PHP - Einführung

Stefan Maihack Dipl. Ing. (FH)
Datum: 19.05.2013

Inhaltsverzeichnis

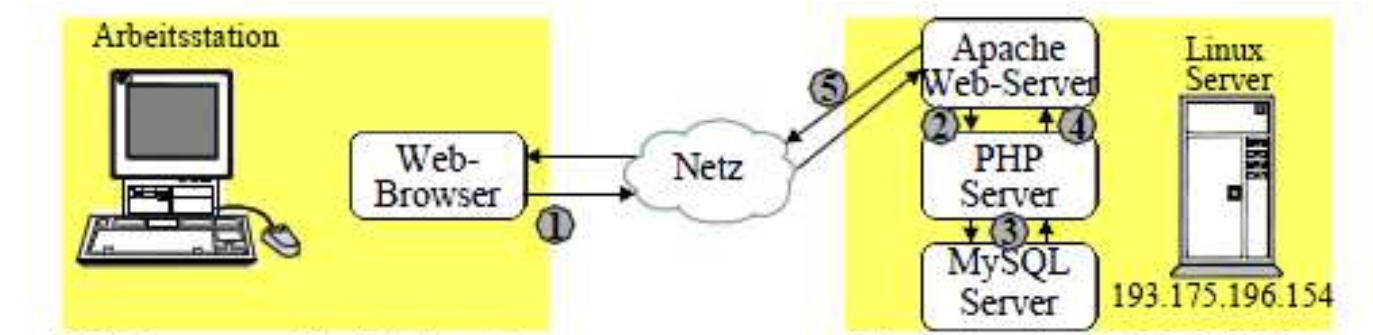
Informationen - Dokumentationen

- HRZ-Dokumentation PHP-HandbuchSehr umfangreiches Referenz-Manual in 10 Sprachen
- SELFPHPSehr umfangreiche Funktionsreferenz mit vielen Beispielen
- WeaverslavePHP-Editor
- Zend Technologies"where PHP meets eBusiness„
- phpWizardWeb Application Development with PHP (Tobias Ratschiller)
- phpCenter.deDeutschsprachiges PHP-PortalphpWelt.deDeutschsprachiges PHP-Portal
- PHP-Homepage.deDeutschsprachiges PHP-Portal
- Bücher RRZN-Handbuchim HRZ erhältlich
- Dynamische Webseiten-GenerierungVorlesung mit Übungen von (A. Wassermann)
- PHP SchulungTutorium (Ulf Wendel, Johann-Peter Hartmann)
- PHP-Kochrezepte für typische AnwendungenBeispielsammlung (Bernd Cappel, Universität Düsseldorf)
- PHP-Usergroups PEAR, Vortrag dazuPHP Extension and Application Repository

„LAMP“ - Konfiguration

Prinzipieller Ablauf:

1) Nutzer ruft mit Browser eine Web-Seite auf dem Apache-Servern (Webserver) auf.
2) Apache-Server übergibt Webseite an PHP-Server
3) PHP-Server liest Daten aus der MySQL-Datenbank
4) PHP-Server verpackt die Daten im HTML-Format (für den Browser)
5) Apache-Server übergibt die Web-Seite an den Browser



Vom Web-Browser zum Web-Server

- Web-Browser fordert Web-Seite vom Web-Server an
  →Übertragungsprotokoll: HTTP (Hypertext Transfer Protocol)

Dienst | IP-Adresse des Webservers | Dateiname

Beispiel: http://192.174.196.154/index.html
http://192.174.196.154/index.php

Die Dateiendung gibt den Dokumententyp an:

.html = Hypertext Markup Language (Seitenbeschreibungssprache)
.php = Hypertext Preprozessor (Seite mit PHP-Befehlen)

PHP in HTML einbetten

- Prinzip:
  PHP ist eine Scriptsprache (Quellcode, nicht übersetzt)
  → Datei ist eine Textdatei (ASCII-Code, mit Texteditor lesbar)

  PHP ist von der Programmiersprache C abgeleitet.

