
# Seminararbeit

Programmierung verteilter Systeme

*PHP: Eine Skriptsprache zur Programmierung von Web-Anwendungen*

Christian-Albrechts-Universität zu Kiel
Sommersemester 2003

Mirko Sarach

**Betreuer**
Michael Hanus

**Literatur**
http://www.php.net/

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Was ist PHP?

PHP[1] („PHP: Hypertext Preprocessor“) ist eine serverseitig interpretierte, in HTML eingebettete Sprache. Die Syntax ist der von C, JAVA oder Perl ähnlich. Der Hauptfokus von PHP liegt auf der Entwicklung von serverseitigen Skripten, somit lässt sich mit PHP alles verwirklichen, was auch mit anderen CGI-Programmen möglich ist, wie z. B. das Generieren dynamischer Webseiten oder das sammeln von Formulardaten. Man hat aber auch die Möglichkeit Kommandozeilenskripte, oder GUI-Applikation zu erstellen. Die grösste Stärke liegt in der Unterstützung einer grossen Anzahl an Datenbankmanagementsystemen.
So lassen sich mit PHP Web-Anwendungen entwickeln die, da die Darstellung auf dem Client in einem Webbrowser erfolgt, unabhängig vom Standort oder vom Betriebssystem des Zielrechners ist.

## 1.2 Die Geschichte

1995 wurde PHP von Rasmus Lerdorf entwickelt. Zu dieser Zeit hieß PHP noch PHP / FI was für „Personal Home Page / Forms Interpreter“ stand. PHP war noch komplett in Perl geschrieben und regelte den Zugriff auf die Webseite von Rasmus Lerdorf.
Die dritte Version erschien 1997 und wurde mit der Hilfe von Zeev Suraski und Andi Gutmans entwickelt. PHP, nun komplett in C geschrieben, war Ende 1998 auf schätzungsweise 10% der Webserver im Internet installiert. Schon kurz nach Erscheinen von Version 3 wurde an der 4. Version gearbeitet. 2000 fertiggestellt gewann PHP 4 durch eine neue Engine nochmals an Effizienz.
Die aktuelle Version ist PHP 4.3.2RC4.

# Kapitel 2

# Grundlagen

## 2.1 Die Funktionsweise von PHP

### 2.1.1 Ablauf einer Clientanfrage

PHP ist eine sereverseitige Sprache, d.h. stellt ein Client eine Anfrage nach einer Datei und diese enthält PHP-Code, so wird dieser Code zunächst auf dem Server ausgewertet und dann das Ergebnis als HTML-Code an den Client gesendet. Es gibt zwei Möglichkeiten PHP zu installieren, einmal als CGI ( “Common Gateway Interface “) oder als Modul für den Webserver. Dementsprechend unterscheidet sich auch die Abarbeitung einer Anfrage.

#### PHP als CGI

CGI ist ein Konzept nach dem Webserver und Programme, zur Erzeugung von dynamischen Webseiten, miteinander kommunizieren können. CGI ist eine weitverbreitete standardisierte Schnittstelle. Ein Browser schickt eine für ein CGI-Skript bestimmte Anfrage an den Server. Dieser entnimmt dem HTTP-Request den Namen der CGI-Datei und startet diese dann in einem eigenen Systemprozess. Das Skript erzeugt dann eine HTML-Ausgabe, die an den Browser zurückgesendet wird. Mit Beendigung des Skripts wird die Verbindung Browser Server geschlossen und die Resourcen des Skriptes werden wieder freigegeben.
Das starten eines eigenen Prozesses erfordert natürlich einige System-Calls, was sich unterumständen in der Laufzeit des Skriptes widerspiegeln kann.

#### PHP als Server-Modul

Schickt ein Browser eine Anfrage an den Webserver, so wird zunächst wiederum die Datei in den Speicher des Webservers geladen. Der Sever erkennt, je nach Konfiguration, an der Dateiendung dass es sich um eine Datei handelt, die PHP-Code enthalten kann. Ist dies der Fall so wird der PHP-Parser gestartet. Dieser filtert den PHP-Code heraus, interpretiert diesen und fügt das Ergebnis in die Datei ein. Ist der Parser am Ende der Datei angekommen, so gibt er die Datei an den Server zurück der diese dann an den Browser schickt.

Hier ein Beispiel wie PHP in HTML eingebunden wird:

```
<html>
    <?
      // eine Moeglichkeit einen PHP-Block zu beginnen
        $date=date("d.m.y");
            // date gibt das heutige Datum formatiert zurueck

      // den PHP-Block beenden
   ?>
<body>
    <h1> Heute ist der <?=$date ?>  <- die kuerzeste Weise
                                    PHP in HTML auszugeben

    <?PHP
        // Ein neuer PHP-Block
        echo"<p> Hier wird HTML-Code ausgegeben </p>
    ?>
</body>
</html>
```

### 2.1.2 Der Aufbau von PHP

Seit Version 4 besteht das PHP-System aus drei voneinander getrennten Teilen.

#### Zend-Engine

Die Zend-Engine ist Parser und Interpreter. Sie analysiert den Quellcode, interpretiert diesen und führt ihn aus.

#### API-Module

Dies sind die einzelnen Bibliotheken. Hier ist ein Grossteil der Funktionalität von PHP implementiert. Dazu zählen unter anderem die zahlreichen Schnittstellen zu Datenbanken, die Grafikbibliotheken, IMAP-Funktionen oder Funktionen zur Erstellung von PDF-Dateien.

#### SAPI-Schnittstelle

Mit Hilfe der SAPI-Schnittstelle ( “Server Application Programm Interface “) werden die Systemunabhängigen PHP-Befehle in serverspezifischen Befehle übersetzt. Dies hat den Vorteil, dass nur diese Schnittstelle neu programmiert werden muss, falls man PHP auf einem noch nicht unterstützten System einsetzten möchte. So ist zu erklären, dass PHP für jeden gängigen Webserver zu haben ist.

