Einführung in die PHP-Programmierung

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

7. August 2006

Inhaltsverzeichnis

Tabellenverzeichnis

List of Algorithms

1. Session: PHP-Grundlagen

1.1 Einführung

Diese Einführung in PHP soll dazu dienen, die grundlegende Arbeitsweise mit PHP zu erlernen, mit dem Zweck selbst dynamische Webseiten mit PHP erstellen zu können. Das vorliegende Skript deckt sicherlich nicht die gesamte Funktionsvielfalt von PHP ab, gibt aber einem Anfänger die Möglichkeit grundlegende Strukturen wie Variablen, Operatoren oder Kontrollstrukturen und deren handhabung mit PHP zu erlernen ohne dabei von der Vielfalt von PHP direkt erschlagen zu werden. Diese Einführung basiert auf den Erfahrungen der Autoren, dem Skript *PHP Einführung* von Thomas Nunninger (welches wiederum auf der PHP-Schulung von Ulf Wendel und Peter Hartmann basiert), der FAQ der Newsgruppen *de.comp.lang.php.** [1] und der offiziellen Dokumentation[2]. Dem Leser wird empfohlen nach der Teilnahme am PHP Kurs im Rahmen des Sommercampus 2006 an der Albert-Ludwigs-Universität Freiburg sich mit Hilfe des Online-Handbuchs von PHP weiter mit dem Thema auseinanderzusetzen. Hier sollte man allerdings auf die englische Version zurückgreifen, da die deutsche Übersetzung des Handbuchs der englischen Version deutlich hinterherhinkt. Bei vielen Problemen hilft die oben genannte FAQ und das Handbuch (insbesondere die User-Comments) weiter, ansonsten kann man sich nach intensiver Lektüre dieser beiden und der Suche nach einer Problemlösung mittels Google auch an die Newsgroups *de.comp.lang.php.** wenden. Die Webseite SelfPHP ist (im Gegenteil zu Selfhtml[3]) allerdings überhaupt nicht zu empfehlen, da diese hoffnungslos veraltet ist.

1.2 Was ist PHP

Die Abkürzung PHP steht offiziell für „PHP: Hypertext Preprocessor“. Dies ist eine rekursive Abkürzung im Stile des GNU-Projektes (GNU's Not Unix)[4]. PHP ist eine serverseitige Scriptsprache[5] zur dynamischen Erstellung von

[1] `http://www.php-faq.de/`
[2] `http://www.php.net`
[3] `http://de.selfhtml.org/`
[4] `http://www.gnu.org/`
[5] `http://de.wikipedia.org/wiki/Scriptsprache`

Webseiten. Die Anweisungen der Sprache sind dabei in den HTML-Code einer Webseite eingebettet, d. h. jede HTML-Seite ist auch ein gültiges PHP-Programm. Die Syntax von PHP ist ähnlich wie die von C/C++, Java oder JavaScript. Die Sprache zeichnet sich vor allen Dingen durch ihre leichte Er-

Algorithm 1 Ein einfaches Hello-World Programm in PHP

```
1 <?php
2 echo 'Hello World!';
3 ?>
```

lernbarkeit, ihre ausgezeichneten Datenbankanbindungen und Internet-Protokolleinbindungen und die Unterstützung zahlreicher weiterer Funktionsbibliotheken aus. PHP stellt so für den Web-Entwickler das ideale Werkzeug zur Erstellung von dynamischen Inhalten dar. PHP ist freie Software im Sinne der Debian Free Software Guidelines (DFSG). Quelltext und Binaries des PHP-Interpreters sind frei erhältlich und können für alle kommerziellen und nichtkommerziellen Zwecke eingesetzt werden; jeder kann den PHP-Quelltext weiterentwickeln und die Änderungen an das PHP-Projekt zurückfließen lassen. Der genaue Lizenztext ist in der Datei LICENSE enthalten, die Bestandteil der PHP-Distribution ist. PHP läuft auf allen gängigen UNIX- und Linux-Versionen, auf den verschiedenen Windows-Versionen (Windows 95/98/ME-/NT/2000/XP) sowie auf MacOS X, OS/2 und einigen anderen Betriebssystemen. Als CGI-Programm kann PHP mit jedem Webserver zusammenarbeiten. Für einige Webserver, allen voran Apache, stehen auch Modulversionen zur Verfügung, die sehr viel effizienter ausgeführt werden. Seit PHP in der Version 4.3.0 gibt es auch ein Command Line Interface (CLI), welches PHP für kommandozeilenbasierende Applikationen attraktiv macht. In Kombination mit PHP-GTK sind auch graphische Oberflächen möglich[6].

1.3 Grundlegendes zu dynamischen Webseiten

Unter dynamischen Webseiten versteht man Webseiten, die erst im Augenblick des Abrufes zusammengesetzt werden. Der Unterschied zu normalen Seiten, die nur in HTML geschrieben sind, ist dass dynamische Internetseiten bei jedem Besucher (teilweise) neu generiert werden. In diese neue Internetseite können dann jegliche Datenquellen und andere Informationen einfliessen, wie z.B Eingaben von Benutzern. Beispiele für einfache dynamische Internetseiten sind z.B Besucherzähler oder Gästebücher. Komplexere wären Internetforen, E-Commerce Systeme oder die Anbindung von Warenwirtschaftssystemen von verschiedenen Firmen. In Abbildung 1.1 sieht man schematisch den Ablauf

[6] Inwiefern man PHP für nicht webbasierte Anwendungen verwenden will bleibt jedem selbst überlassen.

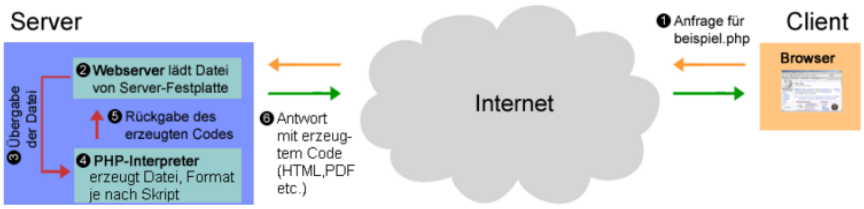

Abbildung 1.1: Funktionsweise einer Anfrage (Quelle: Wikipedia)

einer Anfrage eines Benutzers, der vom Server die Seite „beispiel.php“ anfordert. Der Client (also der Web-Browser) schickt eine Anfrage an den Server auf dem die PHP-Seite gespeichert ist mit einer Anfrage nach beispiel.php. Am Server wird diese Anfrage von einem speziellen Programm, dem HTTP-Server (z.B. Apache) erhalten. Wenn nun eine statische HTML-Seite (.html) gewünscht ist, ließt der Server sie direkt aus und schickt sie zurück an den Client. Ist aber eine PHP-Seite gefragt, leitet er die Anfrage an den PHP-Präprozessor (dem Programm das sozusagen hinter der Programmiersprache PHP steht) weiter. Ob es sich bei einer Anfrage um eine statische HTML-Seite oder eine PHP-Seite handelt, entscheidet der Webserver andhand der Dateinamenerweiterung. Standard sind .html für statische Seiten und .php für PHP-Seiten. Diese Einstellung kann aber auch beliebig verändert werden, so dass z.B. auch PHP-Code in .html-Dateien geparst wird. Der PHP-Präprozessor liest nun den PHP-Quelltext der Seite, parst ihn und generiert daraus die HTML-Seite, die an den HTTP-Server und schlussendlich zum Client weitergeleitet wird. Der Browser stellt nun das zurückgegebe HTML (hier ist also kein PHP-Code mehr enthalten) dar und führt eventuell enthaltenen JavaScript Code aus.

