

Skriptum zum WIFI-Kurs „WEB-Development mit PHP und MySQL“

Markus Penz

Mils, September-Oktober 2001

PHP Version 4.0

Skriptum Version 0.5

Der Autor freut sich über Vervielfältigungen und Nachdrucke dieses Skriptums.
Fragen und Anregungen sind bitte an m.penz@inter.at zu richten.

Inhalt

1.	Einleitung	4
1.1.	Zu diesem Skriptum	4
1.2.	Zur Geschichte von PHP	4
1.3.	PHP ist Open-Source	4
1.4.	„LAMP“ und „WAMP“	4
1.5.	Die Funktionsweise von PHP	5
1.6.	PHP Download und Installation	6
1.7.	Konfiguration von PHP	6
2.	Sprachgrundlagen	8
2.1.	Hallo Welt!	8
2.2.	Grundlegender PHP-Syntax	8
2.3.	Variablen	9
2.4.	String-Variablen	9
2.5.	Arrays (Felder, Vektoren).....	10
2.6.	Typumwandlung	11
2.7.	Geltungsbereich von Variablen.....	11
2.8.	Variable Variablen	12
2.9.	Operatoren	13
2.10.	If	14
2.11.	While und Do..While	15
2.12.	For und Foreach.....	15
2.13.	Break und Continue.....	16
2.14.	Switch	16
2.15.	Include() und Require()	16
2.16.	Prozeduren/Funktionen.....	17
2.17.	Referenzen.....	19
2.18.	Vordefinierte Variablen	19
2.19.	Datenübertragung vom Client zum Skript	19
3.	Konzepte und Features anhand von Beispielen	21
3.1.	Counter	21
3.2.	Gästebuch.....	23
3.3.	Lotto	24
3.4.	Mail-Versenden.....	26
3.5.	Download-Liste.....	26
3.6.	Datei-Upload	27
3.7.	Datum und Uhrzeit.....	28
3.8.	Cookies	28
3.9.	HTTP-Authentifizierung.....	29
3.10.	News-Grabber	31
4.	Datenbankanbindung (an MySQL).....	32
4.1.	PHP, Datenbanken und SQL	32
4.2.	MySQL.....	33
4.3.	phpMyAdmin	33
4.4.	Datenbankanbindung in PHP.....	34
4.5.	Datenbankzugriff in einem Beispiel.....	35
4.6.	Bilder in Datenbanken ablegen	35

5.	Programmierung eines Forums in PHP	38
5.1.	Grundlagen	38
5.2.	Darstellung der Baumstruktur.....	38
5.3.	Einträge speichern	39
5.4.	Weitere Features des Forums	40
6.	Quellenverzeichnis.....	42

1. Einleitung

1.1. Zu diesem Skriptum

Das Skriptum soll parallel zur PHP- und MySQL-Dokumentation als Unterlage für die Kursteilnehmer dienen und sowohl Erklärungen als auch Beispiele bieten.

Die Texte sind zum Teil aus Büchern oder dem Internet entnommen, die Quellenangaben sind ganz am Ende zu finden.

1.2. Zur Geschichte von PHP

Eine erste Vorversion von PHP wurde bereits im Herbst 1994 von Rasmus Lerdorf in Perl für seine private Homepage geschrieben und lief unter dem Namen „Personal Home Page Tools“. Die andere Interpretation der Abkürzung PHP ist die offizielle „rekursive“ Version: „PHP Hypertext Preprocessor“.

1995 wurde der Parser neu geschrieben und unter dem Namen PHP/FI (FI steht für „Form Interpreter“, das Auswerten von HTML-Formularen) verteilt. Diese schon wesentlich fortgeschrittenere Version wurde im Sommer 1997 von Rasmus Lerdorf und Andi Gutmans mit PHP 3.0 zum professionellen Tool für dynamische Weblösungen und fand große Verbreitung im Internet. Die aktuelle Version ist 4.0 mit der eine neue Parser-Technologie namens Zend-Engine eingeführt wurde, am generellen Syntax hat sich damit aber nichts geändert.

Zend ist auch der Name des kommerziellen „Spin-Offs“ von PHP. Die Firma der PHP-Entwickler bietet Optimierungs- und Entwicklungsprogramme für PHP an, die Zend-Engine selbst blieb aber vollkommen kostenfrei.

1.3. PHP ist Open-Source

Das bedeutet, dass der Quellcode von PHP für jeden einsehbar ist und – was wohl für die meisten noch mehr zählt – dass es vollkommen legal runtergeladen, verteilt, modifiziert und verwendet werden kann, sei es für den privaten oder den kommerziellen Gebrauch. Die vollständige PHP-Lizenz ist auf www.php.net zu finden. Sicher ist das auch einer der Hauptgründe für die große Beliebtheit von PHP.