## 2.2 Syntax

### 2.2.1 Abgrenzung von Anweisungen und Kommentare

Wie in C oder Perl wird jede Anweisung durch ein Semikolon beendet.
Einzeilige Kommentare sind bis zum Zeilenende gültig und werden mit “ // “ oder “ # “ eingeleitet. Mehrzeilige Kommentare stehen zwischen “ /* “und “ */ “.

### 2.2.2 Variablentypen und Variablen

In PHP wird dem Variablennamen ein $ vorangestellt. Bei Variablennamen wird zwischen Gross- und Kleinschreibung unterschieden. Ein gültiger Variablenname beginnt mit einem Buchstaben oder einem Unterstrich gefolgt von einer beliebigen Anzahl an Buchstaben, Zahlen oder Unterstrichen. Den Typ der Variable bestimmt PHP zur Laufzeit selbst. Der Variablentyp kann aber auch bei der ersten Benutzung festgelegt werden, indem er davor in Klammern angegeben wird. PHP unterscheidet folgende Variablentypen:

| Typ | Beispiel |
|---|---|
| Boolean | $a = true |
| Integer | $a = 5 |
| Float | $a = 5.0 |
| String | $a = “String “ |
| | oder $a=‘String ‘ |
| | oder heredoc-Syntax |
| Array | $a[0]=2; |
| | oder $a[zahl]=5 |

Mit Resource stellt PHP einen weiteren Typ zur Verfügung, der von einigen Funktionen als Rückgabewert für externe Verbindungen genutzt wird. Resourcen können aber nicht vom Benutzer erzeugt werden. Wird auf eine Resource nicht mehr zugegriffen wird sie von einem garbage collector wieder frei gegeben.
Wird ein String in doppelten Anführungszeichen oder mit heredoc-Syntax angegeben so werden Vorkommen von Variablen ausgewertet. Die heredoc-Syntax wird durch <<< eingeleitet. Danach folgt ein Bezeichner mit dem die Zeichenkette auch wieder schliesst.

```
$str=<<<EOD
Dies ist ein String
in heredoc-Syntax.
Alles was zwischen EOD und EOD; steht
gehoert zum String.
EOD;
```

Mit PHP4 ist auch der Datentyp Boolean eingeführt worden. Bei der Umwandlung einer Variablen in den Typ boolean werden folgende Werte als false interpretiert.

- integer: der Wert 0
- Float: der Wert 0.

- String: ein leerer String oder der String “0“
- Array: ein Array ohne Elemente
- Objekte: ein Objekt ohne Elemente
- Sowie ein spezieller Typ NULL für Variablen, die keinen Wert haben.

Sonstige Werte ergeben true.
Der Gültigkeitsbereich einer Variablen ergibt sich aus dem Zusammenhang. Das folgende Beispiel zeigt zwei unterschiedliche Vorkommen von $a:

```
<?
    $a="Hallo Welt";
    function func(){
        $a="Auf Wiedersehen";
    }
    func();
    echo"$a";  // Ausgabe: "Hallo Welt";
?>
```

Variablen sind global falls sie nicht innerhalb von Funktionen vorkommen. Um innerhalb einer Funktion auf eine globale Variable zuzugreifen muss sie innerhalb der Funktion als global definiert werden.

```
<?
    $a="Hallo Welt";
    function func(){
        global $a;  // $a ist global
        $a="Auf Wiedersehen";
    }
    func();
    echo"$a";  // Ausgabe: "Auf Wiedersehen";
?>
```

Statt global kann man auch das spezielle assoziative PHP-Array $GLOBALS benutzten. Hier sind die Schlüssel die Variablennamen der globalen Variable.

```
<?
    $a="Hallo Welt";
    function func(){
        $GLOBALS["a"]="Auf Wiedersehen";
    }
    func();
    echo"$a";  // Ausgabe: "Auf Wiedersehen";
?>
```

Über den Umgang mit Arrays folgt mehr in Abschnitt 2.2.4.
Ab PHP 4 ist es auch möglich Referenzen auf Variablen zu nutzen. Hierzu wird der Quellvariable einfach ein & vorangestellt.

```
$foo = 'Bob';                    // 'Bob' der Variablen $foo zuweisen.
$bar = &$foo;                    // Zeiger auf $foo in $bar erzeugen.
$bar = " Mein Name ist $bar ";   // $bar ver{\"a}ndern...
echo $foo;
echo $bar;
// Gibt zweimal "Mein Name ist Bob " aus.
```

### 2.2.3 Konstanten

Konstanten werden in PHP über die Funktion define() definiert. Einmal definiert kann eine Konstante weder geändert noch gelöscht werden. Konstanten können nur Werte vom Typ boolean, integer, float oder string enthalten. Weiter sollten Konstanten nicht mit einem $ Zeichen beginnen.

```
define("CONSTANT", "Hallo Welt.");  // Definition von CONSTANT
echo CONSTANT;                      // Ausgabe: "Hallo Welt."
```

### 2.2.4 Arrays

In PHP werden mehrdimensionale Arrays mit mehreren einzelnen eckigen Klammern angegeben. Eine Besonderheit stellen die assoziativen Arrays dar. Hier kann auch ein String als Schlüssel benutzt werden.

```
$person=array();  // Array person erzeugen.
$person[1]["nachname"]="Lustig";     $person[1]["vorname"]="Peter";
$person[2]["nachname"]="Meiser";     $person[2]["vorname"]="Hans";
$person[3]["nachname"]="Mustermann"; $person[3]["vorname"]="Max";
// Ein zweidimensionales Array mit Werten f{\"u}llen

echo $person[2]["nachname"]." ".$person[2]["vorname"];
// Ausgabe: "Hans Meiser"
echo $person[2][0]." ".$person[2][1];
// Ausgabe: "Hans Meiser"
```

### 2.2.5 Operatoren

Operatoren sind analog zu C definiert.

