INTRODUCTION TO WEB PROGRAMMING II

A

LECTURE NOTE

PREPARED

BY

MUHAMMAD SALISU ALI

FEDERAL UNIVERSITY DUTSE

16 APRIL, 2021

How to work with strings

- How to create strings
- There are four ways to create strings. You can use single quotes, double quotes, the heredoc syntax, or the nowdoc syntax.
- When you create a string using double quotes or the **heredoc** syntax, the PHP engine performs variable substitution. Variable substitution replaces the variable name in the string with its value. If necessary, this process converts the variable's value to a string. This is also known as *interpolation*.
- When working with variable substitution, if the variable name is adjacent to text that's interpreted as part of the variable name, the variable substitution won't work correctly. In that case, you can clarify the variable name by enclosing the name, not including the dollar sign, inside braces.
- Creating a **nowdoc** is similar to creating a heredoc, but you enclose the name for the nowdoc in single quotes.
- When you create a heredoc or a nowdoc, the identifier must be immediately followed by a new line. Similarly, the closing identifier must be on a line by itself, and it can't begin with any spaces or tabs. After the closing identifier, the only character that's allowed is the semicolon that ends the statement. Otherwise, a syntax error occurs.

- Assign strings with single quotes

```
$language = 'PHP';
$message = 'Welcome to ' . $language;              // 'Welcome to PHP'
$query = 'SELECT firstName, lastName FROM Users';  // Spans multiple lines
```

- Assign strings with double quotes
  - Using variable substitution

```
$language = "PHP";
$message = "Welcome to $language";
```

- Using braces with variable substitution

```
$count = 12; $item = "flower";
$message1 = "You bought $count $items.";      // 'You bought 12 .'
$message2 = "You bought $count ${item}s.";    // 'You bought 12 flowers.'
```

- Assign a string with a heredoc

  $language = 'PHP';

  $message = <<<MESSAGE

  The heredoc syntax allows you to build multi-line strings in $language. Inside, it acts like a double-quoted

  string and performs variable substitution.

  MESSAGE;

- Assign a string with a nowdoc

  $message = <<<'MESSAGE'

  The nowdoc syntax also allows you to build multi-line strings in PHP. However, no variable substitution takes place inside the nowdoc string. This is similar to single-quoted strings.

  MESSAGE;

How to use escape sequences

- Escape sequences provide a way to insert special characters into text strings. These escape sequences work differently depending on the type of the string.
- Escape sequences only used in some strings

  \\ - Backslash - All strings except nowdocs

  \' - Single quote - Single-quoted strings

  \" - Double quote - Double-quoted strings
- Escape sequences used in double-quoted strings and heredocs

  \$ Dollar sign

  \n New line

  \t Tab

  \r Carriage return

  \f Form feed

  \v Vertical tab

  \ooo Character with the specified octal value

  \xhh Character with the specified hexadecimal value

- Examples of escape sequences

```
$dir = 'C:\\xampp\\php';                    // C:\xampp\php
$name = 'Mike\'s Music Store';              // Mike's Music Store
$quote = "He said, \"It costs \$12.\"";     // He said, "It costs $12."
$comment1 = "This is a\nmulti-line string."; // This is a multi-line string.
$comment2 = 'Not a\nmulti-line string.';    // Not a\nmulti-line string
```

- The htmlentities function

htmlentities($str [, $quotes]**)** - Returns a string with all special HTML characters converted to the HTML entity. You can use the $quotes parameter to control how single and double quotes are converted.

