PHP-Kurs 1: Einführung und PHP-Grundlagen

07.08.2005

Christoph Hermann, Dominik Benz
{hermann,dbenz}@informatik.uni-freiburg.de

Übersicht

1. Organisatorisches
2. Aufbau des Kurses
3. Einführung in PHP

Zeit, Ort, Infos

- Montag, 07.08. bis Mittwoch, 09.08.2005
- 09:00 Uhr bis 13:00 Uhr
- Raum 101-00-010/014
- Homepage (Sommercampus-Wiki):
  http://sommercampus2006.informatik.uni-freiburg.de
- Fragen, Kommentare, Probleme:
  {hermann,dbenz}@informatik.uni-freiburg.de}

Übersicht

1. Organisatorisches
2. Aufbau des Kurses
   - Inhalt
   - Ablauf
3. Einführung in PHP

Inhalte

Montag	Dienstag	Mittwoch
- Einführung - Orga - Einrichtung der Arbeitsumgebung - PHP Grundlagen	- PHP Grundlagen 2 - Objekte in PHP	- Datenbankanbindung - PEAR

Vertiefung, andere Themen, etc.: melden, melden, melden!

Ablauf

Vortrag zum Thema
Pause
Individuelles Arbeiten, Aufgaben, Kontrollfreagen
Pause
Diskussion aufgetauchter Probleme
Pause
weiteres Programmieren, eigene Projekte (?)

Übersicht

1. Organisatorisches
2. Aufbau des Kurses
3. Einführung in PHP
   - Basics
   - Variablen, Datentypen
   - Operatoren, Kontrollstrukturen
   - Funktionen, Module

Was ist PHP

- PHP (PHP: Hypertext Preprocessor) ist für viele Umgebungen verfügbar:
  - Betriebssysteme: Unix, Mac OS, Windows
  - Webserver: Apache, Microsoft IIS, Xitami, ...
  - auch möglich: command line scripting, desktop applications (PHP-GTK)
  - sehr verbreitet: LAMPP (Linux+Apache+MySQL+PHP+Perl)
  - PHP ist freie Software
- Installation: http://www.php.net/manual/en/install.php
- Standard: alle Dateien mit Endung `*.php` werden verarbeitet (kann man ändern!)
- kein besonders Verzeichnis für PHP-Skripte (wie beim cgi-bin)

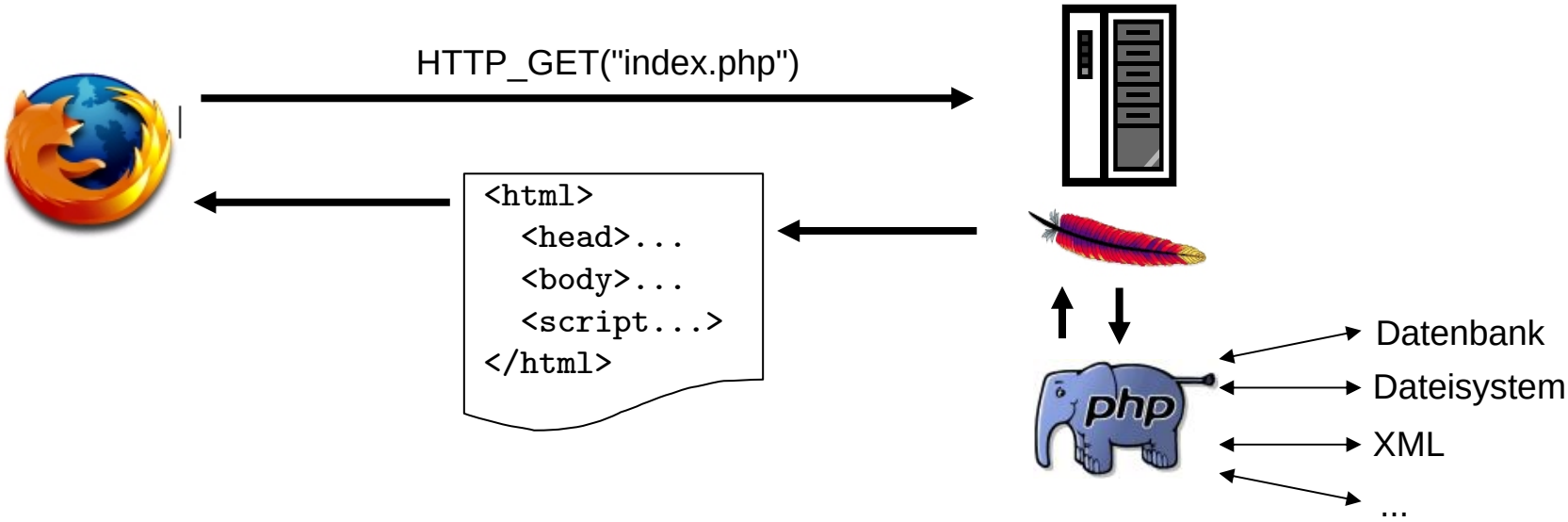
Ist bei meinem Webserver PHP installiert?