  PHP-Kommandos sind in HTML-Code eingebettet
  → als Kommentare, daher in Browserdarstellung nicht sichtbar

- PHP-Kommandos haben eine Start- und Endmarke. Hierzu gibt es mehrere Möglichkeiten:

```
<SCRIPT LANGUAGE=PHP> echo "Guten Tag!" </SCRIPT>
<?PHP echo "Guten Abend!" ?>
<!-- empfohlen! -->
<? echo "Gute Nacht!" ?>
<?php echo „Guten Morgen!“ ?>
```

Regel: Die HTML-Bestandteile bleiben unangetastet.
Die PHP-Teile werden verarbeitet und müssen zu korrektem HTML-Code führen.

Vom PHP-Server zum Web-Browser

- Prinzip:
  PHP-Server erzeugt HTML-Kommandos
  → Diese werden in eine Textdatei geschrieben
  Datei mit HTML-Code wird an Web-Server übergeben
  → Web-Server schickt diese zum Web-Browser
  Web-Browser interpretiert HTML-Code = stellt Webseite dar

- Beispiel PHP-Script:

  PHP Ausgabebefehl

  ```
  <?php echo „<center> Das soll ausgegeben werden </center>“; ?>
  ```

  erzeugt den HTML-Code:

  HTML-Zentrierbefehl

  ```
  <center> Das soll ausgegeben werden </center>
  ```

  Der Browser macht daraus einen zentrierten Text

  Das soll ausgegeben werden

HTML-Grundkurs

- PHP-Code kann mit HTML-Code gemischt werden.
  Prinzipieller Aufbau einer Webseite in HTML

```
<html>
  <head>
        <title> Seitenüberschrift </titel>
  </head>
  <body>
        Hier steht der Text des Seiteninhalts
  </body>
</html>
```

- Kennzeichnend sind Start- und Endemarken (deswegen „Markuplanguage“)
- Weitere Marken sind:
   zentrieren
  <h1> Überschrift </h1>
   Schriftart
   Hyperlink

 Neue Zeile </br> usw.

PHP in HTML einbetten

- PHP-Code durch Start- und Endemarke kennzeichnen
  Beispiel: eine zentrierte Überschrift erzeugen:

  <h1><?php printf(„Überschrift“); ?></h1>

  oder:

  <?php printf(„ <h1>Überschrift</h1> “); ?>

Variablen in PHP

- **Variable in PHP**

Variablen-Namen beginnen immer mit einem Dollarzeichen:

```
$pi = 3.1415982;
$Meldung = "Bitte geben Sie Ihre Geheimzahl ein!";
$auch_eine_Textkonstante='Die Geheimzahl ist falsch,
```

In Variablen-Namen wird Groß-Klein-Schreibung unterschieden!

→ PHP-Variablen werden nicht deklariert. Der Typ wird automatisch erzeugt und ggf. angepasst.

→ Bei der Ausgabe von Texten mit eingebetteten Variablen-Namen
werden diese durch ihre Werte ersetzt:

```
echo "Die Konstante pi hat den Wert $pi";
```

→ Das gilt nur, wenn die Texte in " eingefasst sind, nicht aber wenn sie in ' eingefasst sind.

→ Besonderheit: Der Schlüssel (Index) eines Arrays kann auch ein Text sein:

```
$FH[„Strasse“] = „Molkestraße 30“;
$FH[„Ort“] = „Karlsruhe“;
```

PHP - Variablen

- Weitere Variablenbeispiele:

```
$text = "Ich bin ein String !";

echo $text; echo ("$text,$text,$text");

$l = "langer";

$k = "kurzer";

echo "Ich bin ein $l$l$l$l$l$l Text!";

echo "Ich bin ein $k Text!";

$i = 10;

$j = 5; echo ($i."+".$j."=".$i+$j);
```

- Der Typ der Variablen (ganze Zahl, Gleitpunktzahl, String) wird je nach Verwendung von PHP automatisch bestimmt. Der Benutzer braucht sich darum nur in Spezialfällen kümmern.