- Arithmetische Operatoren:

| Beispiel | Name |
|---|---|
| $a + $b | Addition |
| $a − $b | Subtraktion |
| $a ∗ $b | Multiplikation |
| $a / $b | Division |
| $a % $b | Modulus |

- Bit Operatoren:

| Beispiel | Name | Ergebnis |
|---|---|---|
| $a & $b | Und | Bits, die in a und b gesetzt sind, werden gesetzt. |
| $a \|\| $b | Oder | Bits, die in a oder b gesetzt sind, werden gesetzt |
| $a ^$b | Entweder oder | Bits, die in a oder b aber nicht in beiden gesetzt sind, werden gesetzt |
| ~$a | Nicht | Bits, die in a nicht gesetzt sind, werden gesetzt und umgekehrt |
| $a << $b | Linksshift | schiebe bitweise in a um b Stellen nach links |
| $a >> $b | Rechtsshift | schiebe bitweise in a um b Stellen nach rechts |

- Vergleichs Operatoren:

| Beispiel | Name | Ergebnis |
|---|---|---|
| $a == $b | Gleich | true, falls a und b gleich |
| $a === $b | Identisch | true, falls a und b gleich und vom gleichen Typ ( nur PHP4 ) |
| $a != $b | Ungleich | true, falls a und b ungleich |
| $a <> $b | Ungleich | true, falls a und b ungleich |
| $a !== $b | nicht Identisch | true, falls a und b ungleich oder nicht vom gleichen Typ (PHP 4) |
| $a < $b | Kleiner | true, falls a kleiner b |
| $a > $b | Grösser | true, falls a grösser b |
| $a <= $b | Kleiner gleich | true, falls a kleiner gleich b |
| $a >= $b | Grösser gleich | true, falls a grösser gleich b |

Hier ist zu beachten, dass Strings mit Strings in lexikographischer Reihenfolge verglichen werden. Weiter ist ein Integer immer grösser als ein String. Boolean Werte werden wie in C mit 0,1 als Integer interpretiert.

- Logische Operatoren:

| Beispiel | Name |
|---|---|
| $a and $b | Und |
| $a & & $b | Und |
| $a or $b | Oder |
| $a \|\| $b | Oder |
| $a xor $b | Entweder oder |
| $a ! $b | Nicht |

- Zuweisungsoperator:
Der Zuweisungsoperator ist ein " = ". Er kann aber, durch Vorstellen, mit allen arithmetischen, binären oder mit dem dot-Operator verbunden werden.

```
$a=3;
$b=4;
$a+=$b;
echo $a; // Ausgabe von " 7 "
```

- dot-Operator: Zwei Strings können mit dem dot-Operator " . " verbunden werden.

```
$foo="Hallo";
$bar="Welt";
$foo=$foo." ".$bar;
```

```
foo hat nun den Wert "Hallo Welt"
```

- Fehlerkontrolloperator: Stellt man ein“ @ “ vor einen Ausdruck, so wird die Ausgabe von Fehlermeldungen, die der Ausdruck erzeugen könnte, unterdrückt.

### 2.2.6 Kontroll-Strukturen

In PHP gibt es die Kontroll-Strukturen if, else, elseif, for, foreach, while, do... while, switch, continue und break, wobei die Syntax weitgehend der von C entspricht.

1. **if:** Wie auch in anderen Programmiersprachen erlaubt die if-Struktur die bedingte Ausführung von Programmteilen.

   ```
   if(expr)
       statement;
   ```

   Expr wird auf seinen boolschen Wert hin ausgewertet. Im Falle von true wird statement ausgeführt.
   Will man mehrere Anweisungen ausführen, so werden diese in geschweifte Klammern geschrieben, dies gilt auch für die anderen Kontroll-Strukturen.

2. **else:** Wenn einer if-Struktur eine else-Struktur folgt, so wird diese ausgeführt wenn expr sich zu false auswerten lässt.

   ```
   if($bool)
       statement1;
   else
       statement2;
   ```

   Statement2 wird ausgeführt, falls die Auswertung von $bool false ergibt.

3. **elseif:** Elseif ist eine Kombination von else und if.

   ```
   if($bool)
       statement1;
   elseif($bool_2)
       statement2;
   ```

   Hier wird statement2 nur ausgeführt, falls die Auswertung von $bool false und die von $bool_2 true ergibt.

4. **while:** While-Schleifen sind die einfachste Form von Schleifen in PHP.

   ```
   while($bool)
       statement;
   ```

Statement wird solange ausgeführt bis sich $bool zu false auswertet.

5. **do...while:** Do...while ist dem while sehr ähnlich, jedoch wird die Bedingung erst am Ende überprüft, d. h. der Programmblock wird auf jedenfall einmal ausgeführt.

```
do{
    statement;
}while($bool)
```

6. **for:** For-Schleifen sind die komplexesten Schleifen in PHP.

```
for(expr1; expr2; expr3)
    statement
```

   Zunächst wird in jedemfall expr1 ausgeführt. Danach wird expr2 ausgeführt und dann, falls expr2 sich zu true auswerten lässt, statement. Erst jetzt am Ende des Durchlaufs wird expr3 ausgeführt. Ergibt die Auswertung von expr2 false, so erfolgt der Abbruch und die Schleife wird beendet.