- Uni-Webspace (http://www.informatik.uni-freiburg.de/~LOGIN) → nein :-(
- RZ-Webspace (https://www.rz.uni-freiburg.de/omnibus/) → ja! V4.3.x :-)
- sonst → hello-world-script ausprobieren

Dynamische Inhalte auf Webseiten

Zwei grundlegende Möglichkeiten:

- clientseitiges Scripting (JavaScript, ...)
  - Daten und Programmcode müssen in übertragener Webseite enhalten sein
- serverseitiges Scripting (JSP, ASP, Perl, ..., PHP (="PHP: Hypertext Pre-processor"))
  - Webseite wird erst bei Anfrage generiert



Einbindung in HTML

Gebräuchlichste, portabelste und empfohlene Methode:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN"
<html>
<head>
<title>Meine erste PHP-Seite</title>
</head>
<body> <h1>PHP-Test</h1>
<?php
echo '␣Hello␣World!␣';
?>
</body>
</html>
```

- beliebig viele PHP-Bereiche ("php parsing mode") möglich
- jeder Ausdruck endet mit Semikolon ;
- `echo` "schreibt" die Ausgabe in die HTML-Datei; alternativ: `print`
- Browser erhält "reines" HTML ohne PHP-Tags und -Befehle

Installation, Konfiguration, Error-Reporting

- quick and easy unter Windows: XAMPP
- Experteninstallation: siehe Kapitel 1.5 im Skript
- effiziente Entwicklungsumgebung: Eclipse + PHPEclipse
- beim Entwickeln: Error-Reporting hochschrauben:

```
<?php
error_reporting(E_ALL);
?>
```

Kommentare

```
2  echo "einzeilig, C++"; // bis Zeilenende
3  echo "einzeilig, Unix Shell"; # bis Zeilenende
4  echo "mehrzeilig, C";
5  /*
6    mit Slash (/)
7    und Sternchen (*)
8  */
9  echo "Funktionen sollten mit Doc-Comments kommentiert werden!
10 /**
11  * Beschreibung der Funktion foo
12  */
13 function foo () {
14   echo "bar";
15 }
```

Doc-Comments sind kein offizieller Bestandteil der Syntax, bilden aber die Basis einer automatischen API-Dokumentation (z.B. mit phpDocumentor)

Variablen

einfaches Variablenhandling:

- dynamische Erzeugung durch Benutzung
  - Deklaration, Definition nicht notwendig, jedoch guter Stil
- schwach typisiert, Typwechsel immer möglich (!)
  - type casting: `(typ)$variable` und `settype($variable, typ)`
  - Typ ermitteln:
    `gettype($variable),is_int($variable),is_float($variable)..`
- bei Bedarf: automatische Typwandlung (!)
- Löschung: (ab PHP4) automatisch (Referenzzähler, Garbage Collection); manuell: `unset($variable)`

Variablennamen beginnen immer mit `$`, gefolgt von _ oder Buchstabe; sie sind case sensitiv (`$a != $A`)

- im Projekt auf aussagekräftige, lesbare Konvention einigen (s. http://pear.sourceforge.net/en/standards.naming.php

Datentypen

8 Datentypen:

- skalar (primitiv):
  - Integer, Float(Double), Boolean, String
- zusammengesetzt:
  - Array, Object
- weitere:
  - NULL, "Resource ID" (z.B. von Ergebnis einer Datenbankanfrage)

typecasting einer variablen nach boolean liefert immer `true`; Ausnahme:

- `integer` 0
- `float` 0.0
- der leere `string` oder `"0"`
- ein leeres `array`
- ein `object` ohne Variablen
- vom Typ `NULL` ist

`var_dump($var)` gibt Wert und Typ von `$var` aus