1.4 Einbindung in HTML

PHP-Code kann auf einfachste Weise in HTML-Seiten eingebettet werden. Ist auf dem Webserver PHP aktiviert, so genügt es eine bereits vorhandene HTML-Seite zu kopieren und diese geringfügig zu verändern. Der Code-Ausschnitt 2 zeigt ein Beispiel hierfür. Hier wurde einfach in den HTML-Source eine PHP-Anweisung (der Teil der von <?php und ?> umschlossen ist) eingebettet. Speichert man diese Datei als helloworld-html.php und kopiert sie in ein Verzeichnis auf einem Webserver für das die Ausführung von PHP aktiviert ist, so wird der Teil des Codes zwischen <?php und ?> geparst und serverseitig ausgeführt. Als Ergebnis erhält man im augegebenen HTML-Code dann zusätzlich „Hello World!“. Das hätte man natürlich auch direkt in den HTML-Code schreiben können, allerdings ist das lediglich ein einfaches Beispiel.

Algorithm 2 Ein einfaches Hello-World Programm in PHP eingebettet in HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional
    //EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Meine erste PHP-Seite</title>
</head>
<body>
<h1>PHP-Test</h1>
<?php
/*
* Dies gibt "Hello World!" aus.
*/
echo 'Hello World!';
?>
</body>
</html>
```

1.5 Installation und Konfiguration eines Basissystems zum arbeiten mit PHP

1.5.1 Apache und PHP installieren

Im Folgenden Abschnitt wird kurz beschrieben wie man Apache und PHP (unter Windows) installiert. Benutzer anderer Betriebssysteme folgen bitte der „Experteninstallation“ und passen sie entsprechend Ihrem Betriebssystem an.

1.5.1.1 Einfache Variante: XAMPP

Die einfachste Variante unter Windows ein laufendes System mit Apache und PHP zu installieren ist die Verwendung von XAMPP[7]. XAMPP ist eine Distribution von Apache, MySQL, PHP und Perl, die es ermöglicht diese Programme auf sehr einfache Weise zu installieren. Die Installation von XAMPP ist auf deren Webseite sehr ausführlich beschrieben, weshalb wir an dieser Stelle nicht genauer darauf eingehen.

1.5.1.2 Experteninstallation

Für diejenigen, die sich zutrauen PHP und Apache selbst zu installieren und konfigurieren empfehlen wir sich an folgenden Stichpunkten zu orientieren:

[7] http://www.apachefriends.org/de/xampp.html

- Apache & PHP downloaden[8] und Apache installieren. In der Dokumentation des Apache httpd (Aktuelle Version: 2.2 (7. August 2006)) wird ausführlich beschrieben wir man diesen installiert.
- Dann muss man php installieren. Dafür sollte man das heruntergeladene zip-File nach C:\php\ (oder einem anderen Pfad) entpacken, diesen Pfad sollte man dann optimalerweise seinem Pfad hinzufügen. Dies erreicht man bei Windows XP unter Systemeigenschaften ($[Win]+[Untbr/Pause]$ drücken) ▷ Erweitert ▷ Umgebungsvariablen, dort der Variable PATH den String „C:\php\;“ voranstellen.
- Als nächsten Schritt sollte man die Datei php.ini-recommended nach php.ini kopieren und eventuelle Anpassungen vornehmen (es funktioniert auch ohne).
- Als nächstes muss man dem Apache beibringen PHP-Dateien auszuführen. Dazu muss man die Konfigurationsdatei des Apache (httpd.conf) öffnen und ans Ende folgende Zeilen hinzufügen:

Algorithm 3 Konfiguration des Apache für PHP

```
1 LoadModule php5_module "C:/php/php5apache2.dll"
2 AddType application/x-httpd-php .php
3 PHPIniDir "C:/php"
```

1.5.2 Eclipse

Wir empfehlen für die Entwicklung Eclipse zu installieren. Eclipse hat einige nette Features, wie z.B. die Integration mit Subversion oder CVS, automatische Code-Ergänzung, etc. Eclipse läuft ausserdem auf (fast) allen Plattformen und ist evtl. einigen Personen von der Entwicklung mit Java her schon bekannt. Empfehlenswert ist es bei Eclipse gleich die Web Tools Platform[9] mitzuinstallieren, hier gibt es das ganze auch als Basispaket gleich inklusive Eclipse, dann spart man sich ein bisschen Installationsarbeit. Unter Windows würde man sich also das *wtp-all-in-one-sdk-R-1.5.0-200606281455-win32.zip*[10] Zip-File herunterladen und installieren. Un*xer können sich mit Hilfe Ihres Paketmanagers Eclipse installieren und das Plugin über Eclipses Pluginverwaltung nachinstallieren, oder das .tar.gz verwenden.

[8] Apache: `http://httpd.apache.org/`, PHP5.x zip package von: `http://www.php.net/`
[9] `http://www.eclipse.org/webtools/`
[10] Stand 2006-07-31

1.5.3 PHPEclipse

PHPEclipse ist ein Plugin für Eclipse, das Eclipse um einige nette Features erweitert. Dazu gehören unter anderem:

- PHP, HTML, XML, CSS und Smarty Syntax Highlighting
- Code completion
- Web Browser Preview

Um PHPEclipse zu installieren kann man Eclipses Plugin Manager verwenden, folgende Schritte[11] sind dabei notwendig:

- Help ▷ Software Updates ▷ Find and Install
- (x) search for new features to install
- „New Remote Site“ anklicken
- Als Name z.B. *PHPEclipse* eingeben und als URL: `http://phpeclipse.sourceforge.net/update/releases`
- „Finish“ anklicken
- $[x]$ PHPEclipse anklicken
- Next, next, next usw. bis alles installiert ist ;-).

1.6 Grundlegende Strukturen in PHP

Bevor wir zu den grundlegenden Strukturen in PHP übergehen möchte ich allen ans herz legen das Error-Reporting von PHP sehr hochzuschrauben um möglichst fehlerfrei zu programmieren. Dier erreicht man indem man folgende zeile an den Anfang eines PHP-Skripts schreibt:

Algorithm 4 Error-Reporting anpassen

```
1 <?php
2 error_reporting(E_ALL);
3 ?>
```

Damit werden z.B. Fehler wie fehlende Initialisierung von Variablen etc. angemängelt was durchaus den ein oder anderen Programmierfehler verhindert. Ist eine Anwendung fertig entwickelt und im produktiven Einsatz, sollte man das Errror-Reporting-Level wieder heruntersetzen (z.B. mit

[11] Quelle: `http://www.plog4u.org/index.php/Using_PHPEclipse_:_Installation_:_Installing_PHPEclipse`

error_reporting(E_ERROR)), um die Benutzer vor PHP-Fehlermeldungen zu verschonen und möglichst wenig Informationen über den Quellcode preiszugeben.

1.6.1 Kommentare

In PHP gibt es mehrere Möglichkeiten Kommentare zu verfassen, diese können ein- oder mehrzeilig sein oder dazu dienen den Code so zu kommentieren, dass Kommentare später automatisch verarbeitet werden können.

Algorithm 5 Einzeilige Kommentare