7. **foreach:** Seit PHP4 gibt es foreach um auf eine einfache Art und Weise einen Array zu durchlaufen. Foreach funktioniert nur in Verbindung mit Arrays. Für foreach gibt es zwei Syntaxformen:

   - foreach(array_expr as $value) statement
   - foreach(array_expr as $key => $value) statement

   Die erste Form durchläuft das Array $array_expr. Bei jedem Durchlauf wird der Wert des aktuellen Elements $value zugewiesen und dann der interne Array-Zeiger um eins erhöht. Somit wird beim nächsten Durchgang das nächste Element bearbeitet.
   Die zweite Form arbeitet genauso, außer dass zusätzlich der Variablen $key der aktuelle Schlüssel zugewiesen wird.
   Zu beachten ist auch, dass foreach mit einer Kopie des angegebenen Arrays arbeitet. Somit wird garantiert mit dem ersten Element begonnen. Veränderungen an den ausgegebenen Elementen haben aber keine Auswirkungen auf das originale Array. Trotzdem wird der Array-Zeiger des originalen Arrays bewegt und steht dort, wo der Zeiger der Kopie bei Verlassen von foreach stand.

8. **switch:** Die switch-Anweisung ist gleichbedeutend mit einer Reihe von if-Anweisungen mit gleichem Argument.

```
if($i==1)
    echo" i ist 1";
if($i==2)
    echo" i ist 2";
if($i==3)
    echo" i ist 3";
// Das gleiche nochmal aber anders.

switch($i){
```

```
    case 1:
        echo" i ist 1";
        break;
    case 2:
        echo" i ist 2";
        break;
    case 3:
        echo" i ist 3";
        break;
}
```

9. **break:** Break bricht die Ausführung der aktuellen for, foreach, while, do...while oder switch Anweisungs-Sequenz ab.
   Es besteht die Möglichkeit break ein numerisches Argument anzuhängen und somit die Anzahl der abzubrechenden Strukturen festzulegen.

```
$i=0;
while(++$i){
    switch($i){
        case 5:
            echo" i ist 5";
            break; // Beendet nur switch
        case 10:
            echo" Ende ";
            break 2; // Beendet switch und while
    }
}
```

10. **continue:** Continue bricht die Ausführung der aktuellen Schleife ab und startet einen neuen Durchlauf.
    Mit einem Argument ist es wie bei break möglich, die Anzahl der Schleifen zu bestimmen, die übersprungen werden sollen.

```
$i=1;
while($i > 0){
    while(1){
        while(1){
            echo"Dies wird ausgegeben";
            continue 3;
            echo" Dies wird nie ausgegeben";
        }
        echo" Dies wird nie ausausgegeben";
    }
    echo" Dies wird nie ausgegeben";
}
```

Für if, while, for, foreach und switch gibt es noch eine alternative Syntax, die den Umgang mit HTML-Blöcken erleichtert. So wird die öffnende geschweifte Klammer durch einen “ : “ und die schliessende Klammer durch ein endif;, endwhile;, endfor;, endforeach; bzw. durch endswitch; ersetzt. Dazwischen kann beliebiger HTML-Code stehen.

```
<? if($bool): ?>
    <p> Ist bool true, so wird dieser HTML-Block
    interpretiert </p>
<? endif; ?>
```

### 2.2.7 include und require

Der Befehl include(“dateiname.inc “) fügt an dieser Stelle den Inhalt der Datei dateiname.inc ein. Er ermöglicht es Quellcode, der in mehreren Dateien gebraucht wird, zentral zu halten, so dass Änderungen vereinfacht werden, da sie nur einmal durchgeführt werden müssen.
Die Datei, die eingefügt wird, wird als HTML-Code interpretiert. Somit muss, falls die Datei mit PHP-Code anfängt, diese auch mit <?PHP anfangen und, falls sie mit PHP-Code endet, auch mit ?> aufhören. Soll der include-Befehl in Verbindung mit Bedingungen oder Schleifen eingesetzt werden, so muss er mit geschweiften Klammern umschlossen werden.

```
if($bool){
    include("datei1.inc");
}else{
    include("datei2.inc");
}
```

Require verhält sich analog zu include. Einzige Ausnahme ist der Umgang mit Fehlern. Include erzeugt lediglich ein Warning, d.h. dass die Ausführung des Skriptes fortgesetzt wird. Require dagegen gibt bei einem Fehler einen Fatal Error aus, was die Programmausführung beendet.

Ab PHP4 gibt es eine Variante include_once() bzw. require_once() welche den Inhalt der Datei nur einmal einfügt.

### 2.2.8 Funktionen

Eine Funktion wird wie folgt definiert:

```
function fun($arg_1,..., $arg_n){
    statement
}
```

Es kann jeder beliebige PHP-Code als Funktion definiert werden, sogar andere Funktionen- und Klassendefinitionen.
Es ist nicht möglich, Funktionen zu überladen, da dies als eine neue Definition der Funktion interpretiert wird und so zu einer Fehlermeldung führt. Ab PHP4 müssen Funktionen auch nicht definiert sein bevor sie benutzt werden. Einzige Ausnahme stellen die bedingten Funktionen dar.

```
$bool=true;
// noch kann fun_1() nicht aufgerufen werden
// fun_2 kann schon aufgerufen werden
```

```
fun_2();

if($bool){
    function fun_1(){
        echo" $bool ist wahr";
    }
}

function fun_2(){
    // fun_2 kann jederzeit aufgerufen werden

    if($bool)
        fun_1(); // so laesst sich fun_1 sicher aufrufen
}
```

Mittels des optionalen Befehls return kann die Funktion beendet werden und ein Wert an die aufrufende Umgebung zurückgegeben werden.
Da einer Funktion als Parameter immer Werte übergeben werden, bleibt der Wert des Parameters außerhalb der Funktion unverändert. Will man den Parameter außerhalb der Funktion ändern, so muss er als Referenz übergeben werden.

```
function fun(&$string){
    $string.="Welt"
}
$str="Hallo ";
fun($str);      // hier kein '$'
echo"$str";     // Ausgabe: "Hallo Welt"
```

Bei der Definition einer Funktion besteht die Möglichkeit skalare Parameter mit Vorgabewerten zu belegen, die die Variablen annehmen, falls beim Aufruf kein Wert angegeben ist.

```
function fun($txt="leerer String"){
    echo"$txt";
}
fun();              // gibt "leerer String" aus
fun("Hallo Welt");  // gibt "Hallo Welt" aus
```

Ab PHP4 können auch Funktionen mit variabler Anzahl der Parameter definiert werden.

