Lektion 9: Zähler und Gästebuch, Arbeit mit Dateien

Wolltest du schon immer mal selber einen Zähler in deine Seiten einbinden? Oder ein endlich werbefreies, ganz individuelles Gästebuch?

Kein Problem, in diesem Kapitel zeige ich dir, wie's gemacht wird! Stricke deinen eigenen „Text-Counter" mit PHP.

> Bei dieser Gelegenheit führe ich dir das „Lesen aus" und „Schreiben in Dateien" vor.

Alle Skripte dieses Kapitels habe ich bewusst nach dem Motto „Supersimpel" gestrickt, um dir das Prinzip zu verdeutlichen. Die Beispiele enthalten daher wenig Möglichkeiten zum Fehlerabfangen, aber sie funktionieren!

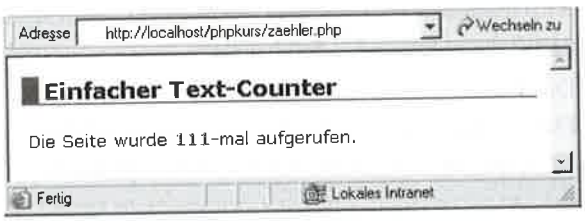
Außerdem sehen wir wohlwollend über die Tatsache hinweg, dass Zähler alles andere als zuverlässig sind. Sie zählen lediglich die Hits, also auch jeden durch Klick auf den „Reload"-Button erzeugten Besuch. (Das lösen wir später.)

Miesmachen gilt nicht, wir skripten einfach los.

Richte einen Zähler ein!

Zuerst legst du dir eine einfache Textdatei an, die wir im Beispiel `zaehler.txt` nennen. Notiere hier eine „1" (oder welche Zahl auch immer), speichere deine `zaehler.txt` und lege diese im Beispiel in das gleiche Verzeichnis wie die noch zu erstellende PHP-Datei.

> Achte darauf, dass du diese Datei beim Dienstleister zum Schreiben freigeben musst. Mehr darüber auf Seite 77.



Ein einfacher Zähler ist mit PHP schnell realisiert

Und nun zur Datei, in der gezählt wird! In PHP ist dabei alles so wie im „richtigen" Leben. Zuerst wird die Datei geöffnet, dann erst kannst du sie auslesen und bearbeiten!

Mit Auslesen ist das Anzeigen der alten Daten gemeint. Mit Einlesen überschreibst du in diesem Beispiel alles mit neuen Werten.

Quelltext der zaehler.php

Hier zeige ich dir zuerst den erfreulich kurzen PHP-Teil des Dokuments. Kleide das Skript in eine Seite namens `zaehler.php` ein.

```
<p>Die Seite wurde
<b><?php
$zeiger = fopen("zaehler.txt", "r+");
$zaehler = fgets($zeiger, 7);
echo $zaehler;
$zaehler++;
rewind($zeiger);
fputs($zeiger, $zaehler);
fclose($zeiger);
?></b>-mal aufgerufen.</p>
```

Schauen wir uns nun genau an, was hinter den einzelnen Funktionen steckt!

Datei öffnen mit fopen()

Zum Öffnen einer Datei benötigst du die Funktion `fopen()`. Diese Funktion besitzt folgende Syntax:

```
$zeiger = fopen(Dateiname, Modus)
```

■ Dateiname

Der Dateiname ist im Beispiel kein großes Rätsel. Wir schreiben an diese Stelle den String (die Zeichenfolge) `zaehler.txt`.

■ Modus

Der Modus ist nun das, was du mit der Datei anstellen kannst.

Du kannst eine Datei im *Nur-Lesemodus* (r wie read only), im *Lese- und Schreibmodus* (r+ wie read + write), im *Nur-Schreibmodus* (w wie write only), im *Anhänge-Modus* (a wie append) usw. öffnen.

Da wir Lesen und Schreiben wollen, verwenden wir den r+-Modus für read and write.

> Wichtiger Hinweis: Das Lesen und Schreiben beginnt bei diesem Modus am Dateianfang.

Der geheimnisvolle Dateizeiger

Doch was verbirgt sich hinter der Variablen `$zeiger`? Was hat es mit diesem geheimnisvollen Dateizeiger auf sich?

Nun, die Funktion `fopen()` positioniert stets einen „Dateizeiger" in der Datei. Ist die Datei einmal geöffnet, arbeitet man mit diesem Zeiger weiter. Dazu wird der Status des Zeigers am Anfang in einer Variablen gesichert.

> Ich habe mir für den Dateizeiger im Beispiel die Variable `$zeiger` ausgedacht.