Einziger Nachteil ist die Problematik der Verantwortlichkeit und das fehlende Support, doch sei darauf verwiesen, dass sich auch große Firmen auf PHP verlassen und das Produkt millionenfach getestet wurde.

1.4. „LAMP“ und „WAMP“

Diese seltsamen Abkürzungen definieren die beliebtesten Webserver-Konfigurationen in Zusammenhang mit PHP. Da PHP zum Open-Source Bereich gehört, wird es natürlich oft gemeinsam mit dem freien Betriebssystem Linux verwendet. Als Webserver und Datenbank-Server können ebenfalls Open-Source Produkte herangezogen werden, wie Apache und MySQL.

Linux, Apache, MySQL und PHP wird gemeinsam als LAMP-Konfiguration bezeichnet, unter Windows wird daraus „WAMP“.

Jedoch ist PHP keinesfalls auf diese Web- und Datenbank-Server beschränkt. Die Server-Schnittstelle (SAPI) unterstützt derzeit Apache, Roxen, Java (servlet), ISAPI (für Microsoft IIS und PWS), AOLserver und die weitverbreitete Schnittstelle CGI. Die Liste der unterstützten Datenbanken ist sogar noch weit länger und lässt sich durch PHP-Extensions noch weiter verlängern: MySQL, mSQL, Oracle, PostgreSQL, Sybase, Microsoft SQL-Server sowie verallgemeinertes ODBC etc.

1.5. Die Funktionsweise von PHP

Die offizielle Abkürzung „PHP Hypertext Preprocessor“ definiert auch die Funktionsweise von PHP recht gut: Statt eine Clientanfrage direkt mit der Lieferung der entsprechenden Seite zu beantworten, gibt der Webserver alle Dokumente eines bestimmten Typs (z.B. mit der Endung „.php“) an ein Programm weiter. Dieses durchsucht den HTML-Code der Seite nach speziellen „Tags“. Wird dieses gefunden, so wird der nachfolgende Text nicht unverändert zurückgegeben, sondern zuvor als Skriptcode interpretiert. Der Skriptcode kann dann wieder Anweisungen enthalten, die eine Ausgabe bewirken, die dann ebenfalls zurückgegeben wird. Folgt ein entsprechendes Schluss-Tag, so schaltet der Parser wieder in den „Durchlaufmodus“.

Als einfaches Beispiel soll die Arbeit eines Kopisten beschrieben werden: Er hat nichts weiter zu tun, als alle weißen Blätter im „Eingang“ abzuschreiben und in den „Ausgang“ zu legen. Kommt jedoch ein rotes Blatt, so müssen die daraufstehenden Befehle ausgeführt werden. Ein solcher Befehl wäre z.B. „addiere zwei und zwei und schreibe es nieder“, also wird „vier“ niedergeschrieben und in den „Ausgang“ gelegt.

Der PHP-Parser arbeitet im Grunde gleich und liefert die Ausgabe zurück an der Webserver, der diese über das Internet an den Client weitergibt. Der Client erhält keine „roten Blätter“ sondern sieht nur das Resultat der Befehle, die dort standen. Damit PHP die Anweisungen von reinem HTML-Code unterscheiden kann, werden diese von den sogenannten PHP-Tags umschlossen, die sich an die normalen HTML-Tags anlehnen und in vier verschiedenen Varianten vorliegen (die aber nicht zwingend alle verstanden werden müssen, das hängt von der PHP-Konfiguration ab, mehr dazu später):

- die (XML-konforme) Langform: `<?php ... ?>`
- die Kurzform: `<? ... ?>`
- die Skript-Version: `<script language="php"> ... </script>`
- die ASP-Version: `<% ... %>`

Dazwischen steht der zu interpretierende PHP-Code. Auf korrektes HTML muss natürlich nicht mehr geachtet werden, da der Client ohnehin nichts von diesen PHP-Tags sieht (wenn doch, läuft irgendetwas schief). Z.B. soll ein PHP-Skript die Schriftfarbe generieren:

<font color="<? ... ?>">PHP-gefärbter Text

1.6. PHP Download und Installation

Die aktuelle PHP Version für verschiedene Plattformen bekommt man unter www.php.net. Der PHP-Source-Code für Linux/Unix muss vor Verwendung erst kompiliert werden (liegt allerdings auch bei vielen Linux-Distributionen bei). Für Windows stehen derzeit zwei Installationspakete bereit: ein kleiner Installer (CGI-Version, keine Extensions) und ein umfangreicheres ZIP-Archiv (CGI- und Server-API-Versionen, mit vielen Extensions).

Der Installer muss nur aufgerufen und die Anweisungen befolgt werden, innerhalb weniger Minuten kann man so PHP auf einem Windows-System mit Webserver zum Laufen bringen.