```
2 $name = "Fred";
3 if ($name){
4   echo "Fred ist wahr!<br>";
5 }
6 $zahl = 0;
7 if ($zahl){
8   echo "Null Ahnung!<br>";
9 }
```

Datentyp: String

- Auswertung von Variablen innerhalb eines Strings (Variableninterpolation) nur innerhalb doppelter Anführungszeichen `"`, nicht bei `'`
- Konkatenation mit `.`
- Zugriff auf einzelne Zeichen mit `$string[2]`
- "Escapen" von Sonderzeichen `$`, `"`, `'`, `\` mit `\`; ausserdem:
  - `\n` (Unix) oder `\r\n` (Windows): Zeilenumbruch
  - `\r`: Wagenrücklauf
  - `\t`: Tabulator

```php
<?php
$i=5;
$str = 'nicht ausgewertet: $i';
echo $str; // nicht ausgewertet: $i
$str = "ausgewertet: $i";
echo $str; // ausgewertet: 5
?>
```

- Heredoc-Syntax:

```php
<?php
$a = <<<EOS
String
mit Zeilen-
umbruch
EOS;
?>
```

Datentyp: Array

Datenstruktur, bei der über Schlüssel auf zugeordnete Werte zugegriffen wird; flexibel und praktisch:

- bei Bedarf automatische Vergrösserung
- beliebige Datentypen für Werte; für Schlüssel: `integer` (indiziert) oder `string` (assoziativ)
- auch verwendbar als Vektor, Hash-Tabelle, Collection, Stack, Queue, ...
- mehrdimensionale Arrays möglich
- bei Bedarf automatische Schlüssel-Generierung (indiziert)
- eigenes Sprachkonstrukt zum Durchlaufen von Arrays: `foreach`
- grosse Bibliothek von Funktionen zur Manipulation/Sortierung/Filterung von arrays
  (`array_merge, sort, array_search, array_map...`)

								"b"	*1*
								"a"	*0*
value	"a"	5	23	"Peter"	true	3	11		...
key	*0*	*1*	*2*	*"name"*	*"0"*	*3*	*4*	*12*	

Array - Beispiele

```
1  <?php
2  $arr = array('123','345');
3  $arr = array(1,2,3,4);
4  $arr = array(
5    1 => 'mystr1',
6    2 => 'mystr2',
7    "str" => "iHaveAStringKey"
8  );
9  // Mehrdimensionale Arrays
10 $arr2[1][3] = 'test';
11 // Automatische Schlüsselerzeugung
12 $arr3 = new array();
13 $arr3[] = 4;
14 $arr3[] = 5;
15 echo $arr3[1];
16 ?>
```

01-array.php

vordefinierte Variablen, Konstanten

- "superglobals" (assoziative Arrays):

`$_POST`	per HTTP POST übergeben
`$_GET`	per HTTP GET übergeben
`$_SESSION`	Session-Variablen
`$_REQUEST`	vereinigt `$_POST, $_GET, $_SESSION`
`$_FILES`	Dateien, per HTTP POST hochgeladen
`$_COOKIE`	innerhalb eines Cookies "im Browser" gespeichert
`$GLOBALS`	alle globalen Variablen
`$_SERVER / $_ENV`	von Webserver / Umgebung gesetzt; zusätzlich diverse CGI-Environment-Variablen (→ `phpinfo()`)

- Konstanten werden mit skalaren Werten angelegt und können nicht verändert werden:

```
2 define('MEINE_KONSTANTE', "Hello World");
3 echo MEINE_KONSTANTE.'<br>';
7 echo "Zeile ".__LINE__." in ".__FILE__."<br>"; //vordefiniert
```

01-constants.php

Geltungsbereich

- globale Gültigkeit im Kontext, in dem die Variable definiert wurde
  - d.h. innerhalb des Skripts selbst incl. eingebundener Skripte (s. dort)
- lokale Gültigkeit innerhalb von Funktionen; Zugriff auf globale Variablen mit
  - keyword `global`
  - das superglobale Array `$GLOBALS`
- Weitergabe von Variablen an den nächsten Skriptaufruf:
  - speichern in `$_SESSION` (serverseitig)
  - speichern in `$_COOKIE` (clientseitig)
  - Übergabe per HTML-Formular + HTTP POST (Zugriff über `$_POST`)
  - Übergabe in der URL: skript.php?name=Peter&id=3 (Zugriff über `$_GET`)