```
1 <?php
2 echo 'Hallo Welt!'; // Dies ist ein einzeiliger Kommentar
3 echo 'Hallo Welt!'; # Dies ist auch ein einzeiliger
      Kommentar
4 ?>
```

Algorithm 6 Mehrzeilige Kommentare

```
1 <?php
2 /*
3 Dies ist ein
4 mehrzeiliger
5 Kommentar
6 */
7 echo 'Hallo Welt!';
8 ?>
```

Algorithm 7 Doc-Kommentare

```
1 <?php
2 /**
3 * Dieser Kommentar beschreibt die funktionsweise des
      nachfolgenden Codes
4 * Es wird Hallo Welt! ausgegeben.
5 */
6 echo 'Hallo Welt!';
7 ?>
```

Doc-Kommentare sind kein offizieller bestandteil der PHP-Syntax, sie sollten aber für eine spätere automatische API-Dokumentation (z.b. mit PHPDo-

cumentor) verwendet werden.

1.6.2 Variablen

Variablen werden in PHP dargestellt durch ein Dollar-Zeichen ($) gefolgt vom Namen der Variablen. Bei Variablen-Namen wird zwischen Groß- und Kleinschreibung unterschieden (case-sensitive: $a ist also eine andere Variable als $A). Variablen-Namen werden in PHP nach den gleichen Regeln wie andere Bezeichner erstellt. Ein gültiger Variablen-Name beginnt mit einem Buchstaben oder einem Unterstrich („_“), gefolgt von einer beliebigen Anzahl von Buchstaben, Zahlen oder Unterstrichen. Variablen müssen in PHP nicht initialisiert werden, es gehört aber zum guten Stil, das zu tun.

Algorithm 8 Variablenzuweisungen

```
1 <?php
2 $var = 7;
3 $var = 2.1;
4 $var = 'foo';
5 ?>
```

Mehreren Variablen können auch gleichzeitig ein Wert zugewiesen werden:

Algorithm 9 Gleichzeitige Zuweisung

```
1 <?php
2 $a = $b = $c = 42;
3 ?>
```

1.6.2.1 HereDoc Syntax

In PHP kann auch die HereDoc-Syntax verwendet werden:

Algorithm 10 Heredoc-Syntax

```
1 <?php
2 $a = <<<EOS
3 Dies ist ein String
4 mit Zeilenumbruch
5 EOS;
6 ?>
```

Hierbei ist zu beachten, dass der endende Bezeichner (hier: *EOS*) alleine auf einer Zeile stehen muss.

1.6.2.2 Datentypen

In PHP gibt es verschiedene Arten von Datentypen, dazu gehören skalare Datentypen, zusammengesetzte Typen und spezielle Datentypen wie Ressourcen.

skalare Datentypen Zu den skalaren (auch primitven) Datentypen in PHP gehören Integer, Strings oder boolesche Variablen.

Algorithm 11 Skalare Datentypen

```
1 <?php
2 // Deklaration eines Integer-Wertes
3 $myint = 123;
4 // Deklaration eines Strings
5 $mystring = "Hallo Welt!";
6 // Deklaration und Zuweisung einer booleschen Variable
7 $mybool = true;
8 $mybool = false;
9 ?>
```

Wenn man eine Variable nach Boolean castet, so liefert der Ausdruck bis auf wenige Ausnahmen immer true:

Algorithm 12 Typcasting nach Boolean

```php
<?php
$str = "123";
if ($str) {
  echo 'bool'; // bool wird ausgegeben
}
$str = '';
if ($str) {
  echo 'bool'; // bool wird NICHT! ausgegeben
}

$int = 0;
if ($int) {
  echo 'bool'; // bool wird NICHT! ausgegeben
}
$float = 0.0
if ($float) {
  echo 'bool'; // bool wird NICHT! ausgegeben
}
?>
```

Die Ausnahmen im Detail sind:

- Der Integer Wert 0
- Der Float Wert 0.0
- Der leere String '' oder der String '0'
- Ein leeres Array
- Ein Objekt ohne Membervariablen
- NULL

Um zu erfahren welchen Typ eine Variable besitzt, kann man *gettype* oder *var_dump*, sowie *print_r* verwenden. Bei **Strings** gibt es in PHP einige Besonderheiten: Im Gegensatz zu Sprachen wie Java, kann man Strings sowohl in einfache '' als auch in doppelte Anführungszeichen „“ setzen. Dabei ist wichtig, dass innerhalb von doppelten Anführungszeichen Ausdrücke interpretiert werden, innerhalb von einfachen nicht.

Algorithm 13 Strings

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

zusammengesetzte Datentypen Zu den zusammengesetzten Datentypen gehören Arrays und Objekte. **Arrays** sind Datenstrukturen bei denen über einen Schlüssel auf das entsprechende objekt bzw. den entsprechenden Wert zugegriffen wird. Arrays sind in PHP beliebig erweiterbar und akzeptieren als Schlüssel entweder einen Integer oder einen String. Diese zwei Arten von Schlüsseln kann man auch mischen. Es sind auch mehrdimensionale Arrays möglich und bei Bedarf generiert PHP die Schlüssel für neu eingefügte Werte auch selbst. Für den Zugriff auf Arrays gibt es verschiedene wichtige Konstrukte wie z.B. *foreach()*, *array_keys()*, *array_values*.

Algorithm 14 Arrays

```
<?php
$arr = array('123','345');
$arr = array(1,2,3,4);
$arr = array(
  1 => 'mystr1',
  2 => 'mystr2',
  "str" => "iHaveAStringKey"
);
// Mehrdimensionale Arrays
$arr2[1][3] = 'test';
// Automatische Schlüsselerzeugung
$arr3 = new array();
$arr3[] = 4;
$arr3[] = 5;
echo $arr3[1];
?>
```

Objekte sind Instanzen von Klassen, darauf wird im Abschnitt 2.2 genauer eingegangen.

spezielle Datentypen In PHP gibt spezielle Datentypen, sogenannte Resourcen. Das können z.b. geöffnete Dateien oder Datenbankverbindungen sein. Da

Ressourcen nur von speziellen Funktionen erstellt werden (z.B. *fopen*) kann man auch andere Variablen nicht zu diesem Typ casten.

Algorithm 15 Resourcen