PHP - Arrays

- Arrays werden genau wie einfache Variable durch einen mit $ beginnenden Namen bezeichnet. Elemente können durch einen Index oder assoziativ durch einen Schlüssel adressiert werden:
- Code

```
$wochentag ["So"]          = "Sonntag";
$wochentag []              = "Montag";
$wochentag ["Di"]          = "Dienstag";
$wochentag []              = "Mittwoch";
$wochentag [6]             = "Samstag";
while (list($key, $wert)   = each ($wochentag)){echo "$key
$wert<br>";}
```

- Ausgabe
  So Sonntag
  0 Montag
  Di Dienstag
  1 Mittwoch
  6 Samstag

PHP – einfache Rechenoperationen

- Folgende Rechenoperationen stehen in PHP zur Verfügung:

  "+": Addition, $i+$j,

  "-": Subtraktion, $i-$j

  "*": Multiplikation, $i*$j

  "/": Division, $i/$j

  "%": Reste-Bildung, $i%$j: z.B. 23%17 ergibt 6, da 23 geteilt durch 17 gleich 1 Rest 6 ist.

  ".": Verknüpft Strings: $l = "langer"; $k = "kurzer"; echo $l.$k; ergibt langerkurzer

- Dazu kommen noch ein paar Abkürzungen, um dem Programmierer das Leben zu erleichtern:

  $i++ erhöht $i um 1.

  ++$i erhöht $i ebenfalls um 1.

  $i-- erniedrigt $i um 1.

  --$i erniedrigt $i ebenfalls um 1.

  Der Unterschied zwischen $i++ und ++$i ist: $i=0; echo $i++; gibt 0 aus, anschließend wird $i auf den Wert 1 erhöht. $i=0; echo ++$i; erhöht zuerst $i auf 1 und gibt den Wert 1 aus.

PHP – Schleifen 1

- Beispiel:

```
$t = "Ich soll meine Uebungsaufgaben selbst erstellen!<BR>\n";
$i = 0;
while ($i<10) {
  echo $t;
$i++;
}
```

- Hier wird 10-mal der Text in der Variablen $t ausgegeben. Zu Beginn wird $i auf 0 gesetzt. $i wird in jedem Schleifendurchlauf um 1 erhöht, bis $i den Wert 10 erreicht. Dann ist die Bedingung ($i<10) nicht mehr wahr und die Schleife bricht ab.

PHP – Schleifen 2

- Hier eine 2. Möglichkeit eine Programmschleife zu erzeugen. Statt

```
$t = "Ich soll meine Uebungsaufgaben selbst erstellen!<BR>\n";
$i = 0;
while ($i<10) {
  echo $t;
  $i++;
}
```

kann man auch schreiben:

```
$t = "Ich soll meine Uebungsaufgaben selbst erstellen!<BR>\n";
$i = 0;
do {
  echo $t;
  $i++;
} while ($i<10);
```

Was ist der Unterschied?
Man bemerkt den Unterschied, wenn z.B. statt $i=0; zu Beginn $i=10; gesetzt wird. Im ersten Fall ist die Bedingung ($i<10) nicht wahr und die Befehle innerhalb der geschweiften Klammern werden nicht ausgeführt. Im zweiten Fall werden zuerst die Befehle innerhalb der geschweiften Klammern ausgeführt, danach wird getestet, ob ($i<10). Dies ist nicht der Fall, also wird abgebrochen. D.h. aber, die Schleife wird **mindestens einmal** durchlaufen.

