

Arbeiten mit Textdateien

Arbeiten mit Textdateien	1
Das Prinzip des Dateizeigers	1
1. Dateizeigeroperationen	1
Dateien auslesen	2
1. Die gesamte Datei auslesen	2
2. Die verschiedenen Möglichkeiten Dateien zu öffnen	2
3. Eine Datei zeilenweise auslesen	3
4. Dateien zeichenweise auslesen	4
In Dateien schreiben	5
Anmerkungen	6

Oft ist es erforderlich Daten in einfachen Dateien abzulegen. PHP stellt uns dafür eine Flut von Funktionen zur Verfügung. In diesem Kurs werden wir uns mit gängigen Methoden zum Schreiben und Auslesen von Dateien näher beschäftigen.

Die folgende Tabelle stellt eine Übersicht über die wichtigsten Funktionen dar:

Funktionsname	Erläuterung
fclose()	Schließt eine Datei bzw. den Dateizeiger
feof()	Prüft, ob der Dateizeiger am Ende der Datei steht
fgetc()	Liest das Zeichen, auf welches der Dateizeiger zeigt
fgets()	Liest eine Zeile von der Position des Dateizeigers aus
fgetss()	Liest eine Zeile von der aktuellen Position des Dateizeigers aus ohne Berücksichtigung von PHP- und HTML-Tags
file()	Liest eine Datei komplett in ein Array
file_exists()	Überprüft, ob eine Datei existiert
filesize()	Gibt die Größe einer Datei aus
fopen()	Öffnet eine Datei, erzeugt den Dateizeiger
fputs()	Schreibt Daten an die Position des Dateizeigers
fread()	Liest Binär-Dateien aus
fseek()	Verschiebt den Dateizeiger an eine beliebige Position
ftell()	Ermittelt die aktuelle Position des Dateizeigers
fwrite()	Schreibt Binärdaten in eine Datei
is_writable()	TRUE, wenn Schreibrechte zur Verfügung stehen
readfile()	Gibt den Inhalt einer Datei aus
rewind()	Setzt den Dateizeiger an den Anfang der Datei

Das Prinzip des Dateizeigers

Den Dateizeiger kann man im wesentlichen mit dem Cursor in einem Textfeld vergleichen. Der Dateizeiger markiert ein Byte, also ein Zeichen bzw. eine beliebige Position in einer Textdatei. Der Dateizeiger ermöglicht es uns somit bestimmte Zeichen oder auch ganze Dateien auszulesen bzw. an eine bestimmte Position in einer Datei etwas zu schreiben.

1. Dateizeigeroperationen

Gelegentlich ist es erforderlich bestimmte Zeichen aus Dateien auszulesen und dann ist es oft erforderlich die genaue Position des Dateizeigers zu ermitteln oder zu verändern.

Die Funktion **ftell()** gibt uns die aktuelle Position des Dateizeigers aus. Mit den Funktionen **rewind()** und **fseek()** können wir die Position verändern.

Der Funktion **ftell()** muss zur Bestimmung der Dateizeigerposition lediglich der Dateizeiger (näheres dazu später) als Parameter übergeben werden (gezählt wird vom Dateianfang aus). Die

Funktion **rewind()** setzt den Dateizeiger auf das erste Zeichen der Datei. Als Parameter muss hier ebenfalls nur der Dateizeiger angegeben werden.

Der Funktion **fseek()** muss außerdem noch als zweiter Parameter die Anzahl der Zeichen übergeben werden, um die Dateizeiger vom Dateianfang aus verschoben werden soll.

Schauen wir uns einfach mal folgendes Beispiel an:

```
1 <?php
2 $Datei = fopen("test.txt", "r"); //Öffnet die Datei, erzeugt einen Dateizeiger
3
4 fseek($Datei, 10);
5 echo ftell($Datei);
6 fclose($Datei);
7 ?>
```

Ergebnis: 10

Der Dateizeiger wird mit Hilfe der Funktion **fseek()** auf das zehnte Zeichen der Datei verschoben. Die Funktion **ftell()** liefert seine aktuelle Position.

Dateien auslesen

Zum auslesen von Dateien stehen uns mehrere Funktionen zur Verfügung. Einige werden in diesem Abschnitt näher erläutert.

1. Die gesamte Datei auslesen

Die Funktionen **readfile()** und **file()** stehen uns zu diesem Zweck zur Verfügung. Mit **readfile()** können wir direkt den Inhalt einer Datei auslesen, während uns **file()** den Inhalt in einem Array ablegt.

