a little bit’ of dynamic content to a web page
  ~ can be alleviated to some extent by ‘packing’ big chunks of HTML markup into a few strings
- Use of an external program requires
  ~ starting a separate process every time an external program is requested
  ~ exchanging data between web server and external program
    ~ resource-intensive
If our main interest is the creation of dynamic web pages, then the programming language we use
- should integrate well with HTML
- should not require a web server to execute an external program

Overview Features

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
- Developed into full-featured, scripting language for server-side web programming
- Shares a lot of the syntax and features with other languages
- Easy-to-use interface to databases
- Free, open-source
- Probably the most widely used server-side web programming language
- Negatives: Inconsistent, muddled API; no scalar objects; compatibility problems between PHP 5.x and PHP 7.x (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 satisfies the criteria we had for a good web scripting language

Processing proceeds as follows:
1 The web server receives a client request
2 The web server recognizes that the client request is for a HTML document containing PHP code
3 The server executes the PHP code, substitutes output into the HTML document, the resulting page is then send to the client

As in the case of CGI programs, the client never sees the PHP code, only the HTML document 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
  - Magento ~ eCommerce platform
    http://www.magentocommerce.com/
  - MediaWiki ~ Wiki software
    http://www.mediawiki.org/wiki/MediaWiki
  - 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
  - Facebook ~ social networking
    http://www.facebook.com
  - Flickr ~ photo sharing
    http://www.flickr.com
  - Frienster ~ social gaming
    http://www.frienster.com
  - SourceForge ~ web-based source code repository
    http://sourceforge.net/
  - Wikipedia ~ collaboratively built encyclopedia
    http://www.wikipedia.org
Since version 4.3.0, PHP also has a command line interface

```
#!/usr/bin/php
<?php
// Author: Ulrich Hustadt
A "Hello World" PHP script.
print ("Hello World\n");
// A single-line comment
?>
Hello World!
```

- 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 format
- PHP can be used to write CGI programs
- PHP can be used as a scripting language outside a web programming context

The constants `NAN` and `INF` are used as return values for some applications of mathematical functions that do not return a number

- `log(0)` returns `-INF` (negative 'infinity')
- `sqrt(-1)` returns `NAN` ('not a number')

In PHP 5
- `1/0` returns `FALSE` and produces a PHP warning
- `0/0` returns `FALSE` and produces a PHP warning and execution of the script continues!

In PHP 7
- `1/0` returns `INF` and produces a PHP warning
- `0/0` returns `NAN` and produces a PHP warning and execution of the script continues!

### Integers and Floating-point numbers: NAN and INF

#### PHP has eight datatypes

- **Four primitive types:**
  - `bool` – booleans
  - `int` – integers
  - `float` – floating-point numbers
  - `string` – strings
- **Two compound types:**
  - `array` – arrays
  - `object` – objects
- **Two special types:**
  - `resource`
  - `NULL`

Integers, floating-point numbers, and booleans do not differ significantly from the corresponding JavaScript types

Strings differ from those in JavaScript

### Integers and Floating-point numbers

- **PHP distinguishes between**
  - `integer numbers` 0 2012 -40 1263978
  - `floating-point numbers` 1.25 256.0 -12e19 2.4e-10
- **PHP supports a wide range of pre-defined mathematical functions**
  - `abs(number)` absolute value
  - `ceil(number)` round fractions up
  - `floor(number)` round fractions down
  - `round(number [, prec, mode])` round fractions
  - `log(number [, base])` logarithm
  - `rand(min, max)` generate an integer random number
  - `sqrt(number)` square root
- **PHP provides pre-defined number constants including**
  - `M_PI` 3.14159265358979323846
  - `NAN` 'not a number'
  - `INF` 'infinity'

#### PHP provides the following `math` functions

- `max()`
- `min()`
- `round()`
- `ceil()`
- `floor()`
- `sqrt()`
- `abs()`
- `log()`

#### Boolean operators

- `&&` (conjunction)
- `||` (disjunction)
- `!` (negation)

**PHP offers the same short-circuit boolean operators as Java and JavaScript:**

- `&&` and `||` can be used instead of `&&` and `||`, respectively
- However, `not` is not a PHP operator
- The truth tables for these operators are the same as for JavaScript
- Remember that `&&` and `||` are not commutative, that is,
  - `(A && B) is not the same as (B && A)`
  - `(A || B) is not the same as (B || A)`
**Type conversion to boolean**