PHP – Schleifen 3

- Eine weitere Möglichkeit, eine Schleife zu programmieren, ist der for-Befehl. Gleich ein Beispiel:

```
$t = "Ich soll meine Uebungsaufgaben selbst erstellen!<BR>\n";
for ($i=0;$i<10;$i++) {
  echo $t;
}
```

  Der for-Befehl besteht aus drei Ausdrücken. for (ausdruck1;ausdruck2;ausdruck3) { ... }
- Mit ausdruck1 wird die Schleife initialisiert, d.h. normalerweise wird die Variable, die die Schleifendurchläufe zählt, auf den Anfangswert gesetzt.
- ausdruck2 gibt die Abbruchbedingung an.
- In ausdruck3 wird die Variable, die die Schleifendurchläufe zählt, erhöht bzw. erniedrigt.
- Der for-Befehl hat den Vorteil, daß alle zur Kontrolle der Schleife nötigen Befehle in einer Zeile stehen. Ein weiteres Beispiel, diesmal wird heruntergezählt.

```
$t = "Ich soll meine Uebungsaufgaben selbst erstellen!<BR>\n";
for ($i=10;$i>0;$i--) {
  echo $t;
}
```

PHP – Fallunterscheidung 1

- Zur Fallunterscheidung gibt es den if-Befehl:

```
if ($i<0) {
  echo "$i ist kleiner als Null\n";
}
```

oder auch

```
if ($i<0) {
  echo "$i ist kleiner als Null\n";
} else {
  echo "$i ist nicht kleiner als Null\n";
}
```

Man kann diesen Befehl auch schachteln:

```
if ($i<0) {
echo "$i ist kleiner als Null\n";
} else if ($i>0) {
  echo "$i ist groesser als Null\n";
} else {
  echo "$i ist Null\n";
          }
```

PHP – Fallunterscheidung 2

- Hat man mehrere Tests der gleichen Variable, so kann man mit dem switch-Befehl evtl. Arbeit einsparen:

```
switch ($name) {
  case "Heinrich": echo "Ich bin der kluge Heinrich"; break;
  case "Hans": echo "Ich bin der dumme Hans"; break;
  case "Agathe": echo "Ich bin die Agathe "; break;
  default: echo "Wir sind der Rest";
}
```

- Falls die Variable $name den Wert "Hans" hat, wird als nächster Befehl echo "Ich bin der dumme Hans"; ausgeführt. Normalerweise würden dann alle nachfolgenden Befehle ausgeführt werden, u.a. echo "Ich bin die Agathe und klüger als Heinrich und Hans"; Dies ist meist nicht erwünscht, man springt deshalb mit break aus dem switch-Befehl heraus.

PHP – Weitere Vergleichsmöglichkeiten

- Bisher konnten wir testen, ob $i<10 oder $i>10 gilt. Es gibt aber noch mehr Möglichkeiten:

`$i==10`: Ist $i gleich 10?

`$i!=10`: Ist $i ungleich 10?

`$i>=10`: Ist $i größer oder gleich 10?

`$i<=10`: Ist $i kleiner oder gleich 10?

- Man kann auch kombinieren:

`($i==10) && ($j>0)`: Ist $i gleich 10 **und** $j größer als 0?

`($i==10) || ($j==0)`: Ist $i gleich 10 **oder** $j gleich 0?

PHP – Vordefinierte Variablen 1

- Es wurde bereits gezeigt, dass, in einer PHP-Datei, die aus einem Formular gestartet wird, die Namen der Formular-Elemente als Variablen zur Verfügung stehen. Dies kann man nützen, um Formular und Befehle, die das Formular behandeln, in eine einzige Datei zu schreiben.

- Angenommen unsere Datei form.html enthält folgendes Formular:

```
<FORM ACTION="form.html" METHOD=POST>
<INPUT NAME="beliebigername">
 <INPUT TYPE="submit">
 </FORM>
```

- Dann ruft sich beim Klicken auf **submit** die Datei selbst auf. Allerdings ist dann die Variable beliebigername beliebiger Name gesetzt. Dies können wir folgendermaßen ausnützen

PHP – Vordefinierte Variablen 2

- ```
  <? if (isset($beliebigername)) {
    echo "Sie haben $beliebigername eingegeben\n";
  } ?>
  <P>
  <FORM ACTION="form.html" METHOD=POST>
  <INPUT NAME="beliebigername">
  <INPUT TYPE="submit">
  </FORM>
  ```