- Examples of the htmlentities function
- An example that doesn't use the htmlentities function

```
$copyright1 = "\xa9 2014";      // Result is '© 2014'
echo $copyright1;               // Some browsers display ? 2014
```

An example that uses the htmlentities function

```
$copyright2 = htmlentities("\xa9 2014");  // Result is '&copy; 2014'
echo $copyright2;                         // Browsers display © 2014
```

How to work with string length and substrings

- The length of a string is the number of characters in the string.
- A substring is a part of a string.
- To specify the position of a character from the left side of a string, use a positive integer value where 0 is the first character, 1 is the second character, and so on. To specify a position from the right side, use a negative integer value.
- Functions for working with string length and substrings

  empty($str**)** -Returns TRUE if $str is an empty string (“”), a NULL value, or “0” (0 as a string). This function also returns TRUE if $str isn’t set.

  strlen($str**)** - Returns the length of the string.

  substr($str, $i[, $len]**)** - Returns a substring of $str starting at the position specified by $i and containing the number of characters specified by $len. If $len is omitted, the function returns the substring from $i to the end of the string.

- Code that checks a string to determine if a string is empty

```
if (empty($first_name)) {
        $message = 'You must enter the first name.';
}
```

- Code that gets the length of a string and two substrings

```
$name = 'Ray Harris';
$length = strlen($name);          // $length is 10
$first_name = substr($name, 0, 3); // $first_name is 'Ray'
$last_name = substr($name, 4);    // $last_name is 'Harris'
$last_name = substr($name, -6);   // $last_name is 'Harris'
```

- Code that formats a phone number in two ways

```
$phone = '5545556624';
$part1 = substr($phone, 0, 3);
$part2 = substr($phone, 3, 3);
$part3 = substr($phone, 6);
$format_1 = $part1 . '-' . $part2 . '-' . $part3; // 554-555-6624
$format_2 = '(' . $part1 . ') ' . $part2 . '-' . $part3; // (554) 555-6624
```

- Code that displays each letter in a string on a separate line

```
$input = 'JAN';
for ($i = 0; $i < strlen($input); $i++) {
    $vert_str .= substr($input, $i, 1);
    $vert_str .= '<br>';
}
```

How to search a string

- Functions that search a string

  strpos($str1, $str2[, $offset]**)** - Searches $str1 for an occurrence of $str2. If $str2 is found, returns an integer value for the position. If $str2 isn't found, returns FALSE. By default, the search starts at position 0, but you can use $offset to specify the start position.

  stripos($str1, $str2[, $offset]**)** - A version of strpos that's case-insensitive.

  strrpos($str1, $str2[, $offset]**)** - A version of strpos that searches in reverse, from the end of the string to the start.

  strripos($str1, $str2[, $offset]**)** - A version of strrpos that's case-insensitive.

- Code that searches a string for spaces

```
$name = 'Martin Van Buren';
$i = strpos($name, ' ');      // $i is 6
$i = strpos($name, ' ', 7);   // $i is 10 – use offset to find second space
$i = strrpos($name, ' ');     // $i is 10 – search string in reverse
```

- Code that searches a string for a substring

```
$name = 'Martin Van Buren';
$i = strpos($name, 'Van');      // $i is 7
$i = strpos($name, 'van');      // $i is FALSE – case-sensitive
$i = stripos($name, 'van');     // $i is 7     – case-sensitive
$i = strripos($name, 'A');      // $i is 8     - reverse + case-insensitve
```

- Code that splits a string into two substrings

```
$name = 'Ray Harris';
$i = strpos($name, ' ');
if ($i === false) {
        $message = 'No spaces were
}  else {
        $first_name = substr($name,0, $i);   // $first_name = Ray
         $last_name = substr($name,$i+1);  // $last_name = Harris
}
```

- Functions that replace part of a string

  str_replace($str1, $new, $str2**)** - Returns a new string with all occurrences of $str1 in $str2 replaced with $new. It is case sensitive.

  str_ireplace($str1, $new, $str2**)** -A version of str_replace that's case-insensitive.

- Code that replaces periods with dashes in a phone number

  ```
  $phone = '554.555.6624';

  $phone = str_replace('.', '-', $phone); // $phone is '554-555-6624'
  ```

  Code that replaces one string with another string

  ```
  $message = 'Hello Ray';

  $message = str_ireplace('hello', 'Hi', $message);      // $message is 'Hi Ray'
  ```

- How to modify strings