Wichtig zu wissen: Beim im Beispiel verwendeten r+-Modus zeigt der Dateizeiger auf den Anfang der Datei, auf das erste Zeichen.

> Schwierigkeiten mit dem Verständnis? Stelle es dir einfach so vor, dass dein „unsichtbarer Cursor" momentan am Anfang der geöffneten Textdatei steht und auf „Anweisungen" wartet!

Die Funktion fgets()

Als nächstes schauen wir uns die Funktion `fgets()` an. Sie liest Daten aus einer Datei. Der Lesevorgang endet, wenn die Zeile bzw. Datei zu Ende ist oder wenn die maximale Zahl an Bytes erreicht ist.

Die Syntax sieht folgendermaßen aus:

```
fgets(Zeiger, Byte);
```

Da wir eine 7 als Byte-Wert angegeben haben, berücksichtigt die Funktion immerhin eine Zahl mit der entsprechenden Länge. Ist die Zahl zu lang, wird der Rest beim Auslesen ignoriert. (Ggf. passt du den Wert einfach an.)

Aber die Gefahr ist momentan nicht groß, denn derzeit steht noch eine „1" in der Datei `zaehler.txt`. Diese wird ausgelesen und in der Variablen `$zaehler` abgelegt.

In der nächsten Zeile geben wir diese Variable aus und erhöhen sie um eins:

```
echo $zaehler;
$zaehler++;
```

Hast du Phantasie? Dann stelle dir vor, dass dein unsichtbarer „Cursor" jetzt zum Ende der Zeile „gehuscht" ist (bzw. an der 7-Byte-Grenze halt gemacht hat).

rewind() und fputs()

Zuerst spulst du nun zurück! Die Funktion `rewind()` setzt den „Cursor" wieder auf den Anfang der Zeile zurück. Dafür sorgt die Zeile

```
rewind($zeiger);
```

Die Funktion

```
fputs($zeiger,$zaehler);
```

wiederum ist dafür verantwortlich, dass der aus der Variablen `$zaehler` stammende String in die Datei geschrieben wird.

Da wir uns im Lese-Schreib-Modus befinden (r+) und wieder am Anfang der Datei stehen, wird der alte Wert (am Anfang „1") nun mit „2" überschrieben.

Datei schließen mit fclose()

Ordentlich wie wir sind, schließen wir die Datei mit `fclose($zeiger)`. Es funktioniert zwar auch ohne diese Maßnahme, doch das schont Systemressourcen!

> Eine offene Datei sollte nach Verwendung stets mit `fclose()` wieder geschlossen werden!

Ein einfaches Gästebuch

Mit diesem Wissen kannst du dir im Prinzip auch ein einfaches Gästebuch „stricken". Das folgende (sehr einfache) Skript ist ein Funktionsmuster. Ich würde es aus Spamschutzgründen allerdings nicht (mehr) für den Einsatz auf einer öffentlich zugänglichen Website empfehlen! Leider!

Die Einträge werden mit „Datumsstempel" versehen

Egal, probieren wir es aus: Das Gästebuch sichert Kommentar, Namen und E-Mail-Adresse. Außerdem sollen neue Einträge zuoberst stehen.

Quelltext der Datei guestbook.php (Gästebuch)

Hier der komplette Quelltext des Skripts im Überblick. Alle wichtigen Aktionen habe ich kommentiert. Schaue in den Ordner `lektion9` und vergleiche mit der Musterdatei `guestbook.php`.

```
<html>
<head>
  <title>Einfaches G&auml;stebuch</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="../css/neu.css">
</head>
<body>
<h1>Einfaches Gästebuch</h1>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">Ihr Kommentar:<br>
<textarea cols="55" rows="4" name="comment"></textarea><br>
Ihr Name:<br>
<input type="text" name="name"><br>
Ihre E-Mail-Adresse:<br>
<input type="text" name="email"><input type="submit" value="Ver&ouml;ffentlichen"></form>
<h3>Bisherige Meinungen</h3>
<?php
// Dateiname in Variable speichern
$datei = "comment.txt";
// Variable Kommentar gesetzt? Name und E-Mail nicht leer?
if (isset($_POST['comment']) && $_POST['name'] != "" && $_POST['email'] != "") {
  $comment = $_POST['comment'];
  $name = $_POST['name'];
  $email = $_POST['email'];
  // Datei wird zum Schreiben-Lesen geöffnet
  $zeiger = fopen($datei, "r+");
  // alte Daten herauslesen und in $alt sichern
  $alt = fread($zeiger, filesize($datei));
  // E-Mail-Link entsteht
  $email = "<a href=\"mailto:$email\">$email</a>";
  // Datum ermitteln und formatieren
  $datum = date("j.n.Y");
  // HTML-Zeichen maskieren, Slashes entfernen, Zeilenumbrüche erhalten
  $comment = htmlspecialchars($comment);
  $comment = stripslashes(nl2br($comment));
  // Meinung "zusammensetzen"
  $meinung="<p><b>$name</b> ($email) schrieb am <i>$datum</i>:<br>$comment</p>\n";
  // unsichtbarer Cursor marschiert zum Anfang
  rewind($zeiger);
  // neue Meinung vor alte in Datei schreiben:
  fputs($zeiger, "$meinung \n $alt");
  // Datei schließen
  fclose($zeiger);
}
// Datei komplett anzeigen
readfile($datei);
?>
</body>
</html>
```