```
1 <?php
2 $res = mysql_connect("localhost", "username", "pass");
3 // $res ist jetzt eine resource (wenn der connect geklappt
      hat)
4 print $result; //Resource ID#1
5 ?>
```

1.6.2.3 vordefinierte Variablen

IN PHP gibt es eine Reihe vordefinierter Variablen:

- $_POST - POST-Variablen
- $_GET - GET-Variablen
- $_SESSION hier kann man Session-Variablen speichern
- $_REQUEST vereinigt $_POST, $_GET und $_SESSION
- $_GLOBALS globale Variablen
- $_COOKIE enthält Variablen, die auf der Client-Seite in Cookies gespeichert werden
- $_FILES hier sind Dateien, die per POST hochgeladen wurden, enthalten
- $_SERVER vom Webserver gesetzte Variablen
- $_ENV Umgebungsvariablen

In $_POST landen alle Variablen die dem Script via HTTP Post übergeben wurden, in $_GET entsprechend alle GET-Variablen. Ruft man ein Script also z.b. so auf: *script.php?var=123*, so kann man ueber $_GET['var'] auf die übergebene Variable var zugreifen. $_GLOBALS enthält globale Variablen, diese sind ueberall verfügbar. D.h. man kann innerhalb einer Funktion auf eine Variablen zugreifen die im globalen Kontext deklariert wurde. Die Funktion *phpinfo()* listet all diese Variablen auf und gibt zusätzliche Informationen über die PHP-Version. Mittels *print_r($_POST)* kann man den gesamten Inhalt des POST-Arrays ausgeben. Äquivalent gilt diese natuerlich auch für die anderen vordefinierten Variablen.

1.6.2.4 Konstanten

Konstanten kann man in PHP auch anlegen, diese werden mit skalaren Werten angelegt und können dann nicht mehr verändert werden. Mit *is_defined()* kann man überprüfen ob eine Konstante schon existiert. Es gibt auch bereits vordefinierte Konstanten wie *__file__* und *__line__*.

Algorithm 16 Konstanten

```
1 <?php
2 define('MY_CONST',123);
3 echo MY_CONST;
4 echo 'Dies wird in Zeile ',__line__,' im Script ',__file__
    ,' ausgegeben.';
5 ?>
```

1.6.2.5 Geltungsbereich

In PHP unterscheided man verschiedene Geltungsbereiche von Variablen:

- globale Gültigkeit: Innerhalb des „globalen Kontextes“ deklarierte Variablen sind überall zugreifbar. Innerhalb von Funktionen können diese mittels *global $var;* in den aktuellen Kontext gezogen werden, dann kann man einfach mittels *echo $var;* darauf zugreifen. Eine einfachere und (imho) bessere Möglichkeit stellt der Zugriff mittels des superglobalen Arrays $_GLOBALS dar: *echo $_GLOBALS['var'];*.
- lokale Gültigkeit: Die innerhalb von Funktionen deklarierten Variablen verlieren ausserhalb der Funktion Ihre Gültigkeit.
- Die Weitergabe von Variablen an ein anderes Skript ist entweder über GET oder POST möglich, oder man speichert Variablen in Cookies (das sollten nur Ids sein) oder als Session-Variablen: *_SESSION['var'] = 124;*. Für die Verwendung von Session-Variablen muss allerdings zuerst eine session mit *session_start()* erzeugt werden.

1.6.3 Operatoren

Es gibt drei Arten von Opratoren. Als erstes gibt es den unären Operator, der nur mit einem Wert umgehen kann, zum Beispiel ! (der Verneinungsoperator) oder ++ (der Inkrementoperator). Die zweite Gruppe sind die sogenannten binären Operatoren; diese Gruppe enthält die meisten Operatoren, die PHP unterstützt. Die dritte Gruppe bildet der ternäre Operator *?:*. Dieser sollte eher benutzt werden um abhängig von einem dritten Ausdruck eine Auswahl zwischen zwei Ausdrücken zu treffen, als zwischen zwei Sätzen oder Pfaden der Programmausführung zu wählen. Den ternären Operator kann man allerdings

auch als verkürzte Schreibweise der If-Bedingung verwenden. Tabelle 1.1 gibt eine Übersicht über verschiedene Operatoren in PHP.

Arithmetisch	+, -, *, /, % (modulo)	
Zuweisung	=	
Kombinierte Zuweisung	+=, -=, *=, /=	*$a+=5;* ...analog zu *$a=$a+5;*
Bitoperatoren	&, |, ~, », «	
Vergleichsoperatoren	<, <=, >, >=, ==, !=, <>, (Wertvergleiche) ===, !== (Identitätsvergleiche)	
Fehlerkontrolle	@	unterdrückt Fehlermeldungen
Programmausführung	“	(Backticks)
Inkrement bzw. Dekrement	++$v, –$v, $v–, $v++	
Logische Operatoren	and, or, xor, !, &&, ||	
Stringverknüpfungen	.	Konkatenation von zwei Strings: *$str = $s1 . $s2;*
Typen	*instanceof*	Instanz einer Klasse

Tabelle 1.1: Operatoren

1.6.4 Kontrollstrukturen

In PHP gibt es verschiedene Kontrollstrukturen die dazu dienen verschiedene Befehle mehrfach hintereinander auszuführen, im Code zu verzweigen oder bestimmte Datenstrukturen wie z.B. Arrays zu durchlaufen. *If*, *Elseif* und *Else* dienen dazu Code nur unter bestimmten Bedingungen auszuführen.

Algorithm 17 If-Bedingungen

```
1  <?php
2  $var=1;
3  if (1== $var) {
4  echo 1;
5  } elseif (2 == $var) {
6  echo 2;
7  } else {
8  echo 'Was anderes';
9  }
10 ?>
```

Der in Algorithmus 17 gezeigte Code kann mittels des switch-Statements besser formuliert werden:

Algorithm 18 Switch-Statement

```
1  <?php
2  switch ($var) {
3    case 1: {
4      echo 1;
5      break;
6    }
7    case 2: {
8      echo 2;
9    break;
10   }
11   default: {
12     echo 'Was anderes';
13     break;
14   }
15 }
16 ?>
```

Hierbei ist zu beachten, dass man jeden Ausdruck innerhalb des switch--case-Statements verwenden kann, nicht nur Integer-Werte. Das *break*-Statement ist wichtig, damit die folgenden Blöcke nicht mehr ausgeführt werden. In Schleifen kann auch der Befehl *continue* verwendet werden, dieser überspringt dann die weiteren Befehle im Schleifenrumpf und geht zur nächsten Iteration über. Des Weiteren gibt es noch verschiedene Schleifen in PHP, die while Schleife, die do-while-Schleife, eine for-Schleife sowie das foreach Konstrukt.

Algorithm 19 Schleifen

```php
<?php
// while-schleife
while (<expr>) {
  // Befehle
}

// do-while-schleife
do {
  // Befehle
} while (<expr>)

// for-schleife
for ($i=0; $i<10; $i++) {
  //Befehle
}

// foreach
foreach ($array as $key => $value) {
  echo '$array[',$key,'] contains ',$value;
}
?>
```

1.6.5 Funktionen

1.6.6 Kontrollfragen

Funktionen können in PHP an jeder Stelle im Code definiert werden. Der Rückgabewert muss nicht typisiert werden und es können beliebig viele Variablen übergeben werden. Algorithmus 20 zeigt eine Funktionsdefinition.

Algorithm 20 Funktionen