### 2.2.9 Klassen

Eine Klasse ist eine Sammlung von Funktionen und Variablen. Klassen werden wie folgt definiert:

```
class Fahrzeug{
    var $max_geschw;
    var $preis;

    /* Konstruktor */
    function Fahrzeug($max_geschw=NULL, $preis=NULL){
```

```
        $this->preis=$preis;
        $this->max_geschw=$max_geschw;
    }

    function bewegen(){
        ...
    }
}
```

Um ein Objekt zu erzeugen, benutzt man den new-Operator. Das Nutzen der Funktionen und Variablen des Objektes erfolgt über " -> ". Um in einer Funktion einer Klasse auf Variablen der Klasse zuzugreifen, muss $this benutzt werden, da PHP sonst davon ausgeht dass es sich um eine neue Variable handelt.
Ein Konstruktor ist eine Funktion, die automatisch beim Erzeugen des Objektes mit new aufgerufen wird. Ab PHP4 heißt der Konstruktor immer wie die Klasse, in der er vorkommt.

```
$auto = new Fahrzeug(200,20000);
$auto->max_geschw=250;
$auto->bewegen();
```

Mit extends können Klassen abgeleitet werden. Hierbei erbt natürlich die abgeleitete Klasse alle Variablen und Funktionen der Basisklasse.

```
class Auto extends Fahrzeug{
    var $tueren;
    var $sonderausstattung="Klima";

    function Auto($max_geschw=NULL, $preis=NULL, $tueren=NULL)
        $this->tueren=$tueren;

        // Konstruktor der Basisklasse
        parent::Fahrzeug($max_geschw, $preis);

        /*
        alternativ:
            parent::max_geschw=$max_geschw;
            parent::preis=$preis;
        */
    }
}
```

Besitzt eine Klasse keinen Konstruktor, so wir der Konstruktor der Basisklasse aufgerufen. Der Aufruf erfolgt allerdings nicht automatisch, sondern muss vom Programmierer vorgenommen werden.
Mit parent ist es möglich, Funktionen und Variablen aus der Basisklasse im Code der abgeleiteten Klasse zu nutzen ohne den Namen der Basisklasse zu verwenden. Mit " :: " ist es möglich auf Funktionen in Klassen zuzugreifen, die noch keine Instanzen haben.

# Kapitel 3

# Webanwendungen mit PHP

## 3.1 Integration von PHP in HTML

Es existieren spezielle HTML-Tags, die den Parser dazu veranlassen, den Text einer Datei zwischen öffnendem und schließendem Tag zu interpretieren. Code außerhalb der Tags wird vom Parser übergangen. Es gibt vier verschiedene Tags, die den PHP-Code kennzeichnen. Auf jedem Webserver verfügbar sind

- 1. <?PHP . . . ? >
- 2. <script language="php">. . .</script>

In der Konfigurationsdatei php.ini können noch zwei weitere Darstellungen aktiviert bzw. deaktiviert werden.

- 3. <? . . . ?>
- 4. < % . . . % >

### 3.1.1 Formulare

Die einzige interaktive Schnittstelle, die HTML bietet, sind Formulare. Nur so können mit HTML Eingaben vom Benutzer an den Server übermittelt und einem Skript zur Verfügung gestellt werden.
Um dies zu realisieren setzt man das action-Attribut des Formulartags auf den Dateinamen der auszuwertenden Datei. Den Formularfeldern gibt man über das name-Attribut einen Namen. Der Browser übergibt beim Abschicken die Namen und Werte der Felder. PHP vereinfacht nun den Zugriff auf diese Werte, indem für jeden übermittelten Namen eine globale Variable mit dem zugehörigen Wert angelegt wird.
Folgendes Beispiel zeigt eine Eingabeseite in HTML und eine Ausgabeseite mit PHP:

```
/*------- Eingabe.html----------------*/
    <html>
    <head>
        <title>Eingabe</title>
    </head>
    <body>
```

```
        <div align="center">
            <form action="ausgabe.php4" method="post">
            <?
                // HTML-Formular mit zwei Textfeldern
            ?>
                Feld1: <input name="feld1" size="60" maxlength="60"><br>
                Feld2: <input name="feld2" size="60" maxlength="60"><br>
                <input type="submit" value="OK">
                <input type="reset" value="Abbrechen">
                <?
                    // Button zum Abschicken und Abbrechen
                ?>
            </form>
        </div>
    </body>
    </html>


/*------ Ausgabe.php----------------*/

<html>
<head>
    <title>Ausgabe</title>
</head>
<body>
    <?php
        echo"<h1> Folgende Eingaben wurden gemacht: </h1>";
        echo"Feld 1: $feld1 ";
        echo"Feld 2: $feld2 ";
        // Ausgabe der Variablen $feld1 und $feld2
    ?>
</body>
</html>
```

Das Attribut method mit dem Wert post sorgt dafür, dass der Browser die URL mit den Variablennamen nicht in der Adresszeile des Browsers angezeigt. Der Wert get ändert dies. Man kann mit hidden Felder, HTML-Formularfelder die im Browser nicht sichtbar sind, Formulare auch dazu nutzen Variablen an ein PHP-Programm zu übergeben. Der Programmierer sollte sich aber darüber im klaren sein, dass der Benutzer die Werte dieser Variablen ändern kann.

