COMP519 Web Programming

Lecture 21: PHP (Part 3)

Handouts

Ullrich Hustadt

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

Contents

1. Control Structures
   - Statement Groups
   - Conditional Statements
   - Switch Statements
   - While- and Do While-loops
   - For-loops
   - Exceptions and Error Handling
2. Revision and Further Reading

Control Structures

PHP control structures

- statement groups
- conditional statements
- switch statements
- while- and do while-loops
- for-loops
- try, throw, catch finally statements

are mostly identical to those of Java

Control Structures: Statement Groups

A statement group is a sequence of zero or more statements delimited by a pair of curly brackets

```
{
   statements
}
```

- It allows to use multiple statements where PHP expects only one statement
- The last statement in a statement group does not need to be terminated by a semi-colon

```
{
   $x = 1
   $y = $x++
}
```

Conditional Statements

The general format of conditional statements is very similar but not identical to that in Java and JavaScript:

```
if (condition)
   statement
elseif (condition)
   statement
else
   statement
```

- the elseif-clause is optional and there can be more than one
  Note: `elseif` instead of `elif` or `else if`
- the else-clause is optional but there can be at most one

Conditional Statements/Expressions

- PHP allows to replace curly brackets with a colon : combined with an endif at the end of the statement:

```
if (condition):
    statements
elseif (condition):
    statements
else:
    statements
endif
```

This also works for the switch statement in PHP

However, this syntax becomes difficult to parse
when nested conditional statements are used and is best avoided

- PHP also supports conditional expressions

```
condition ? if_true_expr : if_false_expr
```

Switch Statement

A switch statement in PHP takes the following form

```
switch (expr) {
   case expr1:
        statements
        break;
   case expr2:
        statements
        break;
   default:
        statements
        break;
}
```

- there can be arbitrarily many case-clauses
- the default-clause is optional but there can be at most one, it does not need to be last
- *expr* is evaluated only once and then compared to *expr1*, *expr2*, etc using (loose) equality ==
- once two expressions are found to be equal the corresponing clause is executed
- if none of *expr1*, *expr2*, etc are equal to *expr*, then the default-clause will be executed
- break 'breaks out' of the switch statement
- if a clause does not contain a break command, then execution moves to the next clause

Switch Statement: Example (1)

Example: Classic Adventure Game Commands

```
switch ($command) {
  case "North":
     $y += 1; break;
  case "South":
     $y -= 1; break;
  case "West";
     $x -= 1; break;
  case "East":
     $x += 1; break;
  case "Search":
     if (($x = 5) && ($y = 3))
        echo "Found a treasure\n";
     else
        echo "Nothing here\n";
     break;
  default:
     echo "Not a valid command\n";
}
```

Switch Statement: Example (2)

Not every case-clause needs to have associated statements

```
switch ($month) {
  case 1:   case 3:   case 5:   case 7:
  case 8:   case 10:  case 12:
     $days = 31;
     break;
  case 4:   case 6:   case 9:   case 11:
     $days = 30;
     break;
  case 2:
     $days = 28;
     break;
  default:
     $days = 0;
     break;
}
```

While- and Do While-loops

PHP offers while-loops and do while-loops

```
while (condition)
    statement

do
    statement
while (condition);
```

Example:

```
// Compute the factorial of $number
$factorial = 1;
do
    $factorial *= $number--;
while ($number > 0);
```

For-loops

- for-loops in PHP take the form

```
for (initialisation; test; increment)
    statement
```

- In PHP *initialisation* and *increment* can consist of more than one statement, separated by commas instead of semicolons

Example:

```
for ($i = 3, $j = 3; $j >= 0; $i++, $j--)
   echo "$i - $j - ", $i*$j, "\n";
3 - 3 - 9
4 - 2 - 8
5 - 1 - 5
6 - 0 - 0
```

Break and Continue

- The break command can also be used in while-, do while-, and for-loops and discontinues the execution of the loop

```
while ($value = array_shift($data) {
  $written = fwrite($resource,$value);
  if (!$written) break;
}
```

- The continue command stops the execution of the current iteration of a loop and moves the execution to the next iteration

```
for ($x = -2; $x <= 2; $x++) {
   if ($x == 0) continue;
   printf("10 / %2d = %3d\n",$x,(10/$x));
}
```

```
10 / -2 =  -5
10 / -1 = -10
10 /  1 =  10
10 /  2 =   5
```

Exceptions and error handling

- PHP distinguishes between Exceptions and Errors
- In PHP 7 both are subclasses of Throwable but not in PHP 5
- A `try ... catch ...` statement allows for exception (throwable) handling
- Since PHP 5.5, a `finally` clause can be added

```
try { statements }
catch (Exception $e) { statements }
finally { statements }
```

Exceptions and error handling

```
$x = "A";
try {
  if ((is_int($x) || is_float($x)) && is_finite($x))
    $y = round($x,2);
  else
    throw new Exception("Not a number");
} catch (Exception $e) {
  echo "Caught: $e\n";
  $y = 0;
} finally {
  echo "y = $y\n";
}
```

```
Caught: Exception: Not a number in try.php:7
Stack trace:
#0 {main}
y = 0
```

Exceptions and error handling

- PHP distinguishes between exceptions and errors
- Errors must be dealt with by an error handling function
  ('Division by zero' produces an error not an exception)
- To take advantage of `try ... catch ...` statements to handle errors, one can turn errors into exceptions

```
function exception_error_handler($errno, $errstr,
  $errfile, $errline ) {
  throw new ErrorException($errstr, $errno,
                           0, $errfile, $errline); }
set_error_handler("exception_error_handler");
```

```
$x = 0;
try {
  $y = 1/$x;
} catch (Exception $e) {
  echo "Caught: $e\n";
}
```

```
Caught: ErrorException: Division by zero in try.php:10
```

Revision and Further Reading

- Read
  - Chapter 4: Expressions and Control Flow in PHP: Conditionals, Looping

  of R. Nixon: Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5. O'Reilly, 2018.
- Read
  - Language Reference: Control Structures
    `http://uk.php.net/manual/en/language.control-structures.php`
    including `if`, `else`, `elseif`, `while`, `do-while`, `for`, `foreach`, `break`, `continue`, `switch`, `return`, `require`, `require_once`, `include`, `include_once`

  of P. Cowburn (ed.): PHP Manual. The PHP Group, 25 Oct 2019. `http://uk.php.net/manual/en` [accessed 26 Oct 2019]