```php
<?php
function outputSomething($str = 'defaultValue') {
  echo 'The String is: ',$str;
}

function addSomething($i1, $i2=5) {
  return $i1 + $i2;
}

echo addSomething(4); // 9
echo addSomething(1,4); // 5
?>
```

1.6.7 Einbinden von weiteren Skript-Dateien

Mittels *include* und *require* bzw. *include_once* und *require_once* können weitere PHP-Skripte eingebunden werden. Include/Require bindet eine Datei so ein, als würde der darinstehende Code an dieser Stelle stehen. Der Unterschied zwischen *include* und *require* ist, dass *require* die Code-Verarbeitung abbricht wenn das einbinden fehlschlägt, bei *include* geht die Verarbeitung weiter. Die *_once*-Funktionen binden die jeweilige Datei nur einmal ein, wenn diese Datei also bereits einmal eingebunden wurde, so wird sie nicht nochmals eingebunden. Beim Einbinden von weiteren Dateien ist darauf zu achten, dass PHP-Code von *<?php* und *?>* eingeschlossen wird, sonst wird eine Ausgabe erzeugt (Das ist insbesondere beim Verwenden von *header()* wichtig.)

1.7 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 Algorithmus 21 der Fehler? Gibt es mehr als einen Fehler?

Algorithm 21 Kontrollfragen - Beispiel

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

1.8 Übungsaufgaben

1. Lass Dir mit *phpinfo()* Informationen über Deine PHP-Installation anzeigen. Welche Informationen darin wären für eventuelle Angreifer / Hacker interessant?

2. Generiere eine Webseite, bei der der Benutzer durch Klicken auf Links ("blau, grün, rot") die Hintergrundfarbe der Seite wechseln kann.

3. Schreibe ein Önline-Notizbuch"(`notes.php`) für mehrere User:

   - der Username soll per URL übergeben werden: `notes.php?user=Hans`
   - Wird kein Username 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?

2. Session: PHP-Grundlagen / Objektorientierung

2.1 ausgewählte nützliche Funktionen

2.1.1 Datum

PHP bietet einige interessante Funktionen mit denen Daten manipuliert und ausgegeben werden können. Dazu gehören unter anderem folgende:

- string date (string format [, int timestamp]) Formatiert einen Zeitstempel (falls angegeben) oder die aktuelle Zeit im angegebenen Format.
- int time (void) Gibt einen Timestamp der aktuellen Zeit zurück.
- mixed microtime ([bool get_as_float]) Gibt den aktuellen Unix-Zeitstempel inklusive Mikrosekunden zurück.

Seit PHP5 gibt es auch einige vordefinierte Datumskonstanten:

- DATE_ATOM Bsp.: 2005-08-15T15:52:01+00:00
- DATE_COOKIE Bsp.: Monday, 15-Aug-05 15:52:01 UTC
- DATE_W3C Bsp.: 2005-08-15T15:52:01+00:00
- etc.

Eine komplette Liste ist unter
`http://www.php.net/manual/en/ref.datetime.php` zu finden.

2.1.2 Dateisystem

Die Funktion *resource fopen (string filename, string mode [, bool use_include_path [, resource zcontext]])* wird verwendet um in PHP Dateien zu öffnen zur weiteren Verarbeitung. Der erste Parameter gibt die zu benutzende Datei an und der Parameter mode gibt an wie diese Datei geöffnet werden soll. Ist mode „r“, so wird die Datei im nur-lesen-modus geöffnet. Ist mode „r+“, so kann gelesen und geschrieben werden, der Dateizeiger zeigt dabei an den Anfang der

Datei. Wird „w“ als mode verwendet, so wird die Datei zum Schreiben geöffnet, der Dateizeiger zeigt auf den Anfang der Datei und sämtlicher Dateiinhalt wird vor dem Schreiben gelöscht. „a“ öffnet die Datei zum Schreiben, der Dateizeiger ist dabei ans Ende der Datei gesetzt (append). Weitere modes sind „a+“, „x“ und „x+“, siehe `http://www.php.net/manual/en/function.fopen.php`.

Mit den Funktionen *is_dir*, *is_file* und *is_link* kann der Typ einer datei überprüft werden. *is_executable* überprüft ob eine Datei ausführbar ist. Genauso gibt es *is_readable* und *is_writable*.

- chdir – In ein bestimmtes Verzeichnis wechseln
- dir – Ein pseudo-objektorientierter Mechanismus zum Lesen von Verzeichnissen.
- opendir – Öffnet ein Verzeichnishandle.
- closedir – Schliesst ein Handle, das von *opendir* geöffnet wurde.
- getcwd – Gibt das aktuelle Arbeitsverzeichnis zurück
- readdir – Liest einen Eintrag von einem Verzeichnis-Handle
- rewinddir – Setzt ein Verzeichnis-Handle zurück.
- scandir – Liest ein Verzeichnis aus
- glob – Findet Pfadnamen die dem angegebenen Ausdruck entsprechen

2.1.3 Arrays

Für die Verwendung von Arrays gibt es in PHP einige nützliche Funktionen, dazu gehören unter anderem folgende:

- array_keys – Liefert alle Schlüssel eines Arrays
- array_values – Liefert alle Werte eines Arrays
- array_flip – Vertauscht Werte und Schlüssel in einem Array
- array_reverse – Liefert ein Array mit umgekehrter Reihenfolge der Elemente
- array_pop – Liefert das letzte Element eines Arrays
- array_shift – Liefert ein Element vom Beginn eines Arrays
- array_map – Wendet eine Callback-Funktion auf die Elemente von Arrays an
- array_walk – Wendet eine Benutzerfunktion an jedem Element eines Arrays an

- array_search – Durchsucht ein Array nach einem Wert liefert bei Erfolg den Schlüssel
- array_merge – Führt ein oder mehrere Arrays zusammen
- asort – Sortiert ein Array und behält die Indexverbindungen
- ksort – Sortiert ein Array nach Schlüsseln
- natsort – Sortiert ein Array in "natürlicher Reihenfolge"
- rsort – Sortiert ein Array in umgekehrter Reihenfolge
- array_multisort – Sortiert mehrere oder multidimensionale Arrays
- ...siehe `http://www.php.net/manual/en/ref.array.php`

Vor allem array_map und array_walk können bei der Verarbeitung eines Arrays nützliche Dienste erweisen, so muss man sich nicht selbst Fuktionen schreiben die ein Array durchlaufen, sondern lediglich eine Funktion die ein Element eines Arrays verarbeitet.

2.1.4 Mail

Mit PHP ist es auch möglich Mails zu versenden, insofern der Server auf dem PHP ausgeführt wird entsprechend konfiguriert ist. Sollte das nicht der Fall sein, gibt es PHP-Klassen die Mail-Funktionen bereitstellen um einen externen Mailserver zu verwenden. Selbstverständlich kann man diese auch selbst schreiben ;-). Die Funktion mail *bool mail (string to, string subject, string message* [*, string additional_headers* [*, string additional_parameters*]] *)* dient dazu Mails zu versenden. Hierbei ist darauf zu achten, dass falls zusätzliche Header angegeben werden diese Zeilen mittels „\r\n“ zu trennen sind. Der Rückgabewert „true“ gibt lediglich an, dass eine Mail an den MTA zum versenden übergeben wurde. Es bedeutet nicht, dass die Mail auch den Empfänger erreicht hat.

Algorithm 22 Mail versenden

```php
<?php
$empfaenger = 'empfaenger@example.com';
$betreff = 'Betreffzeile';
$nachricht = 'Hallo Text foo bar';
$header = 'From: absender@example.com' . "\r\n" .
  'Reply-To: replyto@example.com' . "\r\n" .
  'X-Mailer: PHP/' . phpversion();

mail($empfaenger, $betreff, $nachricht, $header);