### 3.1.2 Werte übergeben

Möchte man ein PHP-Programm starten und ihm Variablen mit Werten übergeben kann man dies zum einen wie oben mit Hilfe eines Formulars machen. Es ist aber auch möglich anstatt eines Formulars einen Link zu benutzen.
Dies ist die Syntax eines HTML-Links auf die Datei datei.php:

```
<a href:="datei.php" > Linktext
```

Um nun die Variablen an datei.php zu übergeben erweitert man den Link um “ ? “gefolgt vom Variablennamen sowie “ = “und dem Wert. Mehrere Variablen werden durch “ & “getrennt.

```
<a href:="datei.php?var1=5&var2=<?=\$value ?>" > Linktext
```

Hier wird natürlich $value bei der Interpretation durch den Parser durch den Wert von $value ersetzt.
Die Variablenübergabe mit Links hat den Nachteil, dass sie, wenn das Ziel nicht in einem Frame dargestellt wird, die URL mit Variablen und Werten in der Adresszeile des Browsers darstellt und es so dem Benutzer sehr einfach macht Werte zu manipulieren. Dies sollte vom Programmierer immer berücksichtigt werden.

## 3.2 PHP und Javascript

Ein Vergleich zwischen PHP und Javascript erübrigt sich, da Javascript im Gegensatz zu PHP clientseitig arbeitet. Viel interessanter in Bezug auf die Web-Programmierung ist die Kombination beider Sprachen. So bietet PHP die Möglichkeit, Javascript-Funktionen zu ändern bevor sie an den Client gesendet werden. PHP hat aber keine Möglichkeit, während der Interpretation der HTML-Datei im Browser, noch Einfluss zu nehmen. Hier ist es mit Javascript möglich, auf Benutzereingaben wie Mausbewegungen zu reagieren und evtl. ein neues PHP-Skript zu laden.

### 3.2.1 Sessionmanagement

Ein wichtiger Bestandteil von dynamischen Webseiten ist das Vorhalten von Informationen über mehrere Seiten hinweg. Da der Webserver die Verbindung zu einem Client sofort nach dem Senden der Antwort auf seine Anfrage trennt, stellt dieses Vorhalten der Informationen eine der grössten Aufgaben bei der Entwicklung von interaktiven und dynamischen Webseiten dar.
Ein Beispiel für eine Anwendung wäre ein Web-Email-Dienst. Zunächst muss hier der Benutzer von Beginn der Nutzung über mehrere Seiten hinweg bis zum Ende eindeutig identifiziert werden. Weiter muss die Darstellung der Seiten sich auch nach Eingaben des Benutzers richten. Es dürfen also keine wichtigen Eingaben des Benutzers verloren gehen.
Eine Lösung des Problems ist die Implementierung eines Sessionmanagements. Bis PHP4 musste man ein Sessionmanagement selbst implementieren. PHP4 bietet eine eigene Implementation.

#### Sessionmanagemant mit PHP 4

Um einen Benutzer wiederzuerkennen ist einen eindeutige Id nötig. Benutzt der Benutzer die Startseite, nutzt man *boolean session_start()*, um einen neue Session_ID zu vergeben. Die Session_ID kann nun als Cookie auf dem Client hinterlegt werden. Bei jedem weiteren Aufruf von session_start() kann nun die Session wieder hergestellt werden. Beim Start der Session wird eine temporäre Datei auf dem Server angelegt, die es nun erlaubt mit der Session_ID dort Variablen nachhaltig zu sichern. Hierfür wird mit dem assoziativen Array $_SESSION eine Variable durch Angabe des Variablennamens als Schlüssel initialisiert und registriert. Nach der Interpretation des Skriptes werden alle registrierten Variablen in die Datei geschrieben.
Da der Benutzer auf seinem Rechner lediglich die Session_ID sieht, hat er auch nicht die Möglichkeit, Werte von Variablen zu manipulieren wie es zum Beispiel mit hidden-Feldern möglich ist.
Wird nun mit session_start() eine Session wiederhergestellt, kann mit
$_SESSION[“Variablenname“] wieder auf die Variable zugegriffen werden. Folgendes Beispiel

startet eine Session und registriert eine Variable $zaehler mit Wert 0. Wird nun die Session wieder aufgerufen, so wird die Variable um 1 erhöht. Die Funktion *int isset($variable)* prüft lediglich, ob eine Variable schon existiert.

```
<?php
    session_start();

    if (!isset($_SESSION['zaehler'])) {
        $_SESSION['zaehler'] = 0;
    } else {
        $_SESSION['zaehler']++;
    }
?>
```

## 3.3 PHP und Datenbankmanagemantsysteme

### 3.3.1 Allgemeines

In Anwendungen spielen Datenbanken eine grosse Rolle. Ein Grossteil der anfallenden Daten wird mit Datenbankmanagementsystemen (DBMS) gespeichert und steht so mehreren Personen und Anwendungen zur Verfügung. Eine Beispielanwendung ist die Implementation eines Shopping Portals unter Verwendung aller Artikeldaten aus einem DBMS. So kann dem Kunden zum Beispiel die Verfügbarkeit eines Produktes aktuell mitgeteilt werden.
PHP unterstützt eine Vielzahl an DBMS. Zu den unterstützten Systemen zählen unter anderem Oracle, dBase, Direct MS-SQL, MySQL, DB2, PostgreSQL. Mit dem dbx-Modul hat PHP eine Abstraktionsschicht, die es ermöglicht, auf einige DBMS mittels einer einzigen Abfragekonvention zuzugreifen. Dies erleichtert zwar die Migration von einem DBMS zu einem andern, hat aber den Nachteil, dass einem nur die Schnittmenge der Funktionen aller Systeme zur Verfügung steht.
Eine Besonderheit die PHP im Umgang mit DBMS zur Verfügung, stellt sind persistente Datenbankverbindungen. Wird eine Datenbankverbindung aufgebaut, so überprüft PHP zunächst, ob nicht schon eine Verbindung zu dem gleichen Host mit demselben Benutzer und Passwort besteht. Ist dies der Fall, so wird diese Verbindung genutzt und keine neue Verbindung aufgebaut. Dies minimiert die Anzahl an Verbindungen zum Datenbankserver, was die Datenbankserver erheblich entlassten kann.