Das ZIP-Archiv enthält eine Textdatei mit Hinweisen zur Installation namens „install.txt“. Der erste Abschnitt behandelt die Installation des ersten Paketes, erst im zweiten wird die manuelle Einrichtung erläutert. Wie schon erwähnt stehen hier zwei Versionen von PHP bereit. Die erste ist eine einfache Executable (php.exe), die über die CGI-Schnittstelle vom Webserver angesprochen wird und den PHP-Code interpretiert. Die zweite liegt in Form von mehreren sog. SAPI-Modulen vor (für verschiedene Webserver), die dann den Server gewissermaßen erweitern. Diese Version liefert wesentlich bessere Performance, es wird jedoch auch davor gewarnt, dass sie noch nicht voll ausgetestet ist.

Für das Testen von PHP-Skripts kann man sich also meist auf die einfache Installer-Version beschränken, das Betreiben eines eigenen öffentlichen Web-servers ist ohnehin meist der Ausnahmefall und man muss sich mit der vom Provider angebotenen PHP-Installation begnügen.

1.7. Konfiguration von PHP

Sowohl unter Linux/Unix als auch unter Windows bietet PHP die Möglichkeit, die Konfiguration über die Datei „php.ini“ zur Laufzeit zu verändern. Diese liegt unter Windows im Verzeichnis des Betriebssystems („Windows“ oder „WinNT“). Jeder Parameter wird darin in eine eigene Zeile geschrieben, Kommentare mit Semikolon eingeleitet.

Es sollen hier nur die wichtigsten Einstellungen behandelt werden, die Datei selbst ist ausführlich kommentiert und die vollständige Dokumentation ist im PHP-Manual enthalten.

<i>engine</i>	schaltet PHP ein oder aus – wird es deaktiviert, so werden alle Skripte direkt an den Client weitergegeben, ohne interpretiert zu werden.
<i>short_open_tag</i>	erlaubt es <? anstatt nur <?php und <script> zu verwenden
<i>asp_tags</i>	erlaubt zusätzlich <% Tags (die sonst in ASP verwendet werden), das ist z.B. praktisch, wenn FrontPage für das

	Erstellen der Seiten verwendet wird, da die Code-Teile dann ausgeblendet werden
<i>safe_mode</i>	aktiviert zusammen mit einigen anderen Einstellungen einen Sicherheitsmodus, was das Ausführen von Programmen am Server betrifft
<i>expose_php</i>	bestimmt, ob von außen erkennbar ist, dass PHP am Server läuft (z.B. im HTTP-Header)
<i>max_execution_time</i>	legt die Zeit in Sekunden fest, die ein Skript maximal zur Ausführung benötigen darf und verhindert damit z.B. Endlosschleifen, die den Server lahm legen
<i>memory_limit</i>	beschränkt den Speicher, den ein Skript beanspruchen darf
<i>error_reporting</i> und <i>display_errors</i>	bestimmen die Art und den Umfang der Fehlerausgabe; v.a. während der Entwicklungszeit sollte man alle Fehler ausgeben lassen
<i>file_uploads</i>	steuert mit zwei weiteren Parametern den HTTP-File-Upload
<i>extension</i>	sollen Erweiterungen geladen werden, kann man einfach den Dateinamen angeben – der Pfad folgt aus dem Parameter <i>extension_dir</i>
<i>SMTP</i>	bei Windows-Systemen muss hier ein SMTP-Server angegeben werden, wenn via PHP auch Mails versendet werden sollen
<i>sendmail_from</i>	gibt die Standard-Absenderadresse für Mails an, die PHP verschickt

2. Sprachgrundlagen

2.1. Hallo Welt!

Um den Zugriff auf PHP-Seiten über das Internet zu ermöglichen, müssen diese natürlich im Webroot-Verzeichnis abgelegt werden. Damit der Webserver die Skripte auch zur Ausführung an den PHP-Parser weitergibt, muss für die entsprechenden Verzeichnisse die Ausführung von Skripten gestattet werden. Das erste Testskript soll nichts anderes tun, als über PHP-Ausgabe „Hallo Welt!“ auf die Seite zu schreiben. Die entsprechende Anweisung ist sehr einfach:

```
<? print "Hallo Welt!"; ?>
```

Mit einem Browser kann nun z.B. unter „http://localhost/hallowelt.php“ nachgeprüft werden, ob das PHP-Skript korrekt ausgeführt wird. Betrachtet man den Quelltext, wird man feststellen, dass zwar von PHP aus alles passt, die Seite selbst jedoch kein gültiges HTML ist. Um also eine richtige HTML-Seite zu erstellen, müsste das Beispiel so aussehen:

```
<html>
<head><title>Erster PHP-Test</title></head>
<body>
<? print "Hallo Welt!"; ?>
</body>
</html>
```

