Arrays

Mittwoch, 18. September 2019 08:39

Oh neeiin! Nur *ein Buch* zum Verkauf anzubieten erscheint mir nun doch nicht besonders raffiniert zu sein. Doch wie soll ich nun bloß ein weiteres Buch auf der Serverseite mit PHP anbieten können?

Arrays

Ihre Aufgabe:

Klären Sie mit Ihrem Banknachbarn, was unseren Freund Homer Kopfzerbrechen bereitet und halten Sie Ihre Erkenntnisse im nachfolgenden Feld fest.

Show

Informationstext zu Arrays:

Bitte lesen Sie sich nun nachfolgenden Informationstext durch und *markieren* Sie sich die Ihrer Meinung nach wichtigen Stellen.

Wenn Sie schon Erfahrung mit Programmiersprachen haben, werden Sie zwei Datenstrukturen kennen, die sehr viel gemeinsam haben und besonders nützlich sind: *Listen* und *Maps*. Eine Liste ist eine geordnete Ansammlung von Elementen, während eine Map eine Sammlung von *Schlüssel-Wert-Paaren* darstellt. Sehen wir uns ein Beispiel an:

List: [„Harry“, „Ron“, „Hermione“]

Map: { „name“: „James Potter“, „status“: „dead“ }

In PHP haben wir Arrays, die beide vorher angesprochenen Datenstrukturen implementieren.

Arrays anlegen

Man kann auf verschiedene Arten Arrays erzeugen. Man kann ein leeres Array anlegen oder bereits am Anfang mit Daten füllen.

```php
<?php

$empty1 = [];

$empty2 = array();

$names1 = ['Harry', 'Ron', 'Hermione'];

$names2 = array('Harry', 'Ron', 'Hermione');

$status1 = [ 'name' => 'James Potter', 'status' => 'dead' ];

$status2 = array('name' => 'James Potter', 'status' => 'dead'
);
```

Im diesem Beispiel erzeugen wir sowohl eine Liste($names) als auch eine Map($status).

Später werden wir sehen, dass Listen wie Maps behandelt werden. Intern ist der $names1 eine Map, ihre Schlüssel sind sortierte Ziffern! Eine andere Darstellung für $names1 zeigt den Zusammenhang.

$names1 = [0 => ‚Harry', 1 => ‚Ron', 2 => ‚Hermione'];

Die Schlüssel eines Arrays können beliebige alphanumerische Werte sein, wie beispielsweise Buchstaben oder Zahlen. Die Werte eines Arrays können beliebig sein: Strings, Ziffern, Boolesche Werte, andere Arrays und so weiter. Sehen sie folgendes Beispiel:

```php
<?php
$books = [ '1984' => [ 'author' => 'George Orwell', 'finished'
=> true, 'rate' => 9.5 ],
        'Romeo' and Juliet' => [ 'author' => 'William Shakespe
are','finished' => false ] ];

var_dump($books);
```

Dieser Array ist eine Liste, die zwei array-basierte Maps enthält. Jede Map enthält verschiedene Werte wie String, Doubles und Booleans.

Füllen von Arrays

Arrays sind nicht unveränderlich, sondern können auch nach ihrer Erzeugung geändert werden. Sie können das Array dabei wie eine Liste oder wie eine Map ansprechen. Bei einer Map sprechen Sie den Schlüssel an bzw. formulieren Sie einen neuen Schlüssel. Bei einer Liste fügen Sie einfach ein weiteres Element hinzu.

```php
<?php
$names = ['Harry', 'Ron', 'Hermione'];
$status = [ 'name' => 'James Potter', 'status' => 'dead' ];
$names[] = 'Neville';

$status['age'] = 32;
print_r($names, $status);
```

Wenn man Elemente aus einem Array herausnehmen will, kann man die unset-Funktion verwenden.

```php
<?php
$status = [ 'name' => 'James Potter', 'status' => 'dead' ];
unset($status['status']);
print_r ($status);
```

Ergebnis: Der neue $status-Array enthält nur noch den Schlüssel name.

Ausgabe eines Arrays im Rahmen einer Schleife

```php
<?php
$status = [ 'name' => 'James Potter', 'status' => 'dead' ];
foreach($status as $key => $value)
{
    echo $key . ": " . $value;
}
```

Zugriff auf Array-Elemente

Der Zugriff auf die Inhalte eines Arrays ist leicht, da nur der Schlüssel angegeben werden muss. Sie müssen deshalb verstehen, wie Listen funktionieren. Intern werden Listen wie Maps behandelt, die einen sortierten numerischen Schlüssel haben. Dieser beginnt immer bei 0 und endet damit bei n-1.