//Weitere Parameter an sendmail übergeben (Das ist u.U.
    notwendig damit die Mails vom MTA akzeptiert werden.)
mail('empfaenger@example.com', 'Betreff', 'Nachricht',
    null, '-fabsender@example.com');
?>
```

2.1.5 Sessions

Sessiondaten sind Daten, die z.B. von einem PHP-Script auf dem Server gespeichert werden, und die meistens für die Dauer des Besuchs (Session/Sitzung) auf mehreren Pages einer Website ihre Gültigkeit behalten. Sessiondaten können z.B. dazu benutzt werden, um dem Besucher restriktiven Zugriff (z.B. nach einem Login) auf Teile einer Website zu erlauben - üblicherweise zu einem Administrationsbereich. Bei Verwendung dieser Funktionen ist zu beachten, dass eine neue Session nur dann angelegt werden kann, wenn noch keine Ausgabe stattgefunden hat. Sessions werden mit dem Befehl *session_start()* gestartet. Der Zugriff auf Sessionvariablen erfolgt über das assoziative Array $_SESSION. Durch das erste Ansprechen eines Arrayelements wird die Sessionvariable angelegt, durch ein *unset()* kann eine Variable gelöscht werden.

Algorithm 23 Sessions

```php
<?php
session_start();
$_SESSION['loggedin'] = true;
?>
```

2.2 Objektorientierung

2.2.1 Grundlegendes

Für die Objektorientierte Programmierung bietet PHP5 ein klassenbasiertes Objektmodell an, das stark an das von C# oder Java angelehnt ist. Ein objektorientiertes PHP-Programm besteht aus einer Menge von PHP-Klassen. Zuallererst stellt sich aber die Frage warum man in PHP objektorientiert programmieren sollte. Dazu schauen wir uns folgendes Beispiel in Algorithmus 24 an.

Algorithm 24 Daten aus einer Datenbank verarbeiten ohne OOP

```
1  <?php
2  function handle_error() {
3    // Fehler verarbeiten
4  }
5
6  $connection = mysql_connect($host,$user,$pass);
7
8  if ($connection === false) handle_error();
9
10 if (mysql_select_db('thedb',$connection) === false)
       handle_error();
11
12 $result = mysql_query("SELECT FROM foo WHERE bar = 'baz'")
       ;
13
14 if ($result === false) handle_error();
15
16 while(($row = mysql_fetch_assoc($result)) !== false) {
17   // Daten verarbeiten
18 }
19
20 mysql_free_result($result);
21 ?>
```

Selbst wenn man diesen Code in eine Include-Datei auslagert, stellt sich das Problem der wiederverwendbarkeit. Es müssen globale Variablen verwendet werden und diese werden überschrieben, wenn man eine zweite Datenbankverbindung aufbauen will. Besser ist hier also die Verwendung von Objektorientierung zur Kapselung der Variablen und Funktionen. Algorithmus 25 veranschaulicht dies.

Algorithm 25 Daten aus einer Datenbank verarbeiten ohne OOP

```php
<?php

class DB {
  var $connection = null;
  var $result = null;

  public function connect($user='',$pass='',$host='',
      $database) {
    // connect to DB
    $this->connection = mysql_connect($host,$user,$pass);
    if ($this->connection === false) $this->handle_error();
    if (mysql_select_db($database,$this->connection) ===
        false) handle_error();
  }

  public function disconnect() {
    if (is_resource($this->connection)) {
      mysql_close($this->connection);
    }
  }

  public function query($sqlstr) {
    if (!is_resource($this->connection)) return false;
    if (is_resource($this->result)) mysql_free_result($this
        ->result);
    $this->result = mysql_query($sqlstr,$this->connection);
  }

  //MYSQL_ASSOC ist eine vordefinierte Konstante
  public function getRow($fetchmode = MYSQL_ASSOC) {
    if (!is_resource($this->result)) return false;
    $row = mysql_fetch_array($this->result,$fetchmode);
    if (!is_array($row)) return false;
    return $row;
  }

  //Verwenden der Klasse:
  require_once('DB.class.php');
  $db = new DB();
  $db->connect('localhost','User','Pass','mydb');
  $db->query("SELECT foo FROM bar");
  while ($row = $db->fetchRow(MYSQL_NUM)) {
    // Daten verarbeiten
  }
  $db->disconnect();
}
?>
```

Wie man sieht ist das Handling der SQL-Verbindung erheblich vereinfacht und der Code ist viel besser wartbar geworden.

2.2.2 Objekte in PHP5: Definition, Instanzierung, Erweiterung

Klassen werden in PHP mit dem Keyword *class* eingeleitet. Mit *var $variable;* können Klassenvariablen definiert werden. Eine Instanz einer Klasse erzeugt man zur Laufzeit mit dem *new*-Keyword. Mittels -> kann auf Funktionen oder Variablen der Klasse zugegriffen werden. Das keyword *static* ermöglicht es Klassenmethoden statisch aufzurufen mittels *Klassenname::methode();*. Statische Variablen werden auch mit diesem keyword versehen und die Änderung des Inhalts einer solchen Variable (*Klassenname::$variable*) eines Objekts ändert den Inhalt dieser Variablen in allen Instanzen dieser Klasse. Mit dem keyword *const* können Klassenkonstanten deklariert werden, diese sind dann via *Klassenname::konstante* zugreifbar. Erfolgt der Zugriff auf eine statische Variable oder Methode aus der Klasse selbst heraus, so kann das keyword *self* verwendet werden: *self::$variable*.

Es gibt seit PHP5 auch Konstruktoren und Destruktoren, diese Methoden haben den speziellen Namen *__construct()* und *__destruct()*;

Klassen können auch erweitert werden indem man Klassen vererbt.

Algorithm 26 Vererbung

