PHP – Funktionen, Globale Variablen

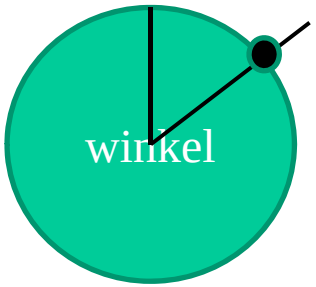
Werden in einer Funktion globale Variablen benutzt, so sind diese als global zu deklarieren.
Beispiel:

```
$xmitte=300;
$ymitte=300;
$radius=150;

function kreispunkt($winkel, &$xpos, &$ypos)
{
        global $xmitte;
        global $ymitte;
        global $radius;

        $xpos = $xmitte + sin($winkel) * $radius;
        $ypos = $ymitte - cos($winkel) * $radius;
}
```

Die Funktion kreispunkt () berechnet die Koordinaten eines Punktes auf einem Kreis, der einem gegebenen Winkel entspricht. Die Kreisparameter werden als globale Variablen übernommen.



PHP – Formularverarbeitung (1)

Die superglobalen Variablen $_GET bzw. $_POST enthalten als Feld Informationen, die auf der Clientseite in Formularelemente, wie INPUT-Elemente, RADIO-, CHECK-Felder usw. eingetragen wurden und serverseitig abrufbar sind.

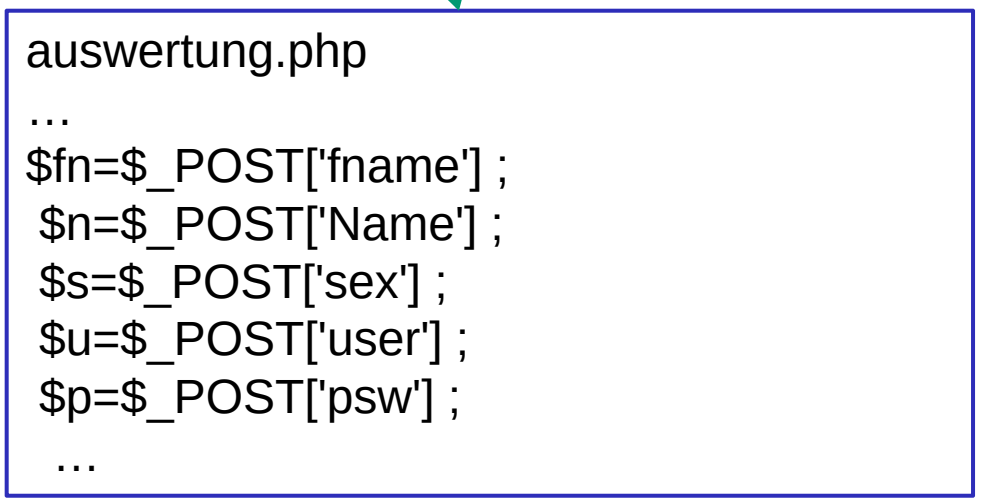
Die Namen der superglobalen Variablen weisen dabei auf die Übertragungsart hin. Ist man nicht sicher, welche Methode, GET oder POST verwendet wurde, sollte anstelle dieser, die superglobale Variable $_REQUEST verwendet werden.

Die superglobalen Variablen $_GET, $_POST, $_COOKIE, $_SERVER, $_FILES, $_ENV, $_REQUEST sind assoziative Felder.

PHP – Formularverarbeitung (2)

Formular in PHP

```
<FORM method="POST" action=“auswertung.php" >
<INPUT NAME="fname" TYPE="text" VALUE=“”>
<INPUT NAME="Name" TYPE="text" VALUE="">
<INPUT NAME="sex" TYPE="radio" VALUE="male" CHECKED>
<INPUT NAME="sex" TYPE="radio" VALUE="femal">
<INPUT NAME="user" TYPE="text" VALUE="">
<INPUT NAME="psw" TYPE="password" VALUE="">
</FORM>
```



PHP – Formularverarbeitung (3)

Im assoziativen Feld muss der Name des betreffenden Formularelementes (Wert des Name-Attributes) als Index angegeben werden. Angenommen es existiert im Formular ein Inputfeld:

<INPUT type="text" name="Vorname" />

so kann im Skript mit dem PHP-Ausdruck *$_POST['Vorname']* der eingetragene Wert übernommen werden.