Datei comment.txt anlegen

Wie schon beim Zähler heißt auch hier die Voraussetzung: Lege zur Speicherung der Daten eine Textdatei an (die *nicht schreibgeschützt** ist). Nenne diese `comment.txt` und packe sie in den gleichen Ordner. Tippe ein Leerzeichen ein, damit schon Inhalt vorhanden ist und `fread()` keine Fehlermeldung erzeugt. (*siehe Anhang auf Seite 77!)

So funktioniert das Skript

Nun zum Skript: Aus Bequemlichkeit speichere ich den Namen der Datei von vornherein in der Variablen `$datei`. Dann teste ich, ob die aus dem HTML-Formular übergebenen Variablen gesetzt (`$_POST['comment']`) bzw. nicht leer sind (`$_POST['name']` bzw. `$_POST['email']`).

(Wir könnten natürlich einen weit aufwändigeren Test einbauen, im Beispiel soll es das jedoch vorerst gewesen sein.)

Als Nächstes wird die Datei wieder zum Lesen und Schreiben geöffnet.

Die Funktion fread()

Neu ist hierbei die Funktion `fread()`. Diese liest im Gegensatz zur nur zeilenweise arbeitenden Funktion `fgets()` den Inhalt der *gesamten* Datei aus. Ansonsten gleicht die Syntax der von `fgets()`. Auch bei `fread()` werden als Argumente der Dateizeiger und die Dateigröße übergeben.

```
fread(Zeiger, Byte)
```

Funktion filesize()

Da ich mich bei der Größe der Datei jedoch nicht von vornherein einschränken möchte, ermittle ich diese mit der Funktion `filesize()`.

Die Syntax von `filesize()` geht wie folgt:

```
filesize(Dateiname)
```

Insgesamt sieht die Zeile folgendermaßen aus:

```
$alt=fread($zeiger,filesize($datei));
```

Was hat es mit der Variablen `$alt` auf sich? Ich sichere den bisherigen „alten" Inhalt der Textdatei und lege diesen in der Variablen `$alt` ab.

E-Mail-Link zusammensetzen

Als nächstes setze ich den HTML-E-Mail-Link zusammen und speichere diesen String in der Variablen `$email`.

Dazu greife ich auf die schon bekannten Backslashs zum Maskieren der Gänsefüßchen zurück. Außerdem „wirkt" das aus HTML bekannte Schlüsselwort `mailto:`.

Datum ausgeben mit date()

Des Weiteren soll zusätzlich ein „Datumsstempel" zu jedem Eintrag „erzeugt" werden. Die am Anfang schon erwähnte Funktion `date()` gibt das formatierte Datum aus. In der von mir gewählten Syntax ist es das aktuelle Datum.

Die Schalter j, n und Y geben den Tag und Monat zweistellig und das Jahr vierstellig aus.

Eine Übersicht über die wichtigsten Schalter von `date()` findest du auf der nächsten Seite.

Im Beispiel wird dieser formatierte String in der Variablen `$datum` abgelegt:

```
$datum = date("j.n.Y");
```

stripslashes() und htmlspecialchars()

Die nächsten Zeilen kennst du tlw. schon von den vorherigen Skripten her.

Die Funktion `htmlspecialchars()`sorgt wiederum dafür, dass HTML-Codezeichen maskiert werden. Denn sonst könnte ein „böswilliger" Besucher HTML- oder JavaScript-Code in deine Datei schreiben oder einen eigenen CSS-Link einbauen und dein Layout komplett verändern! Mit `stripslashes()` entferne ich die Anzeige von Backslashs in der Ausgabe, mit `nl2br()` sichere ich, dass vom Benutzer eingegebene Umbrüche auch als solche dargestellt werden.