2.2. Grundlegender PHP-Syntax

Der PHP-Syntax lehnt sich stark an C an, es wurden jedoch auch Elemente aus Perl und Java übernommen. Anweisungen werden immer mit einem Strichpunkt beendet. Das schließende PHP-Tag beendet eine Anweisung ebenfalls (ein Strichpunkt kann aber dennoch nicht schaden). Folgende Anweisungen sind deshalb identisch:

```
<? print "Hallo Welt!"; ?>
<? print "Hallo Welt!" ?>
```

Sollte man wie im folgenden Beispiel den Strichpunkt vergessen, so tritt der Fehler bei der nachfolgenden Anweisung auf:

```
<?
print "Hallo Welt!<br>"
print "Hier bin ich.";
?>
```

Die Fehlerangabe erfolgt bei den Standardeinstellung als HTML-Ausgabe. Es wird der Name des betroffenen Skriptes und die Zeile, die den Fehler enthält

angegeben. Es wird deshalb empfohlen, mit einem Texteditor zu arbeiten, der die Zeilennummer angibt.

Da das Anweisungsende durch einen Strichpunkt (und nicht wie bei Basic durch einen Zeilenumbruch) definiert wird, kann man auch mehrere Anweisungen in eine Zeile schreiben. Aus Gründen der besseren Lesbarkeit ist dies aber nur selten wirklich sinnvoll:

```
<?
print "Hallo Welt!<br>"; print "Hier bin ich.";
?>
```

Kommentare können entweder wie in C zeilenweise mit // oder blockweise mit /* ... */ markiert werden. Zudem sind zeilenweise Kommentare wie in der Unix-Shell erlaubt, die mit # eingeleitet werden.

2.3. Variablen

PHP unterstützt insgesamt fünf Variablentypen: Integer (Ganzzahlen), Double (Fließkommazahlen), String (Zeichenketten), Array (indiziert und assoziativ) und Object (Objektreferenzen).

Der Name von Variablen beginnt immer mit einem Dollarzeichen (\$) und ist case-sensitive.

Der Typ einer Variable wird in PHP üblicherweise nicht vom Programmierer bestimmt, sondern wird erst zur Laufzeit von PHP ausgewählt, abhängig vom Zusammenhang, in welchem die Variable benutzt wird. Um eine Variable zu initialisieren, wird ihr einfach ein Wert zugewiesen:

```
$a = 42; // Integer
$a = -3.14; // Double
$a = 1.2e2; // Double (1.2 mal 10 hoch 2 = 120)
$a = "Hallo Welt!"; // String
```

2.4. String-Variablen

Strings können von zwei verschiedenen Arten von Begrenzungszeichen umschlossen werden:

Verwendet man doppelte Anführungszeichen ("), so werden Variablen innerhalb des Strings ausgewertet (wobei es Einschränkungen beim Parsen gibt, denn PHP muss feststellen können, wo der Variablenname aufhört und der normale String weitergeht):

```
$a = "Welt!";
$b = "Hallo $a";
print $b;
```

Zur Ausgabe von speziellen Zeichen, kann das Backslash-Zeichen (\) verwendet werden:

\n	neue Zeile (line feed, ASCII 10)
\r	Wagenrücklauf (carriage return, ASCII 13)
\t	Tabulator (ASCII 9)
\\	Backslash
\\$	Dollarzeichen
\"	doppelte Anführungszeichen

Werden einfache Anführungszeichen (') verwendet, werden weder Variablennamen (eingeleitet durch \$), noch Sonderzeichen (eingeleitet durch \, mit der Ausnahme von \\ und \') interpretiert. Innerhalb von einfachen Anführungszeichen können also doppelte normal verwendet werden und umgekehrt.

Seit PHP 4.0 ist auch der sog. „here doc“-Syntax erlaubt, um Strings zuzuweisen. Dabei wird dem String ein Marker (ein alphanumerischer Ausdruck, wie ein Variablenname) eingeleitet durch „<<<“ zugewiesen. Dann wird der Text inklusive Zeilenumbrüchen eingelesen, bis in der *ersten* Textspalte der angegebene Marker und damit auch das Anweisungsende folgt. Innerhalb des eingelesenen Strings werden alle Sonderzeichen gelesen und Variablennamen ausgewertet:

```
$a = <<<ENDE_DES_STRINGS
Hier kommt der Text rein, mit " und ' und \<br>
Auch werden hier Variablennamen ausgewertet.
ENDE_DES_STRINGS;

print $a;
```

Um Zeichenketten zu verbinden, werden sie mit einem Punkt (.) aneinandergestellt. + oder & funktionieren nicht.

```
$a = "Hallo" . "Welt!";
$c = $a.$b;
```

