WEBD 236

Web Information Systems Programming

Week 5

Todd Whittaker and Scott Sharkey
(sharkesc@franklin.edu)

FRANKLIN UNIVERSITY

Agenda

- This week’s expected outcomes
- This week’s topics
- This week’s homework
- Upcoming deadlines
- Questions and answers

Week 5 Outcomes

- Employ string functions to manipulate character-based data
- Employ date and time functions to manipulate date-based data
- Discuss reasons to avoid and alternatives to user-entered HTML markup in web-applications.

Strings

- Strings
  - Single quoted strings: 'Hello $i\n' – no interpolation, no escape sequences
  - Double quoted strings: "Hello $i\n" – interpolation, escape sequences

Strings - Heredocs

- Heredocs and nowdocs

```
<?php
$arr = array('heredoc', 'double-quoted');
$message = <<< END
This is a ${arr[0]} that acts like
a ${arr[1]} string, and so
interpolation and escape sequences
are significant as are line breaks.
END;
print(nl2br($message));
?>
```

Strings - Heredocs

- Heredocs and nowdocs

```
<?php
$arr = array('heredoc', 'double-quoted');
$message = <<< END
This is a ${arr[0]} that acts like
a ${arr[1]} string, and so
interpolation and escape sequences
are significant as are line breaks.
END;
print(nl2br($message));
?>
```

Notice that NetBeans doesn't syntax-highlight the heredoc properly.

This is a heredoc that acts like
a double-quoted string, and so
interpolation and escape sequences
are significant as are line breaks.

Strings - Nowdocs

- Heredocs and nowdocs

```
<?php
$arr = array('nowdoc', 'single-quoted');
$message = <<< 'END'
This is a ${arr[0]} that acts like
a ${arr[1]} string, and so
interpolation and escape sequences
are not significant but line breaks are.
END;
print(nl2br($message));
?>
```

Strings - Nowdocs

- Heredocs and nowdocs

```
<?php
$arr = array('nowdoc', 'single-quoted');
$message = <<< 'END'
This is a ${arr[0]} that acts like
a ${arr[1]} string, and so
interpolation and escape sequences
are not significant but line breaks are.
END;
print(nl2br($message));
?>
```

Notice that NetBeans doesn't syntax-highlight the nowdoc properly either.

This is a ${arr[0]} that acts like
a ${arr[1]} string, and so
interpolation and escape sequences
are not significant but line breaks are.

String Escape codes

Code	Purpose
\\	Backslash
\'	Single quote
\"	Double quote
\$	Dollar sign
\n	Newline
\t	Tab
\r	Carriage return
\xhh	Hexadecimal char

HTML ignores whitespace, so you'd only see \t, \n, \r in "view source"

Strings and Characters

- ASCII values
  - Each character maps to an integer value
    - Ex: ‘A’ is 65, ‘Z’ is 90, etc. (see www.asciitable.com)
  - Use ord() with a character parameter to get the ASCII value back.
  - Use chr() with an integer parameter to get the character value back.

Looping and Strings

- Looping through strings
  - Use str_split() to convert a string to an array of 1-character strings.

```
function asciiEncode($str) {
    $result = '';
    $chars = str_split($str, 1);
    foreach ($chars as $char) {
        $result .= '&#' . ord($char) . ';';
    }
    return $result;
}
$encoded = asciiEncode("todd.whittaker@franklin.edu");
```

Looping and Strings

- Looping through strings
  - Use str_split() to convert a string to an array of 1-character st

```
function asciiEncode($str) {
    $result = '';
    $chars = str_split($str, 1);
    foreach ($chars as $char) {
        $result .= '&#' . ord($char) . ';';
    }
    return $result;
}
$encoded = asciiEncode("todd.whittaker@franklin.edu");
```

Produces:

todd.whittaker@franklin.edu

Learning a Language

- Two basic parts to learning any new programming language
  - Syntactical constructs
    - Control structures, key words, punctuation, data types, etc. I.e. rules of the language
  - Libraries
    - Pre-written routines (functions, objects) that you can use without writing them yourself.

Common String Functions

- Full list http://php.net/manual/en/ref.strings.php

Function	Purpose
`strlen($str)`	Returns the length of the string
`empty($str)`	Returns `TRUE` if the string is empty, `null`, or `'0'`.
`substr($str, $i [, $len])`	Returns a substring of `$str` starting at position `$i` (0-based indexing) and containing `$len` characters (at most).
`strpos($str1, $str2)`	Searches `$str1` for `$str2` and returns the integer value of where it is found or `FALSE` if it is not found. See also `stripos`, `strrpos`, `strripos`.