Sollte die Herkunft des Wertes über GET, POST oder COOKIE keine Rolle spielen, ist zweckmäßigerweise die superglobale Variable $_REQUEST für die Übernahme zu verwenden.
Ob ein Werteeintrag erfolgte, kann mit der PHP-Funktion isset() getestet werden.
Beispiel: *$_REQUEST['Vorname']*

PHP – Formularverarbeitung (4)

Upload von Dateien durch Formulare:

Formular auf Webseite:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
 <input type="file" name="datei"><br>
 <input type="submit" value="Hochladen">
 </form>
```

PHP-Skript, das per HTTP-POST-Operation aufgerufen wird:

```
move_uploaded_file($_FILES['datei']['tmp_name'], "uploaddatei.txt");
```

Der Dateiinhalt und die Angaben zur Datei werden automatisch im Body des HTTP-Requests transportiert und können durch das PHP-Skript in der Variable $_FILES abgefragt werden. Die Datei wird als temporäre Datei auf Serverseite abgelegt. Den Namen der temporären Datei fragt man mit *$_FILES['datei']['tmp_name']* ab. Der Bezeichner *'datei'* stellt die Beziehung zu dem im Formular angegebenen Identifikator her.

PHP – Dateiarbeit (1)

Das Dateikonzept von PHP hat auch große Ähnlichkeit mit dem Dateikonzept der Sprache C. Die meisten Funktionen zum Öffnen, zum Schließen, zum Lesen und zum Schreiben sind identisch.

- Es stehen mehr als 50 Funktionen für Dateiarbeit verfügbar. Hier werden nur die wichtigsten Funktionen behandelt.
- Unterschiede gegenüber Unix bei der Arbeit mit Verzeichnissen.
- Durch PHP kann auf Serverseite auch auf Pfade zugegriffen werden kann, die außerhalb des Publikationsverzeichnisses des Webservers liegen.

PHP – Dateiarbeit (2)

Das Schema der Arbeit mit Dateien besteht aus folgenden Schritten:

- Datei öffnen mit ***fopen()*** , es wird ein Handle (Adresse) angelegt; evtl. wird eine nicht existierende Datei zum Schreiben angelegt; wenn Fehler auftreten werden diese behandelt
- treten keine Fehler auf, werden zum Lesen geöffnete Dateien bis zur Erkennung des Datei-Ende (EOF) gelesen, z.B. mit ***fread()***; zum Schreiben geöffnete Dateien werden solange wie notwendig geschrieben, z.B. mit ***fwrite()***
- Datei schließen mit ***fclose()***

Vor dem Öffnen der Datei kann geprüft werden, ob die Datei existiert oder es können auch Operationen mit dem Verzeichnis vorgenommen werden.

PHP – Dateiarbeit (3)

file_exists()

Eine Datei kann mit dieser Funktion auf Vorhandensein geprüft werden. Als Argument ist der Pfad einschließlich Dateinamen anzugeben. Im Erfolgsfall gibt die Funktion true zurück.

is_file()

Eine Datei kann mit dieser Funktion auf die Eigenschaft reguläre Datei geprüft werden, also z.B. kein Verzeichnis. Als Argument ist der Pfad einschließlich Dateinamen anzugeben. Im Erfolgsfall gibt die Funktion true zurück.

is_readable() / is_writeable()

Eine Datei kann mit diesen Funktionen auf Vorhandensein des Leserechts /Schreibrechts geprüft werden. Als Argument ist der Pfad einschließlich Dateinamen anzugeben. Im Erfolgsfall gibt die Funktion true zurück.

Beispiel: ***file_check.php***

PHP – Dateiarbeit (4)

Eine Formatierte Datei- Eingabe wird durch die PHP-Funktionen ***fscanf()*** erreicht:

mixed fscanf(ressorce handle, string formatstring [, string var 1])

Die gelesenen Variablen können dem zurückgegebenen Array oder den Parametern nach *formatstring* entnommen werden.

Beispiel:

```
<?php
 $handle= fopen("meinedatei.txt","r");
 while ( $info = fscanf($handle, "%s \t %s \n", ))
 { list($wert1, $wert2)= $info;
   // verarbeite wert1 und wert2
 }
 fclose ($handle);
?>
```

PHP – Dateiarbeit (5)

Datei-Management: copy, rename, unlink (Löschen)

Verzeichnis-Zugriff: mkdir, chdir, rmdir, opendir, readdir, rewinddir oder Nutzung der Klasse dir

Schnittstelle und Verhalten der Funktionen weitestgehend nach Clib-Schnittstelle (C- Standardbibliothek)

Verzeichnis auflisten (klassisch):