2.5. Arrays (Felder, Vektoren)

Die Elemente eines Arrays können in PHP entweder über einen Index (indiziertes Array, Vektor) oder über einen Key-String (assoziatives Array) angesprochen werden, letztendlich gibt es aber keinen Unterschied zwischen den beiden Array-Varianten. Der Index (bzw. Key) wird in eckigen Klammern hinter der Array-Variable angeführt. Um ein Array zu erstellen, kann man entweder die list() oder array() Funktionen benutzen, oder die Werte können einfach explizit zugewiesen werden:

```
$a[0] = 10;
$a[1] = 20;
$b["foo"] = "bar";
```

Wird einfach eine leere Klammer verwendet, so werden die Elemente automatisch am Ende des Arrays angefügt, wobei der Index bei 0 beginnt.

```
$a[] = 10;  
$a[] = 20;
```

Bei mehrdimensionalen Arrays wird einfach ein zusätzliches Klammernpaar hinzugefügt:

```
$a[4][2] = -34.5;
```

2.6. Typumwandlung

Hier folgt PHP dem C-Syntax: Vor der umzuwandelnden Variable wird der neue Typ in runde Klammern geschrieben:

```
$a = (int) 3.2; // $a ist ein Integer und es wird automatisch gerundet
```

Folgende Umwandlungen sind möglich:

(int), (integer)	Umwandlung in Integer-Wert
(real), (double), (float)	hin zu Double
(string)	hin zu String
(array)	hin zu Array
(object)	Wandlung zum Objekt

Die gleiche Umwandlung kann auch mit der Funktion `settype()` erreicht werden. `gettype()` liefert dagegen den Typ einer Variable zurück.

2.7. Geltungsbereich von Variablen

Variablen können immer nur innerhalb einer Prozedur verwendet werden, folgendes Skript wird daher keine Ausgabe haben (mehr zu Prozeduren in PHP etwas später):

```
$a = 1; // globaler Bereich  
function test () {  
    print $a; // Referenz auf einen lokalen Variablen-Bereich  
}  
test ();
```

Man kann die Variable innerhalb der Prozedur jedoch als „global“ definieren:

```
$a = 1; $b = 2; // globaler Bereich  
function test () {  
    global $a, $b;  
    print $a + $b;  
}
```

```
test ();
```

Die andere Möglichkeit besteht darin, auf das spezielle `$GLOBALS` Array zuzugreifen, das die Werte aller Variablen des globalen Bereichs zusammenfasst:

```
$a = 1; $b = 2; // globaler Bereich
function test () {
    print $GLOBALS["a"] + $GLOBALS["b"];
}
test ();
```

Es gibt noch die Möglichkeit, Variablen für eine Prozedur als *static* zu deklarieren. Das bedeutet, dass sie ihren Wert bei neuerlichem Ausführen der Prozedur behält:

```
function test () {
    static $a = 0;
    $a = $a + 1;
    print $a;
}
test (); // jetzt ist $a = 1
test (); // jetzt ist $a = 2
```

Die Variable kann mit *static* mit einem Anfangswert initialisiert werden, dies ist aber nicht zwingend erforderlich.

2.8. Variable Variablen

Das Konzept von variablen Variablen erscheint anfangs vielleicht verwirrend, ist im Grunde aber sehr einfach und äußerst praktisch. In manchen Fällen kann es erforderlich sein, eine Variable zu verwenden, deren Namen man erst zur Laufzeit kennt. Das bedeutet, dass der Variablenname selbst eine Stringvariable ist. Die Verwendung solcher variablen Variablen ist sehr einfach:

```
$a = "var"; // normale Stringvariable
$$a = 5; // der Variable mit dem Namen „var“ wird ein Wert zugewiesen
```

Soll die variable Variable in einem String verwendet werden, muss die innere Variable mit geschwungenen Klammern umschlossen werden, da sonst ein Dollarzeichen und der Wert der inneren Variable ausgegeben wird:

```
print "Das Ergebnis ist: $$a"; // führt zu: $var
print "Das Ergebnis ist: ${$a}"; // führt zu: 5
```

Solche Klammerungen sind auch bei Arrays notwendig, wo ansonsten nicht klar ersichtlich ist, auch welche Variable sich der Index bezieht.

2.9. Operatoren

Die normalen arithmetischen Operatoren (+, -, *, /) arbeiten gleich wie in jeder andere Sprache, der Modulus (Rest einer Division) wird durch % repräsentiert.