```
1 <?php
2 class Class2 extends Class1 {
3   // Class2 erbt die Methoden und Variablen von Class1
4 }
5 ?>
```

Mit dem Schlüsselwort *final* deklarierte Methoden und Variablen können nicht weitervererbt werden. PHP unterstützt keine Mehrfachvererbung. Wird in der abgeleiteten Klassen keine Konstruktor- bzw. Destruktor-Methode deklariert, so wird implizit der Vaterklassen-Konstuktor bzw. -Destruktor aufgerufen. Explizt kann man diese auch mittels *parent::__construct();* bzw. *parent::__destruct();* aufrufen.

2.2.3 Funktionen zur Arbeit mit Objekten

Seit PHP4 existieren Methoden zur Introspektion, d.h. für die Analyse von Klassendefinitionen. Die folgenden Beispiele sind anhand der Methodennamen selbsterklärend, für genauere Beschreibungen sei wieder einmal auf das Handbuch verwiesen `http://www.php.net/manual/en/ref.classobj.php`.

- get_class_methods – Liefert die Namen aller Methoden einer Klasse

- get_class_vars – Liefert die Standard-Elemente einer Klasse
- class_exists – Prüft, ob eine Klasse definiert ist
- get_class – Gibt den Namen der Klasse eines Objektes zurück
- get_parent_class – Gibt den Namen der Elternklasse eines Objektes zurück
- is_a – Gibt true zurück, wenn das Objekt eine Instanz dieser Klasse ist oder diese Klasse als Superklasse hat

2.2.4 Vergleich PHP4 / PHP5

Mit PHP Version 5 wurde das Objekt-Handling zum Zweck einer besseren Performance und erweiterter Funktionalität vollständig überarbeitet. In früheren Versionen von PHP wurden Objekte wie primitive Typen behandelt (z.B. Integers oder Strings). Folge dieser Vorgehensweise war, dass bei Zuweisung des Objekts an eine Variable oder be der Übergabe an eine Methode das komplette Objekt kopiert wurde. Im neuen Modell werden Objekte als Referenzen betrachtet, nicht als Werte.

Zusammengefasst wurden mit PHP 5 folgende Neuerungen eingeführt:

- **Einheitliche Konstruktoren** In PHP4 waren Konstruktoren Methoden mit dem gleichen Namen wie die Klasse selbst. PHP 5 führt mit *__construct()* einen einheitlichen Weg ein, um Konstruktoren zu deklarieren.
- **Desktruktoren** Methoden, die bei der Zerstörung eins Objekts aufgerufen werden, können z.B. zum Schließen von Datenbankverbindung oder für andere Äufräumarbeitenßehr praktisch sein. Seit PHP 5 steht hierfür ein Destruktor-Mechanismus über die Methode *__destruct()* zur Verfügung.
- **Abstrakte Klassen und Methoden** Mittels des Schlüsselwortes *abstract* können Klassen und und Funktionen deklariert werden, die nicht instanziiert werden können.
- **Interfaces** PHP 5 verfügt über Interfaces; eine Klasse darf dabei eine beliebige Anzahl von Interfaces implementieren.
- **instanceof** Mittels dieses Schlüsselwortes kann geprüft werden, ob ein Objekt Instanz einer bestimmten Klasse ist, diese Klasse ableitet oder ein entsprechendes Interface implementiert.
- **Konstanten** Mittels *const* können Klassenkonstanten definiert werden.
- **Statische Mehoden / Eigenschaften** Per Schlüsselwort *static* werden statische Methoden deklariert, die auch ausserhalb des Klassenkontexts aufgerufen werden können. Statische Eigenschaften von Klassen können auch initialisiert werden.

- **final** Das Schlüsselwort *final* deklariert Methoden und Klassen, die in Subklassen nicht überschrieben werden dürfen.
- **Exceptions** Vergleichbar mit anderen Programmiersprachen führt PHP 5 ein Exception Modell (*try... catch...*) ein.
- **neue Konstante *__METHOD__*** Diese zeigt die gegenwärtige Klasse / Methode an.
- ***__autoload()*** Diese Interceptor-Funktion wird automatisch aufgerufen, wenn eine nicht deklarierte Klasse instanziiert werden soll.
- **private / protected für Eigenschaften und Methoden** Mittels dieser beiden Schlüsselwörter kann die Sichtbarkeit von Eigenschaften und Methoden verändert werden.
- **Reflection API** ermöglicht Reverse Engineering
- **Object Cloning** mittels der Methode *__clone()*

Die folgende Tabelle fasst nochmals alle wesentlichen Unterschiede zwischen PHP 4 und PHP 5 zusammen:

	PHP4	PHP5
Klassendefinition	`class <name> {...}`	`([abstract] | [final]) class <name> {...} [implements <interface>]`
Methoden	`function(){...}`	`([abstract] | [final] [static]) [public | private | protected] function(){...}`
Konstruktoren	Name der Klasse	`__construct`
Destruktoren	n/a	`__destruct`
Besondere Methoden	`__sleep`, `__wakeup`	`__call`, `__get`, `__set`, `__isset`, `__unset`, `__sleep`, `__wakeup`, `__toString`, `__clone`, `__autoload`
Variablen	`var <name> [= expr;]`	`(public | private | protected) [static] <name> [=expr;]`
Konstanten	n/a	`const <name_ohne_$> [=expr;]`
Mehrfachvererbung	n/a	n/a
Interfaces	n/a	`interface <name> {...}`
Überladung	n/a	Methoden,Variablen (`__call ..`)
Objektiteration	wie Hash (`foreach`)	wie Hash (`foreach`) oder Implementierung der internen Interfaces `Iterator`, `IteratorAggregate`
Patterns	n/a	`Factory`, `Singleton`
Klonen	`$obj1 = $obj2` (!)	`$obj1 = clone $obj2`
Gleichheit von Objekten	`$obj1 === $obj2`: Attribute,Werte,Klasse gleich	`$obj1 == $obj2`: Attribute,Werte,Klasse; `$obj1 === $obj2`: gleiche Instanz der gleichen Klasse
Reflection	n/a	API für reverse engineering
Type hinting	n/a	möglich
Autoloading	n/a	mittels `__autoload`

Tabelle 2.1: Vergleich der objektorientierten Programmierung zwischen PHP4 und PHP5

2.3 Kontrollfragen

- Wie kann ich das aktuelle Datum in der Form tt.mm.jj ausgeben?
- Wie kann ich einen String an eine Datei anhängen? (z.B. bei Log-Files)
- Kann ich Objekte als Keys für Arrays verwenden? Kann ich Objekte in Arrays ablegen?
- Wozu braucht man Sessions?
- Wie rufe den Konstruktor der Vaterklasse auf?
- Wie erzeuge ich eine identische Kopie eines Objekts?
- Kennst Du ein Beispiel eines objektorientierten Codes, der unter PHP5, aber nicht unter PHP4 funktioniert?

2.4 Übungen

- Schreibe eine Klasse *StopWatch* um die Ausführungszeit eines PHP-Skripts zu messen.
- Erweitere die Klasse um eine Funktion *pause()* um Teile des Skripts von der Zeitmessung auszunehmen.
- Entwirf eine Klasse DateTool um den Zugriff auf häufig verwendete Datumsfunktionen zu kapseln. Benutze die Klasse statisch.
- Entwirf die Klassen Person, Student und Mitarbeiter mit einer Methode *toHTML()*. Student und Mitarbeiter sollen dabei von Person abgeleitet sein und zusätzlich zum Vor- und Nachnamen der Person die Matrikelnummer bzw. den Chef des entsprechenden Angestellten ausgeben (inklusive aller seiner Details).

3. Session: Datenbankanbindung / PEAR

3.1 Grundlegendes zu Datenbanken

3.2 Datenbankzugriff am Beispiel von MySQL

Im Abschnitt 2.2 haben wir bereits gesehen wie man eine Klasse entwickelt die den Zugriff auf eine MySQL-Datenbank kapselt. Hier wollen wir deshalb nur noch auf die verwendeten und weitere Methoden im Zusammenhang mit MySQL eingehen:

- mysql_connect – Öffnet eine Verbindung zu einem MySQL-Server
- mysql_close – Schließt eine Verbindung zu MySQL
- mysql_select_db – Auswahl einer MySQL Datenbank
- mysql_query – Sendet eine Anfrage an MySQL
- mysql_real_escape_string – Maskiert spezielle Zeichen innerhalb eines Strings für die Verwendung in einer SQL-Anweisung
- mysql_fetch_array – Liefert einen Datensatz als assoziatives Array, als numerisches Array oder beides
- mysql_free_result – Gibt belegten Speicher wieder frei
- mysql_error – Liefert den Fehlertext der zuvor ausgeführten MySQL Operation
- mysql_errno – Liefert die Nummer einer Fehlermeldung einer zuvor ausgeführten MySQL Operation
- mysql_create_db – Anlegen einer MySQL-Datenbank

Wichtig ist vor allem die Verwendung von *mysql_real_escape_string()* sobald man Variablen in die SQL-Query einbaut. Damit kann man sogenannte SQL-Injections verhindern die von Angreifern verwendet werden um Schwachstellen im Code auszunutzen.

3.2.1 Einfügen / Update / Löschen

Das Einfügen, updaten und löschen von Daten in einer MySQL-Datenbank geschieht über SQL, die Structured Query Language. SQL ist eine Abfragesprache für Relationale Datenbanken (wie z.B. MySQL). Da eine komplette Einführung in SQL den Rahmen dieses Kurses bei weitem sprengen würde, werden hier nur die vier grundlegendsten Befehle erwähnt: *SELECT*, *UPDATE*, *INSERT* und *DELETE*.

Mittels *SELECT feld1, feld2 FROM table* kann man Daten aus einer Tabelle der Datenbank abfragen.

Mittels *UPDATE table1 SET feld1='neuerwert' WHERE feld1='alterwert'* kann man einen speziellen Wert mit einem anderen überschreiben.

Mittels *INSERT INTO table1 (feld1,feld2) VALUES('wert1',123)* kann man in der Tabelle table1 eine neue Zeile erzeugen die die Einträge feld1='wert1' und feld2=123 enthält.

Mittels *DELETE FROM table1 WHERE feld1=123* werden aus der Tabelle table1 alle Zeilen gelöscht, die in feld1 den Wert 123 stehen haben.

3.3 PEAR

PEAR bedeutet PHP Extension and Application Repository und ist eine Sammlung von PHP Klassen die die Arbeit mit PHP erheblich vereinfachen.

3.3.1 Überblick

Es gibt viele verschiedene Klassen in Pear unter anderem zu den Themen Datebankzugriff (Pear::DB), Formularen (PEAR::HTML_Quickform), Authentifizierung (PEAR::Auth) oder automatisierte Tests (PHPUnit). Auf ein paar Pear-Klassen wollen wir nun im Detail eingehen.

3.3.2 Datenbankabstraktion: DB

Für den Zugriff auf relationale Datenbanken bietet es sich an einen Datenbankabstraktionslayer zu verwenden um die Möglichkeit zu haben zu einem späteren Zeitpunkt auf einfache Art und Weise das Datenbankmanagementsystem zu wechseln. Pear bietet hierzu verschiedene Klassen an, dazu gehören Pear::DB und Pear::MDB(2). Da Pear::MDB2 Pear::DB nahezu abgelöst hat, empfiehlt es sich Pear::MDB2 einzusetzen.

Algorithm 27 Pear::MDB

