Westfälische Wilhelms-Universität Münster

Erstellen dynamischer Webseiten mit PHP

WS 2016/2017

wissen.leben
WWU Münster

Dozenten: Patrick Förster, Michael Hasseler

Übersicht der heutigen Inhalte

- Dateien
  - Motivation
  - CSV-Format
  - Dateifunktionen
    - Öffnen/schließen
    - Lesen/schreiben
    - Positionieren
  - Einbinden von Dateien mit require und include
- Nützliche PHP-Funktionen
  - implode
  - explode

Anwendungsbeispiele zur Verwendung von Dateien

- Abspeicherung von Formulardaten
- Dateiinhalte einlesen und von PHP-Skripten weiterverarbeitet werden
- Dateien können als Austauschmedium zu anderen Programmen verwendet werden z. B. csv-Datei für EXCEL

CSV-Dateiformat

- CSV ist die Abkürzung für Comma-separated-values/Character-separated-values
- Datei dient zur Speicherung und dem Austausch einfach strukturierter Daten
- Ein Datensatz wird in einer Dateizeile gespeichert
- Ein Datensatz kann aus mehreren Daten bestehen, die durch ein Trennzeichen z. B. Komma oder Semikolon voneinander getrennt werden
- Beispiel: Max;Mustermann;Musterstadt;01.06.1989
- Optionaler Header in der ersten Zeile benennt die Dateispalten
- Dateien sind erkennbar an der Erweiterung *.csv
- Import für Excel oder Openoffice mit Semikolon als Trennzeichen

Dateifunktion: fopen 1/2

- Datei vorm Zugriff erst öffnen
- Mit der Funktion **fopen**() öffnet man eine Datei
  - Wenn die Datei erfolgreich geöffnet werden konnte, dann erhält man einen Zeiger auf die Datei als Rückgabewert, der als Dateihandler bezeichnet wird. Diesen verwendet man zum weiteren Zugriff auf die Datei.
  - Falls das Öffnen der Datei nicht erfolgreich war, erhält man FALSE.
- Syntax **fopen('Pfad/dateiname', 'Modus')**:
  - 1. Parameter: Öffnet die Datei, die mit Pfad und Dateiname angegeben wird.

Dateifunktion: fopen 2/2

- 2. Parameter: Modus bestimmt, wie die Datei geöffnet werden soll:

'r'	Lesen; setzt den Dateizeiger auf den Dateianfang. (**r=read**)
'r+'	Lesen und Schreiben; setzt den Dateizeiger auf den Dateianfang.
'w'	Schreiben; setzt den Dateizeiger auf den Dateianfang und die Dateilänge auf 0. Wenn die Datei nicht existiert, wird versucht sie anzulegen. (**w=write**)
'w+'	Lesen und Schreiben; setzt den Dateizeiger auf den Dateianfang und die Dateilänge auf 0. Wenn die Datei nicht existiert, wird versucht sie anzulegen.
'a'	Schreiben; setzt den Dateizeiger auf das Dateiende. (**a=append**) Wenn die Datei nicht existiert, wird versucht sie anzulegen.
'a+'	Lesen und Schreiben; setzt den Dateizeiger auf das Dateiende. Wenn die Datei nicht existiert, wird versucht sie anzulegen.

Dateifunktionen: fgets und feof

- Mit der Funktion **fgets** wird der Inhalt einer Datei von der Position des Dateizeigers bis zum Ende der Dateizeile ausgelesen.
  - Syntax: **fgets($dateihandler[, Modus]);**
    - 1. Parameter: $dateihandler ist der Zeiger auf die Datei aus der gelesen werden soll
    - 2. Optionaler Parameter: Mit Modus kann die maximale Anzahl an eingelesenen Zeichen pro Zeile festgelegt werden
- Mit der Funktion **feof** kann ermittelt werden, ob das Ende der Datei zu dem angegebenen Dateihandler erreicht wurde
- Syntax: **feof($dateihandler)**
- Mit den Funktionen fgets und feof in einer while-Schleife als Kombination kann eine zeilenweise komplett eingelesen werden

Dateifunktionen: fread und fwrite

- Mit der Funktion **fread** werden eine bestimmte Anzahl von Bytes aus einer angegeben Datei eingelesen.
- Syntax: **$dateiinhalt = fread($dateihandler, $laenge);**
- Sollen alle Zeichen einer Datei eingelesen werden, so gibt man das 2. Parameter an: **filesize($dateiname)**
- Mit der Funktion **fwrite** wird der Inhalt einer Variablen, die als 2. Parameter übergeben wird, in die im 1. Parameter angegebene Datei geschrieben.
- Syntax: **fwrite($dateihandler, $inhalt);**

Dateifunktionen: fseek, ftell und fclose

- Mit **fseek** Dateizeiger in der Datei positioniert
- Die Funktion **ftell** liefert die aktuelle Position des Dateizeigers
- Wenn eine Datei nicht mehr verwendet wird, dann muss man die Datei schließen, damit die Datei wieder für andere Prozesse zugreifbar ist.
- Syntax: **fclose($dateihandler);**
- Schließen der Datei erfolgreich, dann Rückgabewert TRUE sonst FALSE.