Der Zuweisungsoperator = sollte nicht als „ist gleich“ verstanden werden, sondern besser als „wird gesetzt auf den Wert von“. Er liefert immer den zugewiesenen Wert zurück, somit sind folgende Konstruktionen möglich:

```
$a = ($b = 4) + 5; // ist $a = 9 und $b = 4
```

Es gibt auch sog. kombinierte Operatoren, die eine Operation an einer Variable durchführen und ihr gleich das Ergebnis zuweisen:

```
$a += 5; // $a wird um 5 erhöht  
$b .= "Welt"; // bei $b wird der String "Welt" angehängt
```

Damit verwandt sind die Inkrementierungs- bzw. Dekrementierungsoperatoren, die den Wert einer numerischen Variable um eins erhöhen bzw. vermindern und je nach Position vor oder nach der Variable den Wert danach oder davor zurückgeben.

```
$b = $a++; // $b bekommt den Wert von $a und erst dann wird $a erhöht  
$b = ++$a; // $a wird erhöht und $b bekommt den neuen Wert von $a  
$a--; // sie können natürlich auch alleine stehen
```

Es gibt eine ganze Reihe von Bit- und logischen Operatoren, die im Manual näher erläutert werden, hier soll nicht darauf eingegangen werden.

Vergleichsoperatoren vergleichen Ausdrücke und geben entweder „wahr“ (Boolean true, in PHP alles ungleich 0) oder „falsch“ (Boolean false, in PHP gleich 0). Die Boolean-Werte true und false werden in PHP auch durch zwei Konstanten repräsentiert.

```
$a == $b // gleich: true, wenn $a und $b den gleichen Wert haben  
$a === $b // identisch: true, wenn $a und $b den gl. Wert und Typ haben  
$a != $b // ungleich nach Werten  
$a < $b // kleiner als  
$a > $b // größer als  
$a <= $b // kleiner-gleich  
$a >= $b // größer-gleich
```

Ein weiterer Vergleichsoperator ist der „?:“-Trinitätsoperator. Er überprüft, ob ein Ausdruck true oder false ist und gibt je nach Entsprechung den ersten oder zweiten zusätzlichen Ausdruck zurück. Beide der folgenden Varianten führen zum selben Ergebnis:

```
print ($a == $b) ? (5) : (10);  
($a == $b) ? (print 5) : (print 10);
```

Der Fehler-Kontroll-Operator @ verhindert, wenn er vor einen Ausdruck gestellt wird, eine Fehlerausgabe:

```
@print $a / 0;
```

2.10.If

Die übliche if Struktur gibt es natürlich auch in PHP. Wenn der hinter if in runden Klammern angegebene Ausdruck true ist, wird der folgende, von geschweiften Klammern (wenn es mehrere Anweisungen sind) umgebene Block ausgeführt, ansonsten übergangen. In PHP gibt es keine festen Regeln bezüglich dem Setzen von Zeilenumbrüchen, man sollte aber immer auf gute Lesbarkeit achten:

```
if ($a == 2) { print "Variable hat den Wert zwei."; }
```

ist das gleiche wie

```
if ($a == 2) {  
    print "Variable hat den Wert zwei.";  
}
```

Soll ein bestimmter Block immer ausgeführt werden, wenn eine if-Bedingung nicht eintritt, kann nach if ein else-Block folgen:

```
if ($a == 2) {  
    print "Variable hat den Wert zwei.";  
}  
else {  
    print "Variable hat einen anderen Wert.";  
}
```

Sollen, nachdem if gescheitert ist, noch andere Ausdrücke getestet werden, so kann man Blöcke mit elseif einleiten. Diese werden nur ausgeführt, wenn alle vorigen Bedingungen false waren.

```
if ($a == 2) {  
    print "Variable hat den Wert zwei.";  
} elseif ($a == 3) {  
    print "Variable hat den Wert drei.";  
} elseif ($a == 4) {  
    print "Variable hat den Wert vier.";  
} else {  
    print "Variable hat einen anderen Wert.";  
}
```

Man kann innerhalb eines if-Blocks auch das PHP-Skript abschließen und HTML-Code einfügen, solange der Block später korrekt abgeschlossen wird:

```
if ($a == 2) {  
    ?>Variable hat den Wert zwei.<?  
}
```

2.11. While und Do..While

Eine weit einfachere Struktur ist while. Die nachfolgende Anweisung (oder der geklammerte Block) wird so lange immer wieder ausgeführt, bis der Testausdruck true wird. Ist der Testausdruck von Beginn an true, wird der while-Block nie ausgeführt.

```
while ($a < 10) {  
    $a++;  
}
```

Bei do..while hingegen werden die Anweisungen in jedem Fall mindestens einmal durchlaufen und erst am Ende geprüft. Deshalb ist auch der Syntax leicht verändert:

```
do {  
    $a++;  
} while ($a < 0);
```