When converting to boolean, the following values are considered `FALSE`:
- the boolean
- the integer 0 (zero)
- the float 0.0 (zero)
- the string '0' (but not 0.0 nor '00')
- the empty string ''
- an array with zero elements
- an object with zero member variables (PHP 4 only)
- the special type NULL (including `unset` variables)

SimpleXML objects created from empty tags

Every other value is considered `TRUE` (including any resource)

When converting a boolean to a string,
- `TRUE` becomes "1"
- `FALSE` becomes ""

**Strings**

- PHP supports both single-quoted and double-quoted strings
- PHP also supports heredocs as a means to specify multi-line strings

```php
<<<identifier
document
identifier
identifer
might optionally be surrounded by double-quotes
identifier
might also be surrounded by single-quotes, making the string a nowdoc in PHP terminology

print <<<htmlLang="en-us">
<head><title>Multiline String</title></head>
</html>
print <<<<EOD
<body>Some text</body>
</html>
EOD;
```

PHP distinguishes between
- single-quoted strings and
- double-quoted strings

<table>
<thead>
<tr>
<th>single-quoted strings (‘taken literally’)</th>
<th>double-quoted strings (‘interpreted’/’evaluated’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>'hello' ~ hello</td>
<td>&quot;hello&quot; ~ hello</td>
</tr>
<tr>
<td>'don''t' ~ don't</td>
<td>&quot;don't&quot; ~ don't</td>
</tr>
<tr>
<td>'hello'' ~ hello&quot;</td>
<td>&quot;hello&quot; ~ hello&quot;</td>
</tr>
<tr>
<td>'backslash' ~ backslash\</td>
<td>&quot;backslash&quot; ~ backslash\</td>
</tr>
<tr>
<td>'glass\table' ~ glass\table</td>
<td>&quot;glass\table&quot; ~ glass\table</td>
</tr>
<tr>
<td>'glass\table' ~ glass\table</td>
<td>&quot;glass\table&quot; ~ glass\table</td>
</tr>
</tbody>
</table>

**Binary Assignments**

PHP also supports the standard binary assignment operators:

```
$a += $b;  // addition
$a -= $b;  // subtraction
$a *= $b;  // multiplication
$a /= $b;  // division
$a %= $b;  // modulus
$a //= $b; // same as $a = ($a // $b)
```

**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
  - `expr` is an expression that should evaluate to a value of a scalar type
    - in PHP 7, `expr` can also be an array
  - `case_insensitive` is an optional boolean argument, indicating whether the name of the constant is case-insensitive (default is FALSE)
  - returns `TRUE` on success or `FALSE` on failure

```php
define("PI",3.14159);
define("SPEED_OF_LIGHT",299792458,true);
define("ANIMALS",['bird','cat','dog']);
```
Types and Variables

Variables

Constants

- To use a constant we simply use its name
  ```php
define("PI",3.14159);
define("SPEED_OF_LIGHT",299792458,true);
// PHP
define("ANIMALS","["bird","cat","dog"]");
```

- PHP does not resolve constants within double-quoted strings (or here documents)

```php
$circumference = PI * $diameter;
$distance = SPEED_OF_LIGHT * $time;
myPet = ANIMALS[1];
```

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` that includes its type and value

```php
<?php
print "Type of 23: ",
gettype(23),
"\n";
print "Type of 23.0: ",
gettype(23.0),
"\n";
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 juggling and Type casting

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

```php
2 . "worlds"
"2" * 3
"1.23e2" + 0
"hello" * 3
"10hello5" * 5
"1.23e2"

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

- We can apply an identity function of the target type to force a type conversion

```php
12 + 1  ~ 12
12 + 0.0  ~ 12.0
"12" + 1  ~ "12.1"
"12" . "12.1"  ~ 12.0

Conversion of arrays to strings or numbers does not work
```

Type juggling and Type casting

- PHP also supports explicit type casting via `(type)`

```php
(int) "12"  ~ 12
(int) "10hello5"  ~ 10
(int) "1.23e2"  ~ 1

(float) "1.23e2"  ~ 123
(float) "1.23e2" + 1  ~ 124
(float) "1.23e2"  ~ 123
(float) "1.23e2" + 1  ~ 124

(0)  ~ TRUE
("foo")  ~ TRUE
(array) "foo"  ~ array(0 => "foo")
```

Further Reading

- Read
  Chapter 3: Introduction to PHP

- Read
  Language Reference: Types: Booleans
  Language Reference: Types: Integers
  Language Reference: Types: Floating Point Numbers
  Language Reference: Types: Strings