Als erstes legen wir am besten eine einfache Textdatei für Testzwecke an. In diesem Kurs halte ich mich an folgendes Beispiel:

```
1 Dieser Kurs beschäftigt sich mit dem Bearbeiten von Dateien
2 unter PHP.
3 Dateien ermöglichen es mir Einstellungen ohne Zugriff
4 auf eine Datenbank zu speichern.
```

Beiden Funktionen muss nur der Pfad der Datei als Parameter übergeben werden.

```
1 <?php
2 echo readfile("test.txt");
3 ?>
1 <?php
2 echo implode("", file("test.txt"));
3 ?>
```

Das Ergebnis der Funktion **file()** wird in diesem Fall mit Hilfe von **implode()** ausgegeben. Die Funktion **implode()** wandelt in diesem Beispiel das Array mit dem Dateiinhalt in eine einfache Zeichenkette um.

2. Die verschiedenen Möglichkeiten Dateien zu öffnen

Viele Funktionen die der Verarbeitung von Dateien dienen, setzen voraus, dass die Datei geöffnet ist. Zum Öffnen von Dateien können wir uns der Funktion **fopen()** bedienen. Dieser Funktion müssen 2 Parameter übergeben werden:

```
$Datei = fopen(Datei, Öffnungsmodus);
```

Im ersten Parameter muss also der Pfad zur Datei angegeben werden. Der zweite Parameter erfordert hingegen ein wenig mehr Erklärung.

Folgende Modi stehen uns zur Verfügung:

Modus	Bedeutung/Funktion
r	-nur lesen, Beginn am Dateianfang
r+	-lesen und schreiben, Beginn am Dateianfang
w	-nur schreiben, bisheriger Inhalt wird gelöscht, nichtvorhandene Dateien werden erzeugt
w+	-lesen und schreiben, bisheriger Inhalt wird beim Schreiben gelöscht, nichtvorhandene Dateien werden erzeugt
a	-nur schreiben, Text wird an Datei angehängen, nichtvorhandene Dateien werden erzeugt
a+	-lesen und schreiben, Text wird an Datei angehängen, nichtvorhandene Dateien werden erzeugt

3. Eine Datei zeilenweise auslesen

In jedem Fall müssen wir die Datei zuerst öffnen. Für das zeilenweise Auslesen von Dateien sollten wir den Modus **r** verwenden.

Wir rufen also zuerst die Funktion **fopen()** auf. Als Rückgabewert erhalten wir einen sogenannten **Dateizeiger**. Existiert die zu öffnende Datei nicht, so liefert uns die Funktion **TRUE** zurück.

Zum Auslesen einer Zeile steht uns, wie der Tabelle oben zu entnehmen ist, die Funktion **fgets()** zur Verfügung. Der Funktion müssen 2 Parameter übergeben werden:

```
$Zeile = fgets(Dateizeiger, Anzahl_Zeichen);
```

Der erste Parameter besteht aus dem mit **fopen()** erzeugtem Dateizeiger, während mit dem zweiten Parameter die Anzahl der zu lesenden Zeichen pro Zeile angegeben werden.

Hinweis:

Versuchen Sie diesen Wert recht klein zu halten, um den Server nicht übermäßig auszulasten. Auf einigen Servern führen zu große Werte zu Problemen während des Ablauf des Scriptes. Bei zu kleinen Werten werden nicht alle Zeichen einer Datei korrekt ausgelesen.

Abschließend sollte die Datei wieder geschlossen werden mit Hilfe der Funktion **fclose()**. Dieser Funktion muss lediglich der Dateizeiger als Parameter übergeben werden.

Folgendes Beispiel liest die ersten 100 Zeichen der ersten Zeile unserer Datei aus:

```
1 <?php
2 $Datei = fopen("test.txt", "r");
3 $Zeile = fgets($Datei, 100);
4 fclose($Datei);
5
6 echo $Zeile;
7 ?>
```

Ergebnis: Dieser Kurs beschäftigt sich mit dem Bearbeiten von Dateien

Auf diese Art und Weise lassen sich selbstverständlich auch ganze Dateien auslesen:

```
1 <?php
2 $Datei = fopen("test.txt", "r");
3 if($Datei)
4 {
```

```

5  while(!feof($Datei))
6  {
7      $Zeile = fgets($Datei, 100);
8      echo $Zeile . "<br>";
9  }
10  fclose($Datei);
11  } else echo "Die angegebene Datei konnte nicht gefunden werden!";
12  ?>

```

Der Dateizeiger wird mit jedem Lesevorgang (jedem Aufruf von **fgets()**) um eine Zeile in der Datei vorangeschoben. Mit Hilfe der IF-Abfrage wird überprüft, ob die Datei überhaupt existiert. Die Datei wird nach erfolgreicher Prüfung solange zeilenweise ausgelesen, bis das Dateiende erreicht ist.

Die Funktion **feof()** liefert *TRUE* zurück, sobald das Dateiende erreicht ist. Als Parameter muss der entsprechende Dateizeiger übergeben werden.