- Wird die Datei zum ersten Mal aufgerufen, d.h. ohne daß man auf den submit-Button klickt, so ist die Variable $beliebigername nicht gesetzt, der Ausdruck isset($beliebigername) ist also nicht wahr. Füllt man das Formular dann aus und klickt auf submit, so ist die Variable gesetzt, isset($beliebigername) ist wahr, und es wird "Sie haben $beliebigername eingegeben\n"; ausgegeben. Will man danach abbrechen, so sollte der Befehl exit; verwendet werden.
  ```

PHP – Dateien hinzuladen

- Der Befehl
  `include("dateiname");`
- Liest den Inhalt der Datei „dateiname“ so, als ob er an dieser Stelle stehen würde. Damit kann z.B. einheitliches Layout bei einer größeren Anzahl von Dateien erreicht werden.
- Dateien werden unter dem angegebenen Pfad gesucht, oder, wenn kein Pfad angegeben ist, im **include_path**.
- Wenn im **include_path** die Datei nicht gefunden werden kann, sucht **include** noch im Verzeichnis der ausrufenden Datei und dem aktuellen Arbeitsverzeichnis.
- Wenn keine Datei gefunden wurde, erzeugt **include** eine Warnung, im Gegensatz zu **require** in diesem Fall wird ein Fatal_Error erzeugt.

PHP – Dateien hinzuladen Beispiel

- Datei vars.php

```
<?php

$farbe = ,grüen‘;
$frucht = ,Apfel‘;

?>
```

- Datei test.php

```
<?php

echo „Der $frucht ist $farbe.“;// Der ist .

Include ,vars.php‘;

echo „Der $frucht ist $farbe.“;// der Apfel ist grün.

?>
```

Funktionen in PHP

- PHP-Referenz (www.php.net oder) enthält mehr als 1400 Funktionen, Tendenz steigend.

- Auszug aus der Funktions-Referenz:
  → Mathematische Funktionen: sin(), exp(), round(), tan() …

- Beispiel: $ergebnis = sqrt($zahl); berechnet die Quadratwurzel

  → Zeichenkettenfunktionen: strlen(), strcmp(), sprintf() …
  → Datumfunktionen: date(), localtime() …
  → Dateisystemfunktionen: dir(), copy(), fopen() …
  → Netzwerkfunktionen: fsocketopen(), gethostbyname() …
  → PDF-Funktionen: pdf_open(), pdf_put_image() …
  → Shockwave-Flash-Funktionen: swf_actionplay() …
  → Mail-Funktionen: imap_fetchbody() …
  → Datenbankfunktionen für dBase, Informix, MySQL, Oracle, usw.

PHP – Lesen und Schreiben 1

- Wir wollen nun von Dateien, die auf dem Server liegen, lesen, bzw. in diese Dateien etwas hineinschreiben.
- Dies ist nützlich, um z.B. Zähler oder Gästebücher zu erzeugen.
- **Wichtig:**
  Aus Sicherheitsgründen ist der Webserver einem Benutzer auf dem Server-Computer zugeordnet, der sehr wenig Rechte hat. U.a. darf er nicht einfach in eine Datei in ihrem Verzeichnis schreiben. Genauso wenig darf er eine Datei in Ihrem Verzeichnis erzeugen.
- Hierzu müssen die Verzeichnisse bzw. die Dateien auf die zugegriffen wird, zum schreiben und lesen geöffnet werden (Betriebssystem abhängig).

PHP – Lesen und Schreiben 2

- Zuerst muss die Datei geöffnet werden, dabei muss angegeben werden, was mit der Datei geschehen soll: Wollen wir lesen, schreiben, anhängen,...
- Der Befehl lautet
```
$datei = fopen("xxx","r");
```
- Das heißt, wir öffnen die Datei mit dem Namen xxx und wollen darin lesen ("r"). Die technischen Details lassen wir dabei von der Variablen $datei regeln, d.h. wenn wir später aus der Datei lesen wollen, so benützen wir die Variable $datei.

PHP – Lesen und Schreiben 3

- Nun lesen wir **zeilenweise** den Inhalt der Datei, und zwar so lange, bis wir ans Ende der Datei gelangt sind:

```
while (!feof($datei)) {
  $zeile = fgets($datei,1000);
  echo $zeile;
}
```

- feof($datei) ist wahr, sobald wir an das Datei-Ende gelangt sind. $zeile = fgets($datei,1000); liest maximal die nächsten 1000 Zeichen, hört aber auf, sobald eine neue Zeile beginnt, oder das Ende der Datei erreicht ist.
  echo $zeile; gibt einfach das Gelesene wieder aus.

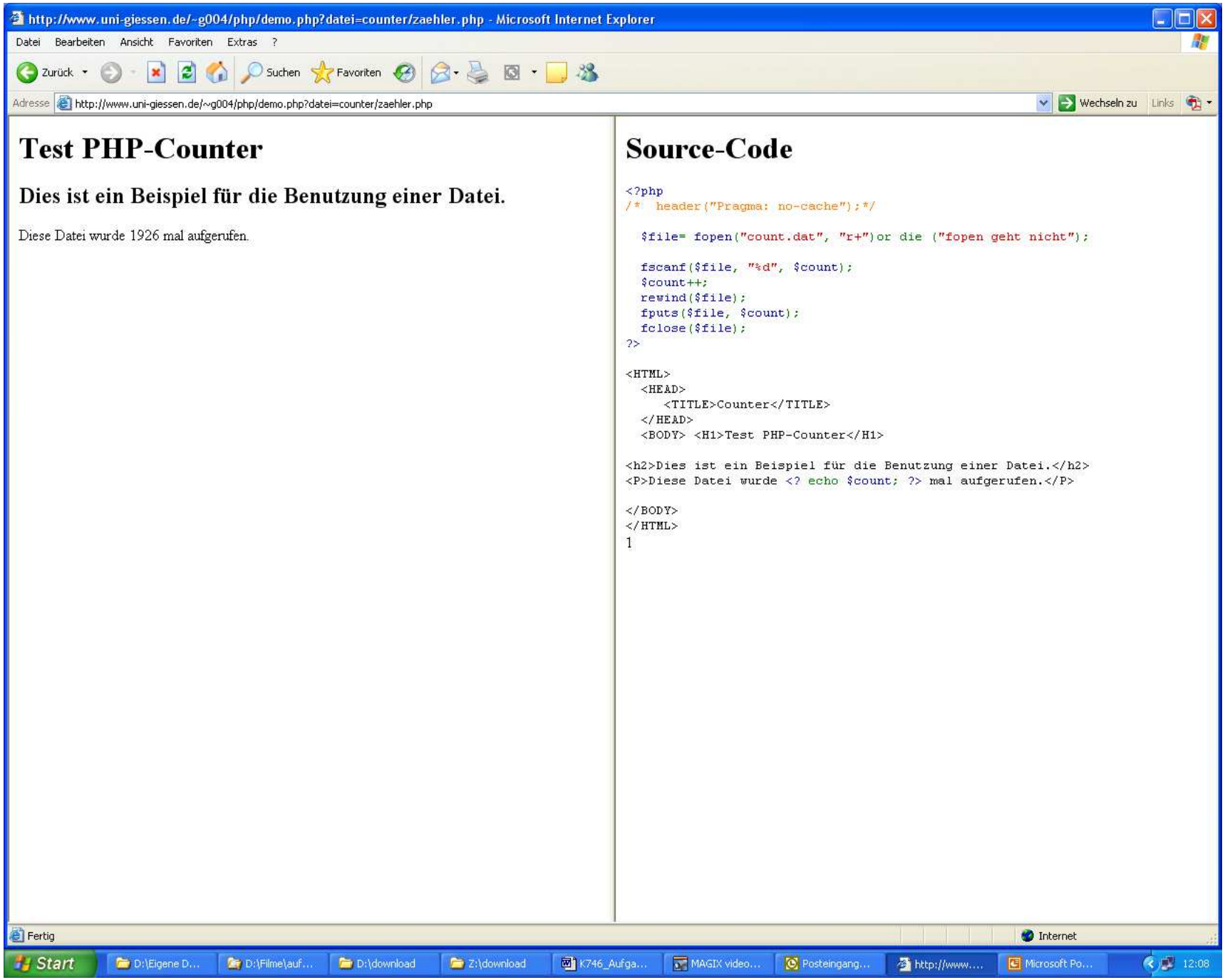
PHP – Lesen und Schreiben 4

- Schließlich muss die Datei noch geschlossen werden: fclose($datei);

  Folgende Möglichkeiten gibt es, eine Datei zu öffnen:

  ➔ "r": nur lesen, begonnen wird am Dateianfang.
  ➔ "r+": lesen und schreiben, begonnen wird am Dateianfang.
  ➔ "w": nur schreiben. Existiert die Datei bereits, wird der bisherige Inhalt gelöscht. Existiert sie nicht, wird versucht. sie zu erzeugen.
  ➔ "w+": lesen und schreiben. Ansonsten wie "w".
  ➔ "a": nur schreiben. Begonnen wird am Ende der Datei (a wie append, anhängen). Existiert sie nicht, wird versucht. sie zu erzeugen.
  ➔ "a+": lesen und schreiben. Ansonsten wie "a". **Achtung:** Es wird beim Lesen natürlich auch am Ende der Datei begonnen (dort steht natürlich nichts mehr...)

- Mit dem Befehl `fwrite($datei, "Ein Text");` kann der String `Ein Text` in die Datei geschrieben werden.


http://www.uni-giessen.de/~g004/php/demo.php?datei=counter/zaehler.php - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse http://www.uni-giessen.de/~g004/php/demo.php?datei=counter/zaehler.php
Wechseln zu Links
Test PHP-Counter