Common String Functions

- Full list http://php.net/manual/en/ref.strings.php

Function	Purpose
`str_replace($old, $new, $orig)`	Replace all occurrences of `$old` with `$new` in the string `$orig`. See also `str_ireplace`.
`ltrim($str), rtrim($str), trim($str)`	Trims whitespace from the string on the left, right, and both sides.
`str_pad($str, $len[, $pad[, $type]])`	Pads a string up to be up to `$len` in length using `$pad`.
`strtolower($str), strtoupper($str)`	Converts a string to lower or upper case respectively.

Common String Functions

- Full list http://php.net/manual/en/ref.strings.php

Function	Purpose
`explode($sep, $str)`	Splits a string into an array based on the `$sep` delimiter.
`implode($sep, $arr)`	Produces a single string from the array with `$sep` between elements.
`strcmp($str1, $str2)`, `strcasecmp($str1, $str2)`, `strnatcmp($str1, $str2)`,	Compares two strings, returning `-1` if `$str1 < $str2`, `0` if `$str1 == $str2`, and `1` if `$str1 > $str2`.

Common Math Functions

- Full list http://php.net/manual/en/ref.math.php

Function	Purpose
`abs($num)`	Returns the absolute value of `$num`.
`ceil($num)`	Returns the next integer greater than or equal to `$num`.
`floor($num)`	Returns the next integer less than or equal to `$num`.
`round($num[, $prec])`	Rounds `$num` to `$prec` decimal places.

Common Math Functions

- Full list http://php.net/manual/en/ref.math.php

Function	Purpose
`max($n1, $n2[, $n3...])`	Returns the maximum of all parameters. See also `min()`.
`pow($base, $exp)`	Raises `$base` to the power `$exp`.
`sqrt($num)`	Computes the square root of `$num`.
`mt_rand($low, $high)`	Returns a random integer between `[$low, $high]`

Formatting Output

- sprintf($format, $val1[, $val2 ...])
  - Returns a string with values inserted at given locations, using the format specified

```
$result = sprintf("Hello, %s, you have %10.2f dollars",
    'Fred', 13.245);
```

Hello, Fred, you have 13.24 dollars

Dates and Times

- Timestamp: an integer number of seconds since 12:00 AM, January 1, 1970 GMT.
- Can use functions to generate timestamps, format output, compute differences, etc.

```
$seconds = time();
$str = date("n/j/Y", $seconds);
```

$seconds is 1328123445,
$str is 2/1/2012

Dates and Times

- Use strtotime to parse date strings into timestamps

```
$seconds = strtotime("2012-02-01 4:35:21pm");
$str = date("g:i:s A, n/j/Y", $seconds);
```

$seconds is 1328110521 ,
$str is 4:35:21 PM, 2/1/2012

Dates and Times

- Use strtotime to parse date strings into timestamps

```
$seconds = strtotime("2012-02-01 4:35:21pm");
$str = date("g:i:s A, n/j/Y", $seconds);
```

```
$seconds = strtotime("+2 weeks 8am", time());
$str = date("g:i:s A, n/j/Y", $seconds);
```

$seconds is 1329289200 ,
$str is 8:00:00 AM, 2/15/2012

Dates and Times

- Can also use a DateTime object to manipulate dates.

```
$dueDate = new DateTime();
$dueDate -> modify("next Sunday 11:59:59pm");
$str = $dueDate -> format("g:i:s A, n/j/Y");
```

$str is 11:59:59 PM, 2/5/2012 based on today being Wednesday, 2/1/2012

Dates and Times

- A DateInterval object holds a difference between dates.

```
$date911 = new DateTime("2001-09-11 9:59:00am");
$today = new DateTime();
$delta = $date911 -> diff($today);
$str = $delta -> format("%R%yy %mm %dd %H:%I:%S");
```

$str has +10y 4m 21d 10:43:10 based on today being 2/1/2012

Given a DateInterval object, you can add or subtract that from a DateTime object as well.