2.12. For und Foreach

Die for-Schleife zählt üblicherweise eine Laufvariable mit jedem Durchlauf weiter und bricht bei einer bestimmten Bedingung ab. Dem for-Befehl folgen drei Ausdrücke: der erste wird in jedem Fall einmal zu Beginn ausgeführt (meist um die Laufvariable zu initialisieren), der zweite wird bei jedem Durchlauf (auch ganz zu Beginn) überprüft - bei false wird abgebrochen, bei true die Schleife fortgefahren und der dritte wird am Ende jedes Durchlaufs ausgeführt (meist um die Laufvariable zu erhöhen). In der Anwendung kann das dann z.B. so aussehen:

```
for ($i = 1; $i <= 10; $i++) {  
    print $i;  
}
```

Foreach bietet eine einfache Möglichkeit, immer alle Elemente eines Arrays zu durchlaufen. Innerhalb der Schleife kann dann auf den Wert und auch den Key (bei assoziativen Arrays) des Elements zugegriffen werden:

```
foreach ($arr as $value) {  
    print "Wert: $value<br>";  
}
```

```

}

foreach ($arr as $key => $value) {
    print "Schlüssel: $key, Wert: $value<br>";
}

```

2.13. Break und Continue

Diese Befehle werden innerhalb einer Schleife verwendet, um diese entweder abubrechen (break) oder nur den aktuellen Durchgang abubrechen und mit dem nächsten zu beginnen (continue). Break funktioniert mit for, foreach, while, do..while und switch und kann als optionales numerisches Argument die Anzahl der abubrechenden Schleifen erhalten (wenn diese geschachtelt sind). Continue macht nur bei for, foreach, while und do..while Sinn und kann ebenfalls mit Parameter aufgerufen werden.

2.14. Switch

Switch ist gleichbedeutend mit einer Reihe von if-elseif-Anweisungen, in welcher immer wieder der selbe Ausdruck mit einem Wert verglichen wird. Wichtig ist es, nach den Anweisungen in einem case-Teil nicht den Befehl break zu vergessen, sonst werden auch alle folgenden Anweisungen innerhalb des ganzen switch-Blocks ausgeführt.

```

switch ($i) {
    case 0:
        print "i ist gleich 0";
        break;
    case 1:
        print "i ist gleich 1";
        break;
    case 2:
        print "i ist gleich 2";
        break;
    default:
        print "i ist weder 0, 1 noch 2";
}

```

2.15. Include() und Require()

Beide Anweisungen lesen die angegebene Datei ein und binden sie in das Skript ein. Dabei wird vom PHP-Modus in den HTML-Modus gewechselt, d.h. innerhalb der eingebundenen Datei müssen Skriptteile erst wieder mit den PHP-Tags eingeleitet werden.

Der Unterschied zwischen require() und include() ist, dass require() in jedem Fall ausgeführt wird, auch wenn es innerhalb eines if-Blockes steht, dessen Testbedingung false ergibt.

Mit `include()` kann man zudem wie mit einer Funktion arbeiten. Wenn innerhalb des eingebundenen Skripts die `return`-Anweisung verwendet wird, wird die Einbindung an dieser Stelle abgebrochen und der Wert zurückgegeben. Mit `require()` kann man mit `return` zwar abbrechen, eine Rückgabe ist aber nicht möglich. Hier ein kleines Beispiel:

Eine Datei „test.inc.php“:

```
<?
print "Vor return.";
return 5;
print "Nach return.";
?>
```

Ein Skript im selben Verzeichnis könnte Folgendes enthalten:

```
$a = include("test.inc.php");
```

Man sollte die eingebundenen Skripte ebenfalls mit „.php“ enden lassen. Zur korrekten Ausführung ist dies zwar nicht nötig, sollte aber jemand den Namen der eingebundenen Skripte herausfinden oder erraten, kann er die Datei betrachten, ohne dass diese zuvor von PHP interpretiert wurde, d.h. der ganze Code liegt offen.

Es gibt noch zwei Varianten dieser speziellen Anweisungen: `include_once()` und `require_once()`. Sie arbeiten im Prinzip genau gleich, nur dass eine Datei in jedem Fall nur einmal eingebunden wird, auch wenn die Anweisung öfters erfolgt. Dies macht z.B. Sinn, wenn man Dateien mit zusätzlichen Prozeduren einbindet, die sich selbst u.U. wieder gegenseitig einbinden. Mit `include()` oder `require()` könnte das leicht zu einem Fehler führen, da Prozeduren doppelt definiert werden. Solche Fehler lassen sich natürlich auch ganz vermeiden.