Dies ist ein Beispiel für die Benutzung einer Datei.
Diese Datei wurde 1926 mal aufgerufen.
Source-Code
<?php
/* header("Pragma: no-cache");*/
$file= fopen("count.dat", "r+")or die ("fopen geht nicht");
fscanf($file, "%d", $count);
$count++;
rewind($file);
fputs($file, $count);
fclose($file);
?>
<HTML>
<HEAD>
<TITLE>Counter</TITLE>
</HEAD>
<BODY> <H1>Test PHP-Counter</H1>
<h2>Dies ist ein Beispiel für die Benutzung einer Datei.</h2>
<P>Diese Datei wurde <? echo $count; ?> mal aufgerufen.</P>
</BODY>
</HTML>
1
Fertig
Internet
Start
D:\Eigene D...
D:\Filme\auf...
D:\download
Z:\download
K746_Aufga...
MAGIX video...
Posteingang...
http://www....
Microsoft Po...
12:08

Übung 1

- Die Aufgabe dieses Programms ist die Addition und Multiplikation mehrerer Zahlen und Anzeige des Rechenergebnisses.
- Zwei Tankfüllungen (*liter1*=40.5, *liter2*=35.7) mit dem gleichen Literpreis (*preis*=1.499) sind zu multiplizieren und als Kosten (*kosten*) in einem Aussagesatz anzuzeigen.
- Dafür wird im 1. Teil die Kostenermittlung durchgeführt.
- Im 2. Teil wird bei der Ergebnisanzeige eine Zeichenkette durch Verbindung aus: Der Zeichenkette "Die Benzinkosten betragen für ", der Addition der Variablen *liter1* + *liter2*, der Zeichenkette "Liter ", der Variablen *kosten* und der Zeichenkette "€" gebildet.
- Das Ergebnis sollte dann so aussehen:

- **Die Benzinkosten betragen für 76.2 Liter 114.2238 €**
-
- Eingesetzte PHP-Elemente: Speicherung in Variablen, Rechenoperationen, Zeichenketten

Lösung zu Übung 1