Funktionen: include und require

- Mit den Funktionen **include** und **require** können Dateiinhalte in eine Datei eingebunden werden, wobei als Parameter der einzubindende Dateiname erwartet wird.
- Falls bei **include** die Datei nicht eingebunden werden kann, wird eine Warnung ausgegeben und das php-Skript weiter ausgeführt.
- Falls bei **require** die Datei nicht eingebunden werden kann, wird direkt mit einer Fehlermeldung das php-Skript beendet.
- Mit der Verwendung **include_once** und **require_once** kann ausgeschlossen werden, dass eine Datei mehrmals eingebunden wird. So kann man z. B. eine Mehrfachdeklaration von Funktionen vermeiden.

Vgl. PHP Grundlagen – Erstellung dynamischer Webseiten, RRZN, 9.Aufl., Seite 89/90

Funktionen zu Dateinamenbestandteilen (b1.php)

- **`pathinfo`** liefert Informationen zu dem Dateipfad
- **`basename`** liefert den Dateinamen
- **`dirname`** liefert den Verzeichnispfad

```
<?php
  $filename = 'c:\ordner\name.ext';
  echo '<h3>Namensbestandteile von "'. $filename. '":</h3>';
  $infos = pathinfo($filename);
  echo '<table border="1">';
  echo '<tr><th> key </th><th> value </th></tr>';
  foreach ($infos as $key => $value) {
    echo '<tr>'.'<td>'. $key .':</td>'.'<td>'. $value .'</td>'.'</tr>';
  }
  echo '</table>';
?>
```

Funktionen explode und implode

- Mit der Funktion **explode** kann ein String in Teilstrings zerlegt werden, wobei als erster Parameter das Trennzeichen angegeben wird.
- Mit der Funktion **implode** kann ein Array zu einem String zusammengesetzt werden, wobei als erster Parameter ein Trennzeichen angegeben werden kann.
- Beispiel: (Auszug von b2.php)

```
$nameserver1 = '128.176.0.12';
echo 'Nameserver1 hat die Adresse: ' . $nameserver1 . '<br />';
$array = explode('.', $nameserver1);
$array[3] += 1;
$nameserver2 = implode('.', $array);
echo 'Nameserver2 hat die Adresse: ' . $nameserver2;
```

Zusammenfassung

- Welche Möglichkeiten gibt es eine Datei zu öffnen?
- Was ist ein Dateihandler?
- Wozu werden die Schlüsselwörter **`include`** und **`require`** verwendet?
- Sie können die Funktionen **`explode`** und **`implode`** verwenden?

- **Achtung:** Verwenden Sie die Beispiele und Aufgaben insbesondere bei Dateispeicherungen nicht in einer für alle Welt zugänglichen Testumgebung. Für eine Veröffentlichung fehlt bisher eine Sicherung vor unerlaubten Zugriff und z. B. Bots könnten illegalen Inhalt in die Dateien einfügen.

Aufgaben

1. Ändern Sie beim Beispiel mit den Weihnachtsrezepten die Einfügungsreihenfolge in der Datei und in der Anzeige so, dass das neue Rezept oben eingefügt wird.
2. Fussballergebnisse in einer Datei speichern
   a. Schreiben Sie ein php-Skript mit den Formularfeldern Jahreszahl, Spieltag, Heimmannschaft, Auswärtsmannschaft und Spielergebnis.
   b. Schreiben Sie ein php-Skript speichere_spieltag, dass die Felder mit den Methoden **`is_numeric()`** und **`empty()`** sinnvoll prüft. Die Jahreszahl sollte im Bereich 1800 bis 2100 sein, der Spieltag im Bereich 1 bis 40 und das Spielergebnis vom Format ziffern1:ziffern2. Geben Sie aussagekräftige Meldungen aus und verwenden Sie für bereits gemachte Feldeingaben hidden-fields.

Aufgaben

c. Bei korrekten Daten speichern Sie die Inhalte in spieltag_ergebnisse.csv. Anschließend sollen alle Datensätze am Bildschirm in Tabellenform ähnlich zu den Weihnachtsrezepten ausgegeben werden, wobei das Semikolon als Trennzeichen nur in der Datei zu sehen sein soll. (Tipp: **`explode`** und **`implode`** verwenden)

Anhang: ZIVinteraktiv-Abstimmung

- QR-Code zur anonymen Abstimmung:

- https://www.uni-muenster.de/ZIV/anw/zivinteraktiv/abstimmung.php?code=5804a66a