### 3.3.2 PHP & MySQL

Folgender Abschnitt soll das Zusammenspiel von Datenbankmanagementsystemen und PHP am Beispiel von MySQL[2] zeigen.
PHP stellt eine Reihe von Funktionen zur Verfügung, um Anfragen an den MySQL-Server zu senden und daraufhin die Ergebnisdatensätze sowie weitere Informationen auszulesen.
Zunächst muss jedoch eine Verbindung zum Datenbankserver hergestellt werden. Dies geschieht mit der Funktion *resource mysql_connect( string $dbServer, string $dbUser, string $dbPasswort).* Bei einer erfolgreichen Verbindung wird ein Identifier zurückgegeben sonst false.
Mit *mysql_close( resource $dbIdent )* wird die Verbindung mit dem Identifier dbIdent wieder geschlossen. Da ein DBMS mehrere Datenbanken enthalten kann, muss nun mit *boolean mysql_select_db( string $dbName [,resource $dbIdent])* eine Datenbank ausgewählt werden. Wird kein Identifier angegeben, so wird die zuletzt geöffnete Verbindung benutzt. Nun ist es möglich, mit

*resource mysql_query( string $query [,resource $dbIdent])* eine MySQL Befehlssequenz an das DBMS zu senden. Dabei wird ein Resultatszeiger auf den ersten Datensatz gesetzt. Das Beispiel soll zeigen, wie man eine Verbindung aufbaut und an eine Datenbank eine Anfrage stellt.

```
<?PHP
    $connection = mysql_connect( "server", "max", "blafasel");
        // Verbindung unter user max aufbeuen
    $ok = mysql_select_db("kunden", $connection);
        // Datenbank Kunden ausw{\"a}hlen
    $result = mysql_query("SELECT vorname, nachname FROM kunden");
        // Anfrage stellen
?>
```

Nun muss das Ergebnis der Anfrage noch vom Datenbankserver geholt werden. Mit *array mysql_fetch_row(resource $result)* wird der Datensatz vom Server eingelesen, auf den der Resultatszeiger zeigt. Nach dem Einlesen wird der Resultatszeiger auf den nächsten Datensatz gesetzt. Im folgenden Beispiel wird noch eine Funktion *die(string $str )* benutzt, um einen String auszugeben, falls ein Fehler auftritt.

```
<?php
    $host="server";
    $user="max";
    $password="blafasel";
    mysql_connect($host,$user,$password)
        or die("Konnte nicht mit Server verbinden");
    mysql_select_db("kunden")
        or die("Konnte nicht DB w{\"a}hlen");
    $result = mysql_query("SELECT vorname, nachname FROM kunden")
        or die("Anfragefehler");
        // hohle ersten Datensatz
    while($row = mysql_fetch_row($result)) {
        echo $row[0];   // erstes Ergebniselement
        echo $row[1];   // zweite Ergebniselement
        // gibt Zeile in HTML aus
        echo" $row[0] $row[1]<br>";
    }
?>
```

Mysql_fetch_row hat den Nachteil, dass die Reihenfolge der Arrayelemente von der Reihenfolge der selektierten Spalten in der Query abhängt. Ändert man diese Reihenfolge in der Query, so muss man dies eben auch im Skript machen. Abhilfe schafft hier *array mysql_fetch_array(resource $result [, int $result_typ])*. Der optionale Parameter result_typ gibt die Art des Rückgabewertes an. Mit MYSQL_ASSOC wird ein assoziatives Array mit den Spaltennamen als Schlüssel zurückgegeben. Mit MYSQL_NUM verhält sich mysql_fetch_array genauso wie mysql_fetch_row, und mit MYSQL_BOTH, was der Standardwert ist, hat man beide Möglichkeiten der Indizierung.

```
<?php
    $host="server";
    $user="max";
```

```
    $password="blafasel";
    mysql_connect($host,$user,$password)
        or die("Konnte nicht mit Server verbinden");
    mysql_select_db("kunden")
        or die("Konnte nicht DB w{\"a}hlen");
    $result = mysql_query("SELECT vorname, nachname FROM kunden")
        or die("Anfragefehler");
         // hohle ersten Datensatz
    while($row = mysql_fetch_array($result)) {
        echo $row["vorname"];   // gibt Spalte vorname aus
        echo $row["nachname"];  // gibt Spalte nachname aus
            // gibt Zeile in HTML aus
        echo" $row[0] $row[1]<br>";
    }
?>
```

Um Fehlermeldungen des Datenbankservers zu erhalten, gibt es die beiden Funktionen *string mysql_error([resource $dbIdent])* und *integer mysql_errno([resource $dbIdent])*, die die Fehlermeldung bzw. die Fehlernummer des auftretenden Fehlers zurückgeben.
Will man die Anzahl der Datensätze, die eine Anfrage erbracht hat, erfahren, so ist dies mit der Funktion *integer mysql_num_rows(resource $result)* möglich. Die Datensätze einer beliebigen MySQL-Query erhält man mit *integer mysql_affected_rows([resource $dbIdent])*. Häufig ist es nötig den Primärschlüssel eines gerade eingefügten Datensatzes auszulesen. Hat die Spalte des Primärschlüssels die AUTO_INCREMENT Eigenschaft, so ist dies mit *integer mysql_insert_id([ resource $dblink])* möglich. Um die Korrektheit des Ergebnises zu gewährleisten ist es wichtig, dass sichergestellt wird, dass zwischen dem Insert und mysql_insert_id keine weitere Inserts oder Updates ausgeführt wurden.
PHP hat für fast jede MySQL-Funktion eine Funktion. Somit ist es nicht verwunderlich, wenn es in PHP geschriebene Administrationsprogramme für MySQl-Datenbankserver gibt. Das populärste ist wohl phpMyAdmin[4].