```
<html>
<head>
<title>PHP-Übung 11</title>
</head>
<body>
<?php
$tanken1 = 40.5;
$tanken2 = 35.7;
$preis = 1.499;
$kosten = ($tanken1 + $tanken2) * $preis;
$einheit1 = "Liter";
$einheit2 = "€";
$gesamt = "Tanken im Monat von " . ($tanken1 + $tanken2) . " " . $einheit1;
$gesamt .= " kosteten " . $kosten . " " .$einheit2 . "<p>";
echo $gesamt;
?>
</body>
</html>
```

Übung 2

- Die Aufgabe dieses Programms dient dazu mithilfe von einer Verzweigung auf unterschiedliche Bedingungen zu reagieren.
- Aufgrund der Wertangaben für eine Reiseentfernung (*entfernung* = 350) und den Kosten für den Entfernungskilometer einer Bahnfahrt (*bahn_km* = 0. 26) und einer PKW-Fahrt (*pkw_km* = 0.29) soll das Programm feststellen, welches Verkehrsmittel kostengünstiger ist und wie hoch der Kostenunterschied absolut und prozentual ist.
- Dafür wird im 1. Teil die Kostenberechnung mit Variablen (*entfernung, bahn_km, pkw_km; kosten_bahn, kosten_pkw*) ausgeführt.
- Im 2. Teil werden der Zeichenkettenvariablen (*aussage*) die Zeichenkette "Bei einer Entfernung von", die Variable *entfernung* und die Zeichenkette "km" zugewiesen.
- Im 3. Teil prüft die Verzweigung, welches Verkehrsmittel günstiger ist. Abhängig vom Ergebnis der Prüfung, errechnet die Variable *vorteil* jeweils die Kostendifferenz beider Verkehrsmittel absolut und die Variable *vorteil_rel* den Kostenunterschied relativ (in Prozent) zum jeweils günstigeren Verkehrsmittel.
- Sodann wird in jedem Verzweigungszweig die Zeichenkettenvariable *aussage* verlängert mit der Zeichenkette "ist die Bahnfahrt um" bzw. "ist die Fahrt mit dem PKW um", der Variablen *vorteil,* der Zeichenkette "*€ =",* der Variablen *vorteil_rel* und der Zeichenkette "% günstiger".
- Im 5. Teil wird die Zeichenkettenvariable *aussage* angezeigt.
- Das Ergebnis sieht dann für den einen Fall mit den vorgegebenen Werten so aus:
- **Bei einer Entfernung von 350 km ist die Bahnfahrt ist um 10.5 € = 11.538461538462 % günstiger.**
- Aufgabenerweiterung: Verändern Sie die Wertangaben im Programm so, dass ein Vorteil für die PKW-Nutzung entsteht!
- Eingesetzte PHP-Elemente: Zuweisungen an Variablen, einfache Verzweigung

Lösung zu Übung 2

```
<?php
  // Wertzuordnung und Berechnung der Kosten
  $entfernung = 355;
  $bahn_km = 0.26;
  $pkw_km = 0.29;
  $kosten_bahn = $entfernung * $bahn_km;
  $kosten_pkw = $entfernung * $pkw_km;
  $aussage = "Bei einer Entfernung von " . $entfernung . " km "; // Aufbau Zeichenkette
  if ( $kosten_bahn < $kosten_pkw) // Verzweigung
  {
    $vorteil = $kosten_pkw - $kosten_bahn;
    $vorteil_rel = $vorteil * 100 / $kosten_bahn;
    $aussage .= "ist die Bahnfahrt ist um " . $vorteil . " € = " . $vorteil_rel . " %
    günstiger.";
  }
  Else
  {
    $vorteil = $kosten_bahn - $kosten_pkw;
    $vorteil_rel = $vorteil * 100 / $kosten_pkw;
    $aussage .= "ist die Fahrt mit dem PKW ist um " . $vorteil . " € = " . $vorteil_rel . "
    %günstiger.";
  }
  echo $aussage; // Ergebnisanzeige
?>
```