Übersicht

Was ist PHP?

PHP ist die Abkürzung für **Professional Homepage Preprocessor**. Die Idee ist folgende: Webseiten, sprich HTML-Seiten werden im Internet auf Webservern bereitgehalten, die der Benutzer im Browser betrachten kann. Man führt nun in HTML-Seiten ein spezielles Tag ein, das Anfang bzw. Ende eines php-Scriptes kennzeichnet. Die HTML-Seite hat dann meist nicht mehr die Endung "*.html", sondern beispielsweise "*.php". Der Webserver erkennt bei einer solchen Seite: "Damit kann der php-Interprter etwas anfangen", schickt die Seite zum PHP-Interpreter, der geht Zeile für Zeile durch, ist es normaler HTML-Code, so wird's unverändert ausgegeben, trifft er aber auf das php-Tag:

```
<?php
  // Hier steht irgendwelcher php-Code
  // Dies ist übrigens ein Kommentar
  // ähnlich wie in Java . . .
  echo "Dies ist ein grandioses PHP-Programm!\n";
?>
```

dann werden die php-Befehle ausgeführt und erscheinen in dieser Form auf der HTML-Seite im Browser. Hier das Ergebnis des php-Interpreters:

```
X-Powered-By: PHP/4.0.4pl1
Content-type: text/html

Dies ist ein grandioses PHP-Programm!
```

Im Browser sieht das dann so aus:

Worin besteht der Unterschied zwischen diesen beiden Darstellungen? Oben ist einfach nur die Ausgabe des php-Interpreters dargestellt, im zweiten Bild sehen wir dagegen eine "richtige" WWW-Seite von einem Webserver (apache) im Browser (Opera). Der Webserver meldet sich beim Start folgendermaßen

```
Apache/1.3.12 (Win32) PHP/4.0.4pl1 running...
```

ein sicherer Hinweis, das jemand den Webserver so konfigueriert hat, das er php-Skripte ausführen kann (Wer?). Nebenbei: die apache-Version ist ziemlich alt . . .

<<<

Ein simples PHP-Skript

Schauen wir uns ein richtiges php-Skript an:

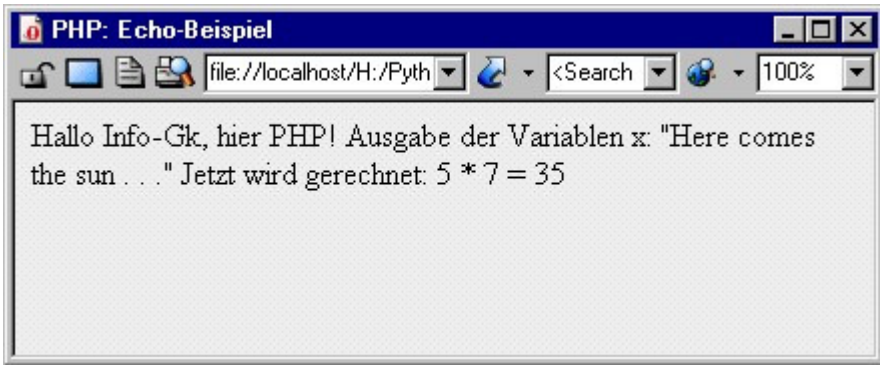
```
<html>
<head>
<title>PHP: Echo-Beispiel</title>
<!--
  HTML-echo-Beispiel in php, ws, 2002-04-01
-->
</head>
<body bgcolor="#efefef" text="#000000">
<?php
        echo "Hallo Info-Gk, hier PHP!\n";
        $x="Here comes the sun . . .";
        echo "Ausgabe der Variablen x: $x\n";
        $a=5;
        $b=7;
        $c= $a * $b;
        echo "Jetzt wird gerechnet: $a * $b = $c\n";
?>
</body>
</html>
```

Der PHP-Interpreter erzeugt folgende Ausgabe:

```
X-Powered-By: PHP/4.0.4pl1
Content-type: text/html


<html>
<head>
<title>PHP: Echo-Beispiel</title>
<!--
  HTML-echo-Beispiel in php, ws, 2002-04-01
-->
</head>
<body bgcolor="#efefef" text="#000000">
Hallo Info-Gk, hier PHP!
Ausgabe der Variablen x: Here comes the sun . . .
Jetzt wird gerechnet: 5 * 7 = 35
</body>
</html>
```