2.16. Prozeduren/Funktionen

In PHP wird kein Unterschied zwischen Prozeduren mit Rückgabewert und solchen ohne gemacht. Alle Prozeduren stehen im globalen Bereich und werden mit `function` eingeleitet. Dann folgt der Name und anschließend in runden Klammern die Parameter. Innerhalb der Prozedur kann dann natürlich jeder beliebige Code vorkommen, es muss nur auf den Gültigkeitsbereich von Variablen geachtet werden. Mit `return` wird die Ausführung abgebrochen und der angegebene Wert zurückgegeben.

```
function machwas ($a, $b, $c) {
    return $a + $b + $c;
}

print machwas(1, 2, 3);
```

Als Parameter können auch Arrays übergeben werden, womit über einen Umweg auch eine beliebige Anzahl von Argumenten möglich wird. Eine spezielle Art der Parameterübergabe ist die Übergabe als Verweis (auch „Referenz“ genannt). Dabei wird nicht der Wert einer Variable an die Funktion weitergereicht, sondern der Speicherort im Arbeitsspeicher. Die Folge davon ist, dass sich eine Änderung der Variable innerhalb der Funktion auch auf die Variable außerhalb auswirkt. Eine generelle Übergabe als Verweis erfolgt durch vorangestelltes &, dies ist auch beim Funktionsaufruf möglich, um eine Übergabe als Verweis zu erreichen, auch wenn die Funktion selbst dies nicht vorsieht. Hier ein Beispiel:

```
function aenderung (&$num) {  
    $num++;  
}
```

```
$a = 1;  
aenderung($a);  
print $a;
```

Um optionale Parameter zu definieren, müssen diese im Funktionskopf mit einem Vorgabewert deklariert werden. Dabei müssen alle optionalen Parameter rechts von den Parametern ohne Vorgabewert stehen. Sollte der Parameter beim Funktionsaufruf nicht mitgeliefert werden, erhält die Variable in der Prozedur den Vorgabewert:

```
function optional ($num, $plus = 20) {  
    print $num + $plus;  
}
```

```
optional(10, 10); // liefert 20  
optional(10); // liefert 30
```

Ähnlich den variablen Variablen unterstützt PHP auch den Einsatz von Variablenfunktionen. Dabei enthält die (String-)Variable den Namen der Funktion. Eine Variablenfunktion wird immer dann als solche erkannt, wenn auf den Variablennamen runde Klammern mit den Parametern folgen (oder leere, wenn es keine oder nur optionale Parameter gibt):

```
function machwas ($a, $b, $c) {  
    return $a + $b + $c;  
}
```

```
$funct = "machwas";  
print $funct(1, 2, 3);
```

2.17. Referenzen

Eine Referenz (oder Verweis) in PHP ist ein Mechanismus, um über verschiedene Namen die selbe Variable anzusprechen. Um eine Referenz zu erstellen, wird eine Variable der anderen über =& zugewiesen, also z.B.:

```
$a =& $b;
```

Änderungen an \$a wirken sich nun auch auf \$b aus und umgekehrt - \$a und \$b sind Aliasse für dieselbe Variable und haben daher immer denselben Wert. Referenzen wurden ja schon bei den Funktionen angesprochen, wo Parameter als Verweis übergeben werden können. Doch kann auch der Rückgabewert einer Funktion ein Verweis sein:

```
function &find_var ($param) {  
    // ...  
    return $found_var;  
}
```

```
$foo =& find_var ($bar);
```

Im Gegensatz zur Parameterübergabe per Referenz ist bei der Rückgabe mittels Referenz an beiden Stellen die Angabe des & notwendig.

2.18. Vordefinierte Variablen

PHP liefert den Skripten eine breite Palette von vordefinierte Variablen, die Auskunft über Art und Zustand des Servers und des Skriptes selbst geben und auch Daten über den Client mitliefern (z.B. welchen Browser er benutzt). Welche Variablen zur Verfügung stehen, hängt aber stark vom verwendeten Betriebssystem und dem Webserver ab, eine komplette Liste erhält man einfach durch die Funktion `phpinfo()`. Erklärungen zu den einzelnen Variablen sind der Hilfe zu entnehmen.

2.19. Datenübertragung vom Client zum Skript

Dieses Kapitel hängt eng mit den vordefinierten Variablen zusammen, da vom Client übertragene Daten in mehreren vordefinierten Arrays mitgeliefert werden. Doch stellt PHP noch eine viel einfachere Methode zur Verfügung, auf diese Daten zuzugreifen.

Grundsätzlich gibt es zwei verschiedene Übertragungsmodi via HTTP: GET und POST. GET bedeutet, dass die Daten über die URL (nach angehängtem ?) übertragen werden. Es muss verwendet werden, wenn es sich um einen Skriptaufruf über Hyperlinks handelt, es kann verwendet werden, wenn man mit einem HTML-Formular arbeitet. POST bedeutet, dass die Daten in einem eigenen Paket an den Server gesendet werden, der Vorteil davon ist, dass die Daten nirgendwo in Klartext aufscheinen (wie bei GET in der Adresszeile). Dennoch stellt die Übertragung mit POST keinen Schutz für geheime Daten dar, da diese genauso auf ihrem Weg zum Server abgefangen werden können