Ordner auslesen mit PHP

== PHP Abwärtskompatiblität ==
Mit PHP 5 ist eine neue Funktion namens scandir eingeführt worden um Ordner auszulesen. Soweit ich das überblicke arbeitet sie besser mit Streams. Die einzige offensichtliche Änderung ist dagegen nur eine sortierte Liste als Rückgabewert. Um den Code PHP4 kompatibel zu halten, definieren wir einen Nachbau der Funktion als Fallback Lösung, falls die Funktion noch unbekannt ist. In den Beispielen inkludieren wir die Funktion über die Datei scandir_fallback.php.

Quellcode

```
1.  if (!function_exists('scandir')) {
2.  function scandir($directory, $sorting_order=0) {
3.  if(!is_dir($directory)) {
4.  return false;
5.  }
6.  $files = array();
7.  $handle = opendir($directory);
8.  while (false !== ($filename = readdir($handle))) {
9.  $files[] = $filename;
10. }
11. closedir($handle);
12. if($sorting_order == 1) {
13. rsort($files);
14. } else {
15. sort($files);
16. }
17. return $files;
18. }
19. }
```

Alles anzeigen

== Order auslesen ganz einfach ==
Es ist ganz einfach einen Ordner mit PHP auszulesen. Hier zeige ich euch die einfachste Variante.
Tragt für den Ordner einen . (Punkt) ein, dann wird der Inhalt des aktuellen Ordners ausgegeben. Mit .. (Punkt-Punkt) lest ihr den Ordner oberhalb des eigenen aus. Wer schon einmal auf der Konsole gearbeitet hat, kennt das.

Quellcode

```
1.  <?php
2.  require_once 'scandir_fallback.php';
3.  $folder = 'unterordner';
4.  $fileArray = scandir($folder);
5.  foreach($fileArray as $file) {
6.  echo $file.'<br />';
7.  }
8.  ?>
```

== Nach Bearbeitungszeit sortieren ==
Auf Basis dieses Skript lassen sich natürlich weitere Anwendungen programmieren..
Wir können das fileArray auch um ein weiteres Element erweitern, dass wir mit beliebigen Informationen füttern können um danach zu sortieren. Es wird hier bewusst nicht der Schlüssel verwendet, da unterschiedliche Elemente mit dem selben Zeitschlüssel so nicht hätten aufgenommen werden können.

Stattdessen schreiben wir eine eigene Vergleichsfunktion, die wir usort als Parameter übergeben.

Quellcode

```
<?php
require_once 'scandir_fallback.php';
$folder = './';
$fileArray = scandir($folder);
foreach($fileArray as $i => $file) {
$modified = filemtime($folder.$file); //Liefere Unix Zeitstempel
$fileArray[$i] = array($modified, $file);
}
usort($fileArray, create_function('$a, $b', 'if ($a[0] == $b[0]) return 0; else return $a[0]>$b[0]? +1 : -1;'));
foreach($fileArray as $row) {
echo $row[1].'<br />';
}
?>
```

Alles anzeigen

== Rekursiv alle Unterordner auslesen ==

Das letzte Beispiel ist schon ausgereifter. Es handelt sich um eine so genannte rekursive Methode. Die Funktion ruft sich selbst auf und stellt alle Ordner rekursiv dar.

Quellcode

```
<?php
require_once 'scandir_fallback.php';
function ordnerinhalt($folder='.') {
$content = "";
foreach(scandir($folder) as $file) {
if($file[0] != '.') { // Versteckte Dateien nicht anzeigen
if(is_dir($folder.'/'.$file)) {
$folderArray[] = $file;
} else {
$fileArray[] = $file;
}
}
}
// Erst die Ordner ausgeben
if(isset($folderArray)) {
foreach($folderArray as $row) {
$content .= '<b>'.$row.'</b><br />';
$content .= '<div style="padding-left:10px;color:#afafaf" />'; //Unterordner nach Rechts einrücken
$content .= ordnerinhalt($folder.'/'.$row); // rekursive Funktion
$content .= '</div>';
}
}
// ...dann die Dateien ausgeben
if(isset($fileArray)) {
foreach($fileArray as $row) {
$content .= '<a href="'.$folder.'/'.$row.'">'.$row.'</a><br />'; //Dateien verlinken
}
}
// Rekursion ende
return $content;
}
echo ordnerinhalt();
```

```
38.  ?>
```

Alles anzeigen

== Demo ==
Den Code findet ihr einmal sichtbar unter demo.easy-coding.de/php/reading-directory/ und als Zip Datei zum Download.