Dates and Times

- A full listing on dates and times in PHP:

http://www.php.net/manual/en/ref.datetime.php

Handling text

- Have used htmlentities and htmlspecialchars to avoid injection vulnerabilities
  - But, it is desirable to allow some formatting, just not all formatting.
  - Special mini-languages for formatting
    - BBCode
    - Markdown

Handling text

- Have used htmlentities and htmlspecialchars to avoid injection vulnerabilities
  - But, it is desirable to allow some formatting, just not all for
  - Special mini-atting
    - BBCode
    - Markdown

What can injection in a web page let you do?

Handling text

```
function markdown($str) {
  $str = htmlspecialchars(ltrim($str),ENT_QUOTES);
  $str = preg_replace('/\*\*(.+)\*\*/u', '<b>$1</b>', $str);
  $str = preg_replace('/\*([^\*]+)\*/u', '<i>$1</i>', $str);
  $str = preg_replace('/#### ([^\n]*)\n/',"<h4>$1</h4>\n",$str);
  $str = preg_replace('/### ([^\n]*)\n/',"<h3>$1</h3>\n",$str);
  $str = preg_replace('/## ([^\n]*)\n/',"<h2>$1</h2>\n",$str);
  $str = preg_replace('/# ([^\n]*)\n/',"<h1>$1</h1>\n",$str);
  $str = preg_replace('/\[([^\]]+)\]\(([^\)]+)\)/',
      '<a href=\'$2\'>$1</a>', $str);
  $str = preg_replace('/([^\n\r]{2,})(?:(\r\n){2,}|
      \r{2,}|\n{2,}|$)/u', "<p>$1</p>\n\n", $str);
  return $str;
}
```

Handling text

```
function m
  $str = ht
  $str = pr
  $str = pr
  $str = pr
  $str = pr
  $str = pr
  $str = pr
  $str = pr
     '<a hr
  $str = pr
     \r{2,}|\
  return $s
}
```

Mini-markdown

This form lets you submit a mini-markdown document that will be rendered into HTML. This is safer than allowing HTML markup in your web applications. See Wikipedia for a complete Markdown syntax. Note, this is merely a demonstration, and not production-ready code. There are some complete Markdown libraries that are available for PHP.

```
# My Simple Markdown

This is a test of my simple markdown.  You can *emphasize* things with
asterisks, or **really emphasize** things with two asterisks.

Paragraphs are separated by two newlines.

You can even embed [simple links](http://en.wikipedia.org/wiki/Markdown).
```

Submit

My Simple Markdown

This is a test of my simple markdown. You can *emphasize* things with asterisks, or **really emphasize** things with two asterisks.

Paragraphs are separated by two newlines.

You can even embed simple links.

Handling text

```
function markdown($str) {
  $str = htmlspecialchars(ltrim($str),ENT_QUOTES);
  $str = preg_replace('/\*\*(.+)\*\*/u', '<b>$1</b>', $str);
  $str = preg_replace('/\*([^\*]+)\*/u', '<i>$1</i>', $str);
  $str = preg_replace('/#### ([^\n]*)\n/',"<h4>$1</h4>\n",$str);
  $str = preg_replace('/### ([^\n]*)\n/',"<h3>$1</h3>\n",$str);
  $str = preg_replace('/## ([^\n]*)\n/',"<h2>$1</h2>\n",$str);
  $str = preg_replace('/# ([^\n]*)\n/',"<h1>$1</h1>\n",$str);
  $str = preg_replace('/\[([^\]]+)\]\(([^\)]+)\)/',
      '<a href=\'$2\'>$1</a>', $str);
  $str = preg_replace('/([^\n\r]{2
      \r{2,}|\n{2,}|$)/u', "<p>$1</p>\
  return $str;
}
```

"Mini-markdown" for the simplest of formatting. See "minimarkdown.zip" example. Full markdown parsers are much better.

Handling text

- General rule
  - Escape all HTML markup
  - Store Markdown (or BBCode) text in the DB
  - Convert to HTML only when sent back to the browser.

Handling text

- Alternatives
  - Use a WYSIWYG HTML editor (such as TinyMCE or CKEditor) combined with…
  - An HTML sanitizer library (such as http://htmlpurifier.org/) to limit tags.
  - Store HTML directly in the DB without escaping.

Upcoming Deadlines

- Readings for next week
  - Chapters 11 and 12 in *PHP and MySQL*
- Assignments
  - Homework 4 due end of week 5
  - Lab 2 due end of week 7
- Next week:
  - Arrays, cookies, sessions

General Q & A

- Questions?
- Comments?
- Concerns?

FRANKLIN UNIVERSITY