```
10 // Aufruf: 01-variable-scope.php?name=Peter
12   $name = $_GET['name'];
13   echo "$name, Du hast mich gerufen!<br/>\n";
14   //Speichern der Variablen in einer Session
15   $_SESSION['name']  = $name;
```

01-variable-scope.php

Operatoren

unäre, binäre und ternäre Operatoren:

arithmetisch	`+, -, *, /, % (modulo)`	
Zuweisung	`=`	`$a = 3` hat den Wert 3!
kombinierte Zuweisung	`.=, +=, -=, *=, /=, ...`	`$a += 5` analog zu `$a = $a + 5`
bitweise	`&, |, ^, ~, >>, <<`	
Vergleich	`<, <=, <=, >, ==, !=, <>` (Wertvergleich) `===, !==` (Identität)	
Fehlerkontrolle	`@`	unterdrückt Fehlermeldungen
Programm-ausführung	` `` `	führt Kommandozeilenbefehl aus
In-/Dekrement	`$v++, $v--, ++$v, --$v`	auch auf Strings
logisch	`and, or, xor, !, &&, ||`	
String	`.`	Konkatenation
Typen	`instanceof`	Instanz einer Klasse

Kontrollstrukturen

- if, else, elseif

```
if (<expr>) {
  <statements>
}
elseif (<expr>) {
  <statements>
}
else {
  <statements>
}
```

- switch / case

```
switch ($var) {
  case $value1:
    <statements>
    [break;]
  case $value1:
    <statements>
    [break;]
  ...
  default:
    <statements>
    [break;]
}
```

- Bedingter Ausdruck (ternärer Operator)

```
(<expr>) ? <statement_true> : <statement_false>
```

Kontrollstrukturen - 2

- while / do - while

```
while (<expr>) {
  <statements>
}
// oder
do {
  <statements>
} while (<expr>)
```

- foreach
  (speziell für Arrays)

```
foreach ($arr as [$key =>] $val {
  <statements>
}
```

- for

```
for (<expr_init>; <expr_test>; <expr_exec>) {
  <statements>
}
```

- break / continue
  `break` bricht Schleife ab, `continue` überspringt restlichen Code im Schleifen-rumpf

Funktionen

Definition eigener Funktionen (ab PHP4 an beliebiger Stelle):

