Kontrollstrukturen

Colletta Hagert

Unister

Überblick

- Einführung
- if-Anweisung
- switch-case-Anweisung
- do-schleife
- for-schleife
- foreach
- break
- Alternative Syntax für Kontrollstrukturen

Einführung

Ziel

Wissen,

- **wie Bedingte Anweisungen programmiert werden**
- **welche Möglichkeiten der Fallunterscheidung es gibt**
- **wie Schleifen programmiert werden**
- **wie die Ausführung von Schleifen beendet werden können**

Unister

Ein PHP-Skript besteht aus einer Abfolge von Anweisungen, die in der Regel mit einem Semikolon enden.

PHP kennt folgende Anweisungen:

- **Zuweisungen**
- **Funktionsaufrufe**
- **Schleifen**
- **Bedingungen oder sogar**

Geschweifte Klammern werden benutzt, um Anweisungen zusammenzufassen

=> Bildung einer Anweisungsgruppe

Warum sind Kontrollstrukturen notwendig?

Die Anweisungen in einem PHP-Skript werden in der Reihenfolge ausgeführt, in der sie angegeben sind, falls keine Anweisung einen Befehl enthält, der diese Reihenfolge ändert.

=> Ein PHP-Skript wird in diesem Fall sequenziell abgearbeitet

Um von der sequenziellen Abarbeitung der Befehle abzuweichen werden Kontrollstrukturen eingesetzt

Anweisungen über Bedingungen auswählen

- Eine Bedingung ist eine Möglichkeit, den Ablauf eines Skripts durch Entscheidung zu beeinflussen.

- In einer Bedingung werden Ausdrücke miteinander verglichen bzw. ein logischer Ausdruck ausgewertet.

- Zur Formulierung der Bedingungen stehen Vergleichsoperatoren und die logischen Operatoren zur Verfügung

- Eine Bedingung kann entweder mit „Ja“ beantwortet werden (true bzw. wahr) oder mit „Nein“ (false bzw. false).

=> Welche Anweisung des Programms abgearbeitet werden und welche nicht

if-Anweisung

- **Das if-Konstrukt ermöglicht die bedingte Ausführung von Kodefragmenten.**

- **Ist eines der wichtigsten Features vieler Programmier-sprachen, so auch in PHP**

```
if (Bedingung) {
    Anwesungsblock;
}
```

Evaluiert Bedinung zu TRUE wird/werden die Anweisung(en) von PHP ausgeführt, anderenfalls wird es ignoriert.

Beispiel mit zwei Anweisungen

```
<?php

if ($a > $b) {
    echo "a ist größer als b";
    $b = $a;
}
```

else-Konstrukt

- Bietet die Möglichkeit eine Anweisung auszuführen, wenn eine bestimmte Bedingung erfüllt ist und eine andere Anweisung, wenn dies nicht der Fall ist.
- else erweitert eine if-Anweisung um eine weitere

Anweisung die dann ausgeführt werden soll wenn der Ausdruck in der if-Anweisung zu FALSE ausgewertet wird.

```
if (bedingung) {
   Anweisungsblock1;
} else {
   Anweisungsblock2;
}
```

else-Konstrukt

Beispiel

```
if ($a > $b) {
    echo "a is größer als b";
} else {
    echo "a ist gleich groß wie b oder kleiner als b";
}
```

elseif/else if-Konstrukt

- eine Kombination aus if und else
- Wie else erweitert es eine if-Kontrollstruktur, um alternative Befehle auszuführen, wenn die urprüngliche if-Bedingung nicht zutrifft.
- Die Alternativ-Befehle werden nur ausgeführt, wenn die elseif-Bedingung zutrifft.
- Innerhalb einer if-Kontrollstruktur können mehrere elseif-Strukturen benutzt werden.

=> Die erste, deren Bedingung zutrifft, wird ausgeführt.

elseif/else if-Konstrukt

```
if (Bedingung1) {
    Anweisung(en);
} elseif (Bedingung2) {
    Anweisung(en);
} else {
    Anweisung(en);
}
```

elseif/else if-Konstrukt

Beispiel

```
<?php
if ($a > $b) {
    echo "a is größer als b";
} elseif ($a == $b) {
    echo "a ist gleich groß wie b";
} else {
    echo "a ist kleiner als b";
}
```

Verschachtelte if-Konstrukt

Oft ist eine Auswahl von einer Bedingung abhängig, die wiederum von einer vorhergehenden Bedingung abhängig ist.

```
if (Bedingung1) {
    Anweisungsblock1;
    if(Bedingung2) {
        Anweisungsblock2;
    } else {
        Anweisungsblock3;
    }
} else {
    Anweisungsblock4;
}
```

switch-case-Anweisung

- **Mit der Funktion switch lassen sich abhängig vom Inhalt einer Variablen verschiedene Code-Abschnitte ausführen.**

- **Die Funktion arbeitet also ähnlich wie mehrere if-Anweisungen, die die gleiche Variable betreffen**

- **Für jede case-Anweisung ist nur ein Vergleichswert zulässig.**

- **Optional kann mit der default-Anweisung ein Code-Abschnitt definiert werden, der ausgeführt wird, sofern keiner der Vergleiche in den case-Anweisungen zutrifft**

```
switch($variable) {
    case Wert 1:
        auszuführender Code, sofern Variable gleich Wert 1
    break;
    case Wert 2:
         auszuführender Code, sofern Variable gleich Wert 2
     break;
     [weitere Fälle von case]
    default:
        auszuführender Code, wenn kein Fall von case  zutrifft
}
```

```
<?php
if ($i == 0) {
    echo "i equals 0";
} elseif ($i == 1) {
    echo "i equals 1";
} elseif ($i == 2) {
    echo "i equals 2";
}
```

```php
<?php
switch ($i) {
    case 0:
        echo "i equals 0";
        break;
    case 1:
        echo "i equals 1";
        break;
    case 2:
        echo "i equals 2";
        break;
}
?>
```

while-Schleife

Die Schleife weist PHP an, die untergeordnete(n) Anweisung(en) wiederholt auszuführen, solange die while-Bedingung zutrifft.