Bei `fputs()` sorge ich zusätzlich dafür, dass neue Kommentare *vor* den alten Dateiinhalt geschrieben werden. Denn schließlich soll eine neue Meinung stets zuoberst ausgegeben werden:

```
fputs($zeiger, "$meinung \n $alt");
```

Die Funktion readfile()

Doch was steckt hinter `readfile($datei)`? Die geniale Funktion `readfile()` liest den gesamten Inhalt der Datei aus und schickt die Ausgabe an den Browser. Das Gästebuch ist einfach zu pflegen: Wenn dir ein Eintrag nicht passt, öffnest du einfach die Textdatei und löschst die jeweilige Passage.

Schalter der Funktion date()

Zurück zur Funktion `date()`. Damit kann man nicht nur das Datum inklusive Wochentag, sondern auch die aktuelle Uhrzeit ausgeben.

Erstaunliche Formatiermöglichkeiten für das Datum!

Hier eine Kurzübersicht der wichtigsten Schalter von `date()`:

Datumsschalter

Folgende Schalter „kümmern sich" um die Anzeige des Datums.

Schalter	Erklärung	Beispiel
j	Monatstag ohne führende Null	1 bis 31
d	Monatstag mit führender Null	01 bis 31
n	Monat ohne führende Null	1 bis 12
m	Monat mit führender Null	01 bis 12
S	englische Aufzählung (st, nd, rd, th)	1st 3rd 5th
y	Jahreszahl mit zwei Stellen	02
Y	Jahreszahl mit vier Stellen	2005
D	ausgeschr. Wochentag, kurz	Mon
l (kl. „L")	ausgeschr. Wochentag, lang	Monday
M	ausgeschr. Monatsname, kurz	Jan
F	ausgeschr. Monatsname, lang	January
w	Wochentag (0 = Sonntag)	0 bis 6
z	Tag des Jahres	0 bis 365
t	Anzahl der Monatstage	28 bis 31

Du möchtest das Datum im Format *1.7.06* ausgeben? Dann schreibst du beispielsweise:

```
echo date("j.n.y");
```

Du wünschst z.B. folgende englische Schreibweise: *3rd of July 2006*? Dann schreibst du:

```
echo date("jS of F Y");
```

Kommt es dir dagegen nur auf den ausgeschriebenen Wochentag in der Langform an, genügt

```
echo date("l");
```

Wie du siehst kannst du zum Formatieren mit Leerräumen und *in Grenzen* selbst mit anderen Zeichen arbeiten! Setze diese einfach an der gewünschten Stelle innerhalb von date("") ein.

Möchtest du angeben, wie viele Tage ein Monat hat, schreibst du beispielsweise:

```
echo "Der Monat hat " . date("t") . " Tage!";
```

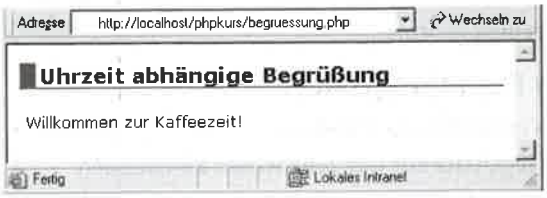
Uhrzeitsschalter

Natürlich stehen dir auch etliche Schalter zur Angabe der Uhrzeit zur Verfügung

Schalter	Erklärung	Beispiel
a	am oder pm	am
A	AM oder PM	AM
h	Stunden im 12-h-Format	1 bis 12
H	Stunden im 24-h-Format	1 bis 24
i	Minuten von 00 bis 59	04
s	Sekunden von 00 bis 59	12

Folgende Zeile gibt den Zeitpunkt des Aufrufs einer Seite aus: `echo date("H:i:s");`

Das nächste Skript begrüßt dich in Abhängigkeit von der Uhrzeit mit einem anderen Spruch.



Je nach Uhrzeit wird der Besucher anders begrüßt

Vergleiche mit der Datei `begruessung.php`:

```
<?php
if (date("H") <= 11) {
  echo "Guten Morgen!";
} elseif (date("H") <= 14) {
  echo "Mahlzeit!";
} elseif (date("H") <= 17) {
  echo "Willkommen zur Kaffeezeit!";
} else {
  echo "Guten Abend!";
}
?>
```

Frage: Wären in PHP eigentlich Ticker mit sich ständig aktualisierender Zeitangabe sinnvoll? Nein! Antwort: Da das Skript z.B. im Sekundentakt aufgerufen werden müsste, wäre die Last für den Server zu groß.

Solche Aufgaben sind ideal für JavaScript, siehe „Homepages für Fortgeschrittene", Plus 12.