```
<?php
function addSomething($i1, $i2=5) {
  return $i1 + $i2;
}
echo addSomething(4); // 9
echo addSomething(1,4); // 5
?>
```

- beliebige Rückgabewerte, default: keiner
- immer global gültig, Namen nicht case-sensitiv, nur einmal definierbar
- ab PHP4: variable Anzahl von Argumenten möglich; default-Werte:
  `function foo($arg1 = 0, ...`
- als `static` deklarierte Funktionsvariablen behalten ihren Wert über mehrere Aufrufe
- > 4000 vordefinierte Funktionen:
  http://www.php.net/manual/en/funcref.php ..die PHP-Bibel ;-)

Einbinden von Skripten / Modulen

Sehr sinnvoll: grosse Skripte auf viele Dateien verteilen; Einbinden mit `include(), require()`:

- schlägt das Einbinden fehl, bricht `require` ab, `include` warnt nur
- `include_once()` und `require_once()` binden eine Datei auch bei mehrmaligem Aufruf nur einmal ein
- PHP-Code innerhalb eingebundener Dateien muss mit `<?php .. ?>` Tags umschlossen sein

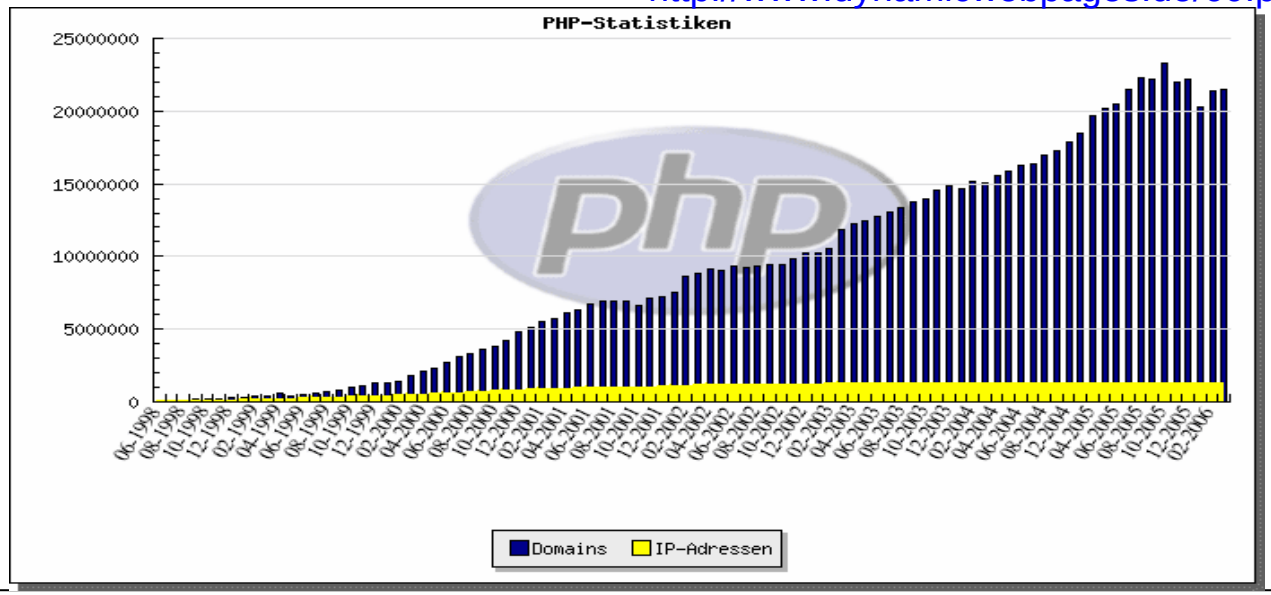
```
2 // praktisch: den Pfad global setzen (> PHP 4.3)
3 set_include_path("/php/include/");
4 $name = "Juergen";
5 //functions.inc.php hat Zugriff auf $name:
6 include_once("functions.inc.php");
7 // bricht bei Fehler ab:
8 require_once("unbekannt.html");
```

01-include.php

kurze Geschichte

- Frühling 1995: PHP / FI 1.0 (Rasmus Lerdorf, "Personal Homepage Tools, Form Interface")
- 1997: PHP 3.0 (echter Parser, Interpreter, ...)
- Frühling 2000: PHP 4.0 (neuer Sprachkern "Zend", viele Funktionen...)
  - Mai 2003: ca. 13 Millionen Domains benutzen PHP
- Juli 2004: PHP 5.0 (Zend 2.0, neues Objekt-Modell, ...)
  - Juli 2005: 21 Millionen Domains
- 24. November 2005: PHP 5.1

http://www.dynamicwebpages.de/60.php-statistiken.php



Übungsaufgaben

1. Lege in Deinem Homeverzeichnis ein Verzeichnis `sc_html` mit den Berechtigungen `chmod 755` an; teste darin ein hello-world-script und lass Dir mit phpinfo() Informationen über die Installation anzeigen; Zugriff: http://iapetus.informatik.uni-freiburg.de/~<login>/helloworld.php
2. Generiere eine Webseite, bei der der Benutzer durch Klicken auf Links ("blau, grün, rot) die Hintergrundfarbe der Seite wechseln kann.
3. Schreibe ein "Online-Notizbuch" (`notes.php`) für mehrere User:
   - der Username soll per URL übergeben werden: notes.php?user=Hans
   - Wird kein Usernamen übergeben, soll das Skript mit einer Fehlermeldung abbrechen (Tip: Funktion `die()`);
   - für jeden User wird ein Textfile `<username>.txt` angelegt (siehe dazu die Filesystem-Funktionen, http://www.php.net/manual/en/function.fopen.php), das für den jeweiligen User eingebunden wird
   - neue Notizen sollen über ein HTML-Formular eingegeben werden `<textarea>...`
   - Zusatz: für jeden User soll das Datum der letzten Änderung seiner Notiz angezeigt werden (Tip: `filemtime(), date()`)
4. Welchen Datentyp haben die Variablen, die per URL weitergegeben werden?

Kontrollfragen

- Was sind skalare Datentypen? Was für skalare Datentypen gibt es in PHP?
- Was ist der Unterschied zwischen server- und clientseitigem Scripting?
- Wie kann man Kommentare in PHP verfassen?
- Welche Funktionen kann der +-Operator haben? Auf welche Datentypen kann man Ihn anwenden?
- Wozu ist das foreach-Konstrukt?
- Wo liegt in folgendem Algorithmus der Fehler? Gibt es mehr als einen Fehler?

```
beispiel1.php:
--------------
<?php
$user = 'John Doe';
include('beispiel2.php');
?>
beispiel2.php:
--------------
echo 'Hallo $user';
```

Quellen, Resourcen

PHP-Schulung von Johann-Peter Hartmann und Ulf Wendel
http://www.php-center.de/phpschulung
PHP-Einführung von Thomas Nunninger, Softwarepraktikum SS 2004
http://www.ks.uni-freiburg.de/download/papers/general/swprakt_doku.pdf
zentrale Anlaufstelle:
http://www.php.net
Handbuch, Sprach- und Funktionsreferenz:
http://www.php.net/manual/en
Wikipedia über PHP:
http://en.wikipedia.org/wiki/Php