## 3.4 HTTP-Authentifizierung mit PHP

Werden Anwendungen entwickelt, die z.B ein DBMS nutzen, oder unterschiedlichen Nutzern unterschiedliche Rechte zugestehen, so ist es notwendig den Benutzer eindeutig zu erkennen und dann Benutzername und Passwort zu verarbeiten. Ist PHP als Apachemodul[3] installiert ist es möglich mit der Funktion header() einen HTTP-Header, eine HTTP Anfangsinformation, an den Browser des Clients zu schicken, und ihn so zu veranlassen sein Passwort-Eingabe-Fenster zu öffnen. Hat der Nutzer die Eingabe getätigt, so wird die URL des PHP-Skriptes erneut aufgerufen und die Variablen $PHP_AUTH_USER, $PHP_AUTH_PW und $PHP_AUTH_TYPE enthalten die Daten. Folgender PHP-Code erzwingt eine Passwort-Eingabe.

```
<?php
    if(!isset($PHP_AUTH_USER)){
        header("WWW-Authenticate: Basic realm=\"My Realm\"");
        header("HTTP/1.0 401 Unauthorized");
        echo "Text bei druecken des Cancel-Button\n";
        exit; // Beenden
```

```
    }else{
        echo "Hallo $PHP_AUTH_USER.<P>";
        echo "Dies $PHP_AUTH_PW ist ihr Passwort.<P>";
    }
?>
```

Hat man nun zu jedem Benutzernamen das Passwort in einer Datenbank oder in einer Datei gespeichert ist es möglich den Benutzer zu identifizieren. Folgender PHP-Code zeigt die Kontrolle mit einer MySQL-Datenbank. Es ist auf dem DBMS-Server eine Datenbank user eingerichtet die eine Tabelle user_daten mit den Spalten username, passwort, vorname, nachname enthält. Die Datenbank-Verbindung wird mittels eines Satandard-Users hergestellt. Nach der Eingabe wird dann mittels des Benutzernamen und des Passworts der Vorname und der Nachname aus der Datenbank geholt.

```
<?php
    function authenticate($msg){
        // Unauthorized zurueckgeben
        Header("WWW-Authenticate: Basic realm=\"My Realm\"");
        Header("HTTP/1.0 401 Unauthorized");
        echo $msg;
        exit; // Beenden
    }

    function check_passwd($username, $passwort){
        // ueberpruefen von Benutzername und Passwort
        $host="server";
        $user="max";
        $password="blafasel";
        mysql_connect($host,$user,$password)
            or die("Konnte nicht mit Server verbinden");
        mysql_select_db("user")
            or die("Konnte nicht DB w{\"a}hlen");
        // Verbindung zu DBMS-Server und Datenbank herstellen
        $qry="SELECT vorname, nachname, passwort ";
        $qry.=" FROM user_daten where username like
        $qry.='$username' and passwort like '$passwort'";
        // Die Query hohlt vorname und nachname aus DB.
        // Spalten username und Passwort muessen unique
            sein
        $result = mysql_query($qry)
            or die("Anfragefehler");

        while($row = mysql_fetch_array($result)) {
            /* Benutzerdatensatz  hohlen
               existiert kein Benutzer mit dieser
               Kombination von username und passwort nimmt
               $row den Wert false an.
            */
        }
```

```
        return $row;
    }

    if(!isset($PHP_AUTH_USER) || !isset($PHP_AUTH_PW)){
        $msg="Dies ist ein geschuetzter Bereich ! <br>";
        $msg.="Zutritt nur mit Benutzername und Passwort";
        authenticate($msg);
    }

    if(isset($PHP_AUTH_USER) && isset($PHP_AUTH_PW)){
        $user=$check_passwd($PHP_AUTH_USER, $PHP_AUTH_PW);
        if(!$user){
            // Benutzer nicht vorhanden oder Eingabe
                falsch
            $msg="Passwort oder Benutzename falsch";
            authenticate($msg);
        }else{
            echo" Guten Tag $user[vorname] $user[nachname]";
        }
    }
?>
```

Zu beachten ist, das die Authentifizierung so nicht mit jeder Kombination von Webserver und PHP-Installation funktioniert. Weiter gilt die Authentifizierung nur für diese eine Seite. Will man Dateien oder Verzeichnisse schützen, so ist eine Server-Lösung notwendig.

# Kapitel 4

# Zusammenfassung

Die Entwicklung von Web-Anwendungen stellt höhere Anforderungen als nur das Anzeigen statischer Webseiten. Wie gesehen setzt PHP genau hier an und ermöglicht das Entwickeln komplexer Anwendungen mit einer Plattform und Standort unabhängigen Ausgabe, die ohne Installation von Zusatzsoftware bei den Anwendern, ein Browser ist überall installiert, funktioniert.

# Literaturverzeichnis

[1] http://www.php.net/ Seiten des PHP-Projektes

[2] http://www.mysql.com/ Seiten des Mysql-Projektes

[3] http://www.apache.org/ Seiten derApache Software Foundarion

[4] http://www.phpmyadmin.net/ Seiten des phpMyAdmin Projektes.