```php
<?php
require_once 'MDB2.php';
$dsn = 'pgsql://user:pass@hostname/mydb';
$options = array(
   'debug' => 2,
   'result_buffering' => false,
);

$mdb2 =& MDB2::factory($dsn, $options);
if (PEAR::isError($mdb2)) {
   die($mdb2->getMessage());
}

// ... Die mdb-klasse verwenden

$mdb2->disconnect();
?>
```

Algorithmus 27 zeigt wie man eine Verbidnung zur Datenbank herstellt, jedoch sollen auch Daten abgefragt werden.

Algorithm 28 Pear::MDB Query

```php
// Select ausführen
$res =& $mdb2->query('SELECT * FROM table1');

// Zeilenweise Daten holen bis keine Daten mehr verfügbar
    sind
while (($row = $res->fetchRow())) {
   echo $row[0] . "\n";
}
```

Wie man sieht ähnelt das Prinzip von MDB dem der in Algorithmus 25 gezeigten Vorgehensweise.

3.3.3 Internationalisierung: i18n

Internationalisierung hat in grösseren Projekten mehr und mehr Bedeutung. Für I18N - so das Kürzel für Internationalisierung - kann entweder die eingebaute Funktionalität von PHP verwendet werden oder auf eine der zahlreichen I18N-Klassen zurückgegriffen werden. Es ist auf jeden Fall sinnvoll die ausgegebenen Strings mit einer Funktion zu kapseln, die i18n-Funktionalität kann dann ja im Nachhinein auch entsprechend ausgetauscht werden.

Algorithm 29 Simples I18N Beispiel

```
<?php
define('DEFAULT_LANG','en');
$GLOBALS['i18ntexts'] = array(
  'de' => array(
    'HELLO_WORLD' => 'Hallo Welt',
    'BYE_WORLD' => 'Tschüss Welt'
  ),
  'en' => array(
    'HELLO_WORLD' => 'Hello World',
    'BYE_WORLD' => 'Bye World'
  )
);

function getI18NStr($key='',$lang=DEFAULT_LANG) {
  if (isset($GLOBALS[$lang][$key])) return $GLOBALS[$lang][
      $key];
  else return @$GLOBALS[DEFAULT_LANG][$key];
}

// Einen internationalisierten String ausgeben
echo getI18NStr('HELLO_WORLD');
?>
```

Algorithmus 29 zeigt ein einfaches Beispiel wie man Internationalisierung realisieren kann. Praktischerweise kann man nun das Tool xgettext[1] verwenden um sich alle I18n-Strings aus den PHP-Dokumenten herauszuziehen, vorausgesetzt man hat immer die Funktion *getI18NStr* verwendet.

3.3.4 Authentifizierung: Auth

Mittels Pear::Auth kann eine sehr simple Benutzerauthentifizierung implementiert werden. Pear::Auth kann dabei verschiedene Authentifizierungsmechanismen verwenden, u.A. mittels Pear::DB gegen eine Datenbank authentifizieren oder einen LDAP-Server abfragen.

[1] Siehe GNU gettext: `http://www.gnu.org/software/gettext/manual/html_chapter/gettext.html`

Algorithm 30 Pear::Auth

```php
<?php
require_once "Auth.php";

// Diese Funktion zeigt das Login-Formular an
function loginFunction() {
   echo '<form method="post" action="test.php">';
   echo '<input type="text" name="username">';
   echo '<input type="password" name="password">';
   echo '<input type="submit">';
   echo '</form>';
}

// Authentifizierungsdetails festlegen
$dsn = "mysql://user:password@localhost/database";
$auth = new Auth("DB", $dsn, "loginFunction");

$auth->start();
if ($auth->checkAuth()) {
  // An dieser Stelle ist der Benutzer authentifiziert
  echo 'Willkommen ',$auth->getUsername();
}
?>
```

Algorithmus 30 zeigt wie einfach die Verwendung der Auth-Klasse ist. Es kann über verschiedene Optionen konfiguriert werden wo die Authentifizierungsinformationen zu finden sind oder welche Datenbankfelder man verwenden will. Genauere Details dazu finden sich in der Dokumentation des Pakets: `http://pear.php.net/manual/en/package.authentication.auth.php`. Für fortgeschrittene Nutzer die ausser einer Authentifizierung auch ein Rechteverwaltungssystem benötigen, sei an dieser Stelle auf Pear::LiveUser[2] verwiesen.

3.4 Kontrollfragen

- Was heisst SQL?
- Wie heisst das SQL-Sprachkonstrukt um Daten aus einer relationalen Datenbank abzufragen?
- Welche Datenbanken unterstützt PHP?
- Welche Möglichkeiten gibt es die eigenen PHP-Skripte unabhängig von einer bestimmten Datenbank zu machen?
- Was heisst I18N? Was heisst L10N?
- Welche Funktionen bietet Pear? Kann man das essen?

3.5 Übungen

- Benutze die Klasse Pear::Auth mit Datenbankauthentifizierung um eine passwortgeschützte Seite zu realisieren.