```
$verzeichisname = "/";
$verzeichnis = opendir($verzeichnisname);
while($dateiname = readdir($verzeichnis))
   printf("%s <br/>\n",$dateiname);
closedir($verzeichnis);
```

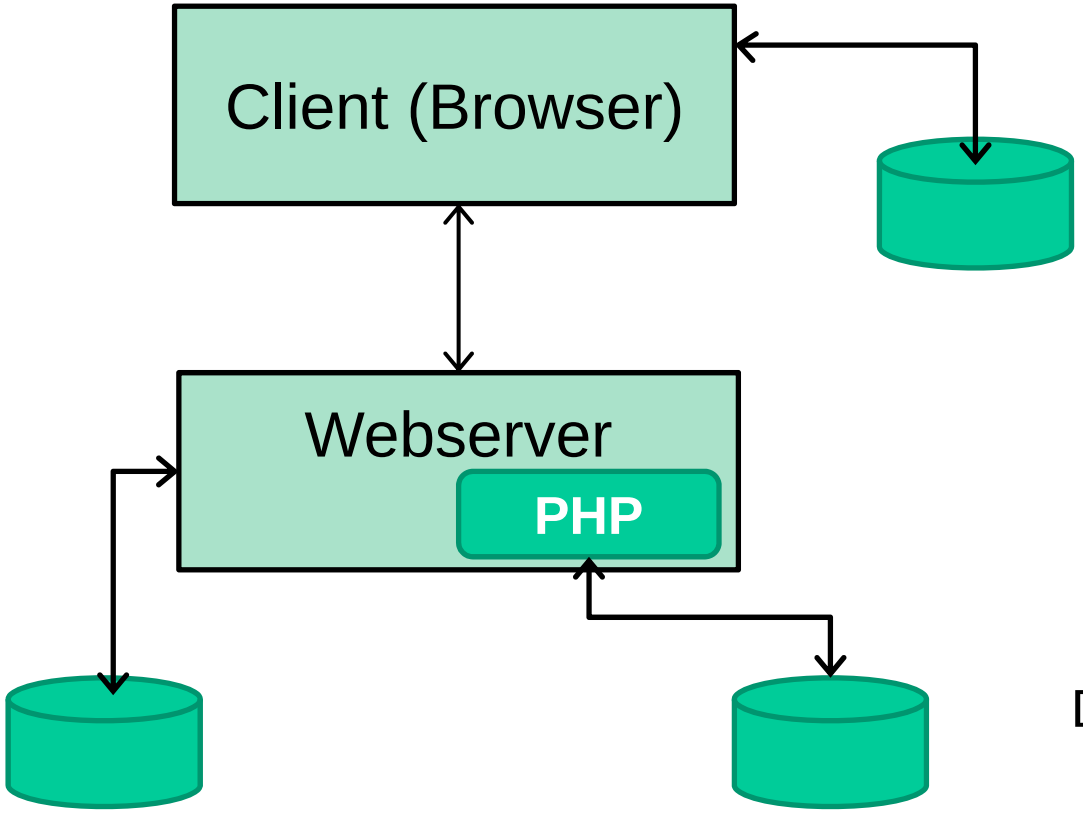
PHP – Dateiarbeit (6)

Verzeichnisarbeit bei Benutzung der Dir-Klasse:

```
$verzeichnis = dir("/");
while($dateiname=verzeichnis->read())
        printf("%s <br/>\n",$dateiname);

$verzeichnis->close($verzeichnis);
```

PHP – Cookies (1)



Cookies zur clientseitigen Speicherung von Informationen

Dateien und Datenbanken zur serverseitigen Speicherung von Informationen

Dateien als Speicherform der HTML-Seiten, PHP-Skripte, eingebetteten Inhalte

PHP – Cookies (2)

Aufbau eines Cookies

Die nachfolgende Tabelle gibt den schematischen Aufbau eines Cookie wieder.

Funktion	Beschreibung
name	Name des Cookies
value	die zu speichernden Daten
expires	Verfallsdatum des Cookies
path	legt fest welcher Pfad das abgelegte Cookie wieder lesen darf
domain	legt die Domain fest, die dieses Cookie auslesen kann
secure	beschränkt den Cookie auf SSL-Verbindungen

Der Nutzinhalt eines Cookies ist die Name-Wert-Angabe. In der Regel besteht ein Cookie aber aus mehreren Name-Wert-Paaren, die dann beim Auslesen die Elemente eines assoziativen Feldes bilden. In PHP heißt das assoziative Feld $_COOKIE.

PHP – Cookies (3)

PHP-Funktionen zum Umgang mit Cookies

Sind die Namen bekannt, z.B. Name1, so erhält man mit $_COOKIE[Name1] den zugeordneten Wert. Sind die Cookie-Namen nicht bekannt, so lassen sich alle Inhalte über eine foreach-Schleife auslesen, wie es das Skript-Beispiel zeigt.