4. Dateien zeichenweise auslesen

Die Vorgehensweise beim zeichenweisen Auslesen entspricht im wesentlichen der beim zeilenweisen Auslesen von Dateien. Lediglich der Name der Funktion lautet anders, in diesem Fall **fgetc()**. Dieser Funktion muss allerdings nicht die Anzahl der auszulesenden Zeichen als zweiter Parameter übergeben werden. Es wird immer das Zeichen gelesen, auf dem sich der Dateizeiger momentan befindet.

```

1  <?php
2  $Datei = fopen("test.txt", "r");
3  if($Datei)
4  {
5      while(!feof($Datei))
6      {
7          $Zeichen = fgetc($Datei);
8          echo $Zeichen;
9      }
10  fclose($Datei);
11  } else echo "Die angegebene Datei konnte nicht gefunden werden!";
12  ?>

```

Ergebnis: Dieser Kurs beschäftigt sich mit dem Bearbeiten von Dateien unter PHP. Dateien ermöglichen es mir Einstellungen ohne Zugriff auf eine Datenbank zu speichern. (ohne Zeilenumbrüche)

Die Zeilenumbrüche werden in den HTML-Code übernommen, im Browser jedoch nicht als solche ausgegeben. Sollten Sie auf die Zeilenumbrüche nicht verzichten wollen, so empfiehlt es sich die Umbrüche durch `
`-Tags zu ersetzen.

```

1  <?php
2  ...
3      $Zeichen = fgetc($Datei);
4      $Zeichen = str_replace("\n", "<br>\n", $Zeichen);
5      echo $Zeichen;
6  ...
7  ?>

```

Die Zeilenumbrüche werden nun korrekt ausgegeben. Im wesentlichen haben wir nur `\n` (ein Zeilenumbruch im HTML-Code) durch einen `
`-Tag mit einem anschliessenden Zeilenumbruch im HTML-Code in der Variable **\$Zeichen** ersetzt.

Da die Funktion **fgetc()** lediglich das Zeichen ausliest, auf dem sich momentan der Dateizeiger befindet, können wir auch jedes beliebiges Zeichen in der Datei auslesen:

```
1 <?php
2 $Datei = fopen("test.txt", "r");
3 fseek($Datei, 10);
4 echo fgetc($Datei);
5 fclose($Datei);
6 ?>
```

Ergebnis: s

Dieser Code gibt das elfte Zeichen der Textdatei aus, **da die Zählung mit 0 beginnt**.

In Dateien schreiben

Wir haben uns bisher recht ausführlich mit dem Lesen aus Dateien befasst und wollen uns nun der Thematik des Schreibens zuwenden. Die Vorgehensweise entspricht hier im wesentlichen der des Schreibens. Die Datei muss zuerst geöffnet und anschließend auch wieder geschlossen werden.

Beim Schreiben ist die Wahl des Modi umso wichtiger, deshalb sollte man vorher genau überlegen, was man eigentlich erreichen möchte. Zum Platzieren von Text in einer Datei steht uns die Funktion **fputs()** zur Verfügung. Als erster Parameter muss dieser Funktion wieder der Dateizeiger übergeben werden, während im zweiten Parameter der zu speichernde Text notiert werden sollte.

Hinweise:

Als Text können selbstverständlich vorformatierte Zeichenketten gespeichert werden, nähere Informationen erhalten Sie im offiziellen PHP-Handbuch unter: <http://www.php.net>.
Einen Zeilenumbruch erzeugen Sie mit der Zeichenfolge "\n".

```
1 <?php
2 ...
3 $Zeile = $Zeile . "\n";
4 ...
5 ?>
```

Spätestens das folgende Beispiel sollte den Umgang mit der Funktion **fputs()** verdeutlichen:

```
1 <?php
2 //Text in Datei schreiben, bisheriger Inhalt wird gelöscht
3 $Text = "Das ist unsere erste Zeile!\nUnd das ist die Zweite.";
4
5 $Datei = fopen("test.txt", "w");
6 fputs($Datei, $Text);
7 fclose($Datei);
8
9
10 //Text anhängen
11 $Text = "\nDiese Zeile wurde angehängt.";
12
13 $Datei = fopen("test.txt", "a+");
14 fputs($Datei, $Text);
15 fclose($Datei);
16 ?>
```

Ich hoffe, dass ich Ihnen mit diesem Kurs die Bearbeitung von Text-Dateien ausreichend vermitteln konnte. Die in diesem Kurs gewählten Beispiele lassen sich selbstverständlich abändern und erweitern. Ich persönlich empfehle es Ihnen sich in diesem Zusammenhang näher mit der Formatierung und Bearbeitung von Zeichenketten zu beschäftigen.

Anmerkungen

Die Verbreitung oder die kommerzielle Vermarktung dieses Kurses ist ohne das Einverständnis des Autors nicht gestattet! Sollten Sie Fragen haben, so kontaktieren Sie mich bitte per Mail oder Verfassen Sie einen Beitrag im Forum unter <http://www.coder-treff.de>.

Autor: Stephan Altmann

Mail: webmaster@programmers-club.de

Homepage: <http://www.programmers-club.de>