Die Bedingung wird jedes Mal am Anfang der Schleife überprüft

Wenn die while-Bedingung von Anfang an nicht zutrifft, werden die untergeordneten Anweisung nicht ein einziges Mal ausgeführt.

Mehrere Anweisungen können innerhalb der selben Schleife mit geschleiften Klammern gruppiert werden

```
while (bedingung) {
   Anweisung(en);
}
```

Beispiel

```
<?php
$i = 1;
while ($i <= 10) {
   echo $i++;  /* der ausgegebene Wert ist $i bevor
                  er erhöht wird (post-increment) */
}
```

do-while-Kontrukt

do-while-Schleifen sind sehr ähnlich zu while-Schleifen, außer dass der Wahrheitsausdruck erst am Ende eines jeden Durchlaufs statt zu dessen Beginn geprüft wird.

Unterschied zu while-Kontrukt:

- **Die do-while-Schleife wird garantiert mindestens einmal durchlaufen**
- **Eine reguläre while-Schleife muss nicht zwingend immer ausgeführt wird**

=> Evaluiert die Bedingung zu Beginn der Durchlauf zu FALSE, wird die Verarbeitung der Schleife sofort abgebrochen

```
do {
    Anweisung(en);
} while (bedingung);
```

Beispiel

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

for-Schleife

Die for-Schleifen sind die komplexesten Schleifen in PHP.

```
for (Startanweisung; Bedingung; Aktualisierung)  {
   Anweisung(en);
}
```

- Der erste Ausdruck „Startanweisung“ wird vor Ausführung der Schleife ausgeführt.
- Am Anfang jedes Schleifendurchlaufs wird die Anweisung „Bedingung“ ausgeführt.
- Am Ende jedes Schleifendurchlaufs wird die Anweisung „Schleifenanweisung“ ausgeführt.
- Jede der Anweisungen kann leer sein oder mehrere durch Kommata getrennte Anweisungen enthalten.

for-Schleifen

Beispiel

```
<?php
/* Beispiel 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

for-Schleifen

Beispiel

```
for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
```

foreach-Schleife

- **Die foreach-Schleife ermöglicht es, auf einfache Weise ein Array zu durchlaufen.**
- **foreach funktioniert nur in Verbindung mit Arrays.**
- **Mithilfe von foreach wird das Durchlaufen eines Arrays wesentlich vereinfacht.**
- **Gegenüber der while-Schleife mit list und each ist die foreach-Schleife syntaktisch deutlich im Vorteil.**

Unister

Syntax

```
foreach (array_ausdruck as $value) {
   Anweisung(en);
}
```

```
foreach (array_ausdruck as $key => $value) {
   Anweisung(en);
}
```

Unister

Beispiel

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}

// $arr is now array(2, 4, 6, 8)

unset($value); // break the reference with the last element
```

Unister

Beispiel

```
/* foreach example 3: key and value */
$a = array(
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach ($a as $k => $v) {
    echo "\$a[$k] => $v.\n";
}
```

break

- **break beendet die Ausführung der aktuellen for-, foreach-, while-, do-while- oder switch-Struktur.**

- **break akzeptiert ein optionales numerisches Argument, das angibt, aus wie vielen der es umschließenden verschachtelten Strukturen ausgebrochen werden soll.**

Beispiel

```
<?php
$arr = array('eins', 'zwei', 'drei', 'vier', 'stop', 'fünf');
while (list(, $val) = each($arr)) {
    if ($val == 'stop') {
        break;    // Sie könnten hier auch 'break 1;' schreiben.
    }
    echo "$val<br />\n";
}
```

continue

- **continue wird innerhalb von Schleifen verwendet, um den Rest des aktuellen Schleifendurchlaufs abzubrechen und mit der Auswertung der nächsten Bedingung fortzufahren, um dann den nächsten Durchlauf zu beginnen.**

- **continue akzeptiert ein optionales numerisches Argument, das angibt, wie viele Ebenen umschließender Schleifen bis zu ihrem Ende übersprungen werden sollen.**

Beispiel

```
for($k=0;$k<2;$k++)
{//First loop
   for($j=0;$j<2;$j++)
   {//Second loop
     for($i=0;$i<4;$i++)
     {//Third loop
       if($i>2)
          continue 2; // If $i >2 ,Then it skips to the Second loop(level 2), And starts the next step,
       echo "$i\n";
      }
   }
}
```

Alternative Syntax für Kontrollstrukturen

- **PHP bietet eine alternative Syntax für einige seiner Kontrollstrukturen an:**

- **if, while, for, foreach und switch**

- **Die Grundform der alternativen Syntax ist ein Wechsel der öffnenden Klammer gegen einen Doppelpunkt (:) und der schließenden Klammer in endif;, endwhile;, endfor;, endforeach; respektive endswitch;.**

Beispiel

```
if ($a == 5):
        echo "a gleich 5 <br />";
        echo "...";
elseif ($a == 6):
        echo "a gleich 6 <br />";
        echo "!!!";
else:
        echo "a ist weder 5 noch 6";
endif;
```

return

- **return() beendet augenblicklich die Ausführung der Funktion und übergibt den Parameter als Rückgabewert der Funktion.**

- **return() beendet ebenfalls die Ausführung von Code innerhalb von eval() oder einer Datei.**