```
<?php
// Durch den Web-Server generierter Cookie-Inhalt
echo "\n<b> \$_COOKIE - Inhalt (Elemente zeilenweise):\n";
   foreach ($_COOKIE as $k => $v) {
      echo "<br>   $k = $v\n";
   }; echo "</b>"
?>
```

Demonstration: *cookie-1.php*

PHP – Cookies (4)

PHP-Funktionen zum Umgang mit Cookies (Fortsetzung)

Das Verfallsdatum ist in Sekunden nach dem 1.1.1970 anzugeben. Den aktuellen Zeitstempel holt man sich dabei zweckmäßigerweise durch den Aufruf der time()-Funktion und addiert das Verfallsintervall in Sekunden.

Die Anzahl der Cookies, d.h. der vorhandenen Name-Wert-Paare lässt sich durch die PHP-Funktion count() ermitteln. Der Aufruf count($_COOKIE) liefert die Anzahl der Cookies.

Mit der Funktion array_keys() lassen sich die Namen der Name-Wert-Paare ermitteln, wobei der Index des Paares in einer foreach-Schleife definiert werden muss. Der erste Index ist 0. Der Aufruf in der Schleife muss array_keys($_COOKIE) sein.

Beispiel-Skript: cookie-5.php zur Verwendung dieser PHP-Funktionen.

PHP – Cookies (5)

PHP-Funktionen zum Umgang mit Cookies (Fortsetzung)

Funktion ***setcookie()*** zum Erzeugen von Cookies.
Die Argumente des Aufrufs entsprechen dabei der Reihenfolge der Parameter in der angegebenen Tabelle.

Syntax: *setcookie(name, wert, expires, path, domain, secure);*

Beispiel: *setcookie(“user”, “unbekannt”, time()+3600*24);*

Der Aufruf von *setcookie()* muss dabei, zeitlich vor allen Ausgaben der Cookie-Namen und -Values erfolgen, damit das Cookie sichtbar ist!

Beispiel: cookies-2.php umseitig

PHP – Cookies (6)

Beispiel für das Setzen von Cookies:

```
<?php
#Hier wird Cookie-Inhalt erzeugt
setcookie("Nutzer","Mustermann",time()+3600);
setcookie("Profil","Standard",time()+3600);
setcookie("Login","geheim",time()+3600);

echo "<b>Auslesen von übertragenen Cookie-Inhalten:</b><br/>\n";
echo "\n<b> \$_COOKIE - Inhalt (Elemente zeilenweise):\n";
  foreach ($_COOKIE as $k => $v) {
     echo "<br>   $k = $v\n";
  }; echo "</b>"
?>
```

PHP – Cookies (7)

Verändern von Cookie-Inhalten in PHP

Zum Verändern von Cookie-Inhalten müssen die Namen der Name-Wert-Paare bekannt sein.
Es ist dann nur ein neuer Aufruf von setcookie() mit dem Namen und dem neuen Inhalt zu kodieren.
Für den expires-Parameter ist eine 0 (keine Änderung!) anzugeben.

Angenommen, es ist der Inhalt des Cookie Login in „Niemand“ zu verändern. Dann genügt ein Aufruf *setcookie(“Login“, “Niemand“, 0);*

Beispiel-Skript cookies-3.php zur Erzeugung von Cookies

PHP – Cookies (8)

Löschen von Cookie-Inhalten:
Dazu müssen die Namen der Name-Wert-Paare bekannt sein.
Es ist dann nur ein neuer Aufruf von setcookie() mit dem Namen und einer leeren Zeichenkette für den Inhalt zu kodieren. Für den expires-Parameter ist time() anzugeben.

Damit wird das Verfallsdatum auf die jetzt geltende Zeit gesetzt, was einem Löschen des betreffenden Cookies entspricht!
Angenommen, es ist das Cookie Login zu löschen. Dann genügt ein Aufruf *setcookie(“Login“, ““, time());*

Beispiel-Skript cookies-4.php zur Erzeugung von Cookies