Sie können jeden Schlüssel einem vorhandenen Array hinzufügen, auch wenn vorher nur numerische Schlüssel angegeben wurden. Wenn man einen Wert hinzufügen will, wird ihn PHP hinter den letzten numerischen Schlüssel anhängen. Die Ausgabe eines bestimmten Teils eines Arrays wird durch die Angabe seinen Schlüssel ermöglicht:

```php
<?php
$names = ['Harry', 'Ron', 'Hermione'];
print_r($names[1]);
//prints 'Ron'
```

Der Zugriff auf einen nicht-existierenden Schlüssel wird einen NULL-Wert zurückgeben.

empty() und isset() Funktionen

Es gibt zwei nützliche Funktionen, um Informationen über den Inhalt eines Arrays zu erhalten. Die empty-Funktion gibt true zurück, wenn ein Array keinerlei Inhalt hat. Die isset-Funktion gibt an, ob die angefragte Position im Array vorhanden ist oder nicht.

Suchen in einem Array

Eine der am häufigsten benutzen Funktionen im Zusammenhang mit Arrays ist wohl *in_array()*. Die Funktion hat zwei Parameter, nämlich den gesuchten Wert und den Array, in dem gesucht werden soll. Die Funktion gibt True zurück, wenn der Wert gefunden wird, ansonsten False. Noch nützlicher ist manchmal die Funktion *array_search()*. Sie gibt die Position der Fundstelle zurück oder False.

Sortieren von Arrays

Ein Array kann nach verschiedenen Kriterien sortiert werden. Die Default-Sortierung (Reihenfolge der Eingabe), kann durch die Sortierung nach Schlüsseln oder den Werten ersetzt werden. Beim Sortieren kann noch darauf geachtet werden, ob die Schlüssel erhalten werden sollen oder nicht. Eine komplette Übersicht erhalten Sie unter *http://php.net/manual

Suchen in einem Array

Eine der am häufigsten benutzen Funktionen im Zusammenhang mit Arrays ist wohl *in_array()*. Die Funktion hat zwei Parameter, nämlich den gesuchten Wert und den Array, in dem gesucht werden soll. Die Funktion gibt True zurück, wenn der Wert gefunden wird, ansonsten False. Noch nützlicher ist manchmal die Funktion *array_search()*. Sie gibt die Position der Fundstelle zurück oder False.

Sortieren von Arrays

Ein Array kann nach verschiedenen Kriterien sortiert werden. Die Default-Sortierung (Reihenfolge der Eingabe), kann durch die Sortierung nach Schlüsseln oder den Werten ersetzt werden. Beim Sortieren kann noch darauf geachtet werden, ob die Schlüssel erhalten werden sollen oder nicht. Eine komplette Übersicht erhalten Sie unter *http://php.net/manual/en/array.sorting.php*.

Weitere Funktionen bei Arrays (verkürzt):

`array_flip($feld)`	Im Array werden Indizes mit Werten vertauscht.
`array_key_exists(wert, $feld)`	Prüfung, ob ein Schlüssel in einem Array vorhanden ist.
`array_keys($feld)`	Liefert die Indizes des angegebenen Arrays zurück.
`array_merge($feld1, $feld2…)`	Fügt die Elemente mehrerer Arrays zu einem Array zusammen.
`array_push($feld, werte)`	Das Array wird um den oder die angegebenen Werte am Ende des Arrays erweitert.
`array_search(wert, $feld)`	Das Array wird nach dem angegebenen Wert durchsucht.
`array_sum($feld)`	Addiert die Werte des Arrays und liefert das Ergebnis zurück.
`array_unique($feld)`	Es wird ein neues Array erstellt, aus dem doppelte Werte des angegebenen Arrays gelöscht wurden.
`array_values($feld)`	Alle Werte des Arrays werden zurückgeliefert.
`count($feld)`	Gibt die Anzahl der Elemente des Arrays zurück.
`sort($feld)` `rsort($feld)`	Sortiert die Werte des angegebenen Arrays aufsteigend bzw. absteigend.
`ksort($feld)` `krsort($feld)`	Sortiert das angegebene Array nach Schlüssel aufsteigend bzw. absteigend.

Alle Arrayfunktionen können Sie der PHP Dokumentation je nach Bedarf entnehmen.

Fragen an Dr. Simpson (jun.)

Given the following C++ statements:

```
int  val = 0;
int& ir  = val;
auto x   = ir;
```

What type is x?

Show

F: Ist es auch möglich, in PHP typsichere Arrays zu erzeugen?

Show

F: Woran kann ich eigentlich erkennen, ob ich ein indiziertes oder assoziatives Array nutzen soll?

Antwort