Die HTML-Seite können wir uns auch im Browser anschauen:



Man sieht: schreiben wir in unseren php-Code kein HTML-Line-Break:
, dann gibt's auch keine neue Zeile. Wichtig sind folgende Punkte:

- **echo**: Der echo-Befehl in php leistet in etwa das gleiche wie der Write-Befehl in Delphi. Man benutzt ihn gerne zur Erzeugung von HTML-Code, Beispiel:
  `echo "Hallo Info-Gk, jetzt kommt ein Line-Break:<br>\n";`

- **$**Variablenname: Mit dem **$**-Zeichen fängt in php ein Variablenname an, im Gegensatz zu Delphi muss man Variablen **nicht** vereinbaren, sondern kann munter Variablen ins Leben rufen . . .

- Der **;** beendet in PHP eine Anweisung: `$a=5;`
  Vergisst man den Strichpunkt am Ende einer Anweisung, gibt's eine Fehlermeldung:
  `<b>Parse error</b>: parse error in . . .`

- HTML-Befehle werden vom PHP-Interpreter ungefiltert weiter gereicht, sprich: überlesen. Das ist Absicht!

<<<

---

for-Schleifen in PHP

Die for-Schleife funktioniert in php ähnlich wie in Java:

```
<?php
        // for-schleife in php
        $x="Here comes the sun . . .";
        echo "Jetzt kommt eine FOR-Schleife:\n";
        for ($i=1; $i<=5; $i = $i + 1):
                echo "$x\n";
        endfor;
?>
```

Zur Erklärung:

- **$i=1;** Damit wird der Schleifenvariable $i der Startwert zugewiesen.

- **$i<=5;** Das ist die sogenannte Abbruchbedingung, denn die for-Schleife soll ja irgendwann mal verlassen werden . . .

- **$i = $i + 1** Bei jedem Schleifendurchlauf soll die Variable $i um eins erhöht werden, C-Programmierer schreiben an dieser Stelle gerne `$i++` (meinen aber dasselbe), diese Schreibweise -also `$i++`- ist in PHP auch erlaubt.

- Die for-Anweisung beginnt nach dem Doppelpunkt
  `for ($i=1; $i<=5; $i = $i + 1):`
  und geht bis zur Zeile
  `endfor;`
  Auch hier gibt es eine Java- bzw. C-orientierte Schreibweise:

  ```
  <?php
          // for-schleife in php
          $halloc="Hallo C-Programmierer . . .";
          for ($i=1; $i<=3; $i = $i + 1)
          {
                  echo "$halloc\n";
          }
  ?>
  ```

  Der Block, also die Befehle innerhalb der for-Schleife werden jetzt durch geschweifte Klammern **{ }** eingerahmt, dafür ist der Doppelpunkt sowie das `endfor;` verschwunden.

Mir persönlich gefällt natürlich der Doppelpunkt sowie das `endfor;` besser . . .

Hier eine Probelauf:

```
X-Powered-By: PHP/4.0.4pl1
Content-type: text/html

Jetzt kommt eine FOR-Schleife:
Here comes the sun . . .
Here comes the sun . . .
Here comes the sun . . .
Here comes the sun . . .
Here comes the sun . . .
```

<<<

Beispiele

Siehe oben, später mehr . . .

<<<

Aufgaben

1. Schreibe ein eigenes PHP-Skript, das mittels `echo` den Namen ausgibt!
2. Lass PHP etwas rechnen, versuche auch eine Division durch Null!
3. Probiere den Namen mit Variablen und vermische das Ganze mit HTML.
4. Hier eine Batch-Datei zum Aufruf des php-Interpreters:

   ```
   v:\
   v:\php\php.exe %1
   ```

   mit Notepad abspeichern unter dem Namen **php.bat** im eigenen Verzeichnis, das php-Skript mit der Maus auf das Symbol von php.bat fallenlassen und hoffen, dass es funktioniert . . .
5. Erzeuge folgende Ausgabe in PHP mit Hilfe von for-Schleifen!

   ```
   X-Powered-By: PHP/4.0.4pl1
   Content-type: text/html

    Here comes the sun . . .
     Here comes the sun . . .
      Here comes the sun . . .
       Here comes the sun . . .
        Here comes the sun . . .
         Here comes the sun . . .
          Here comes the sun . . .
           Here comes the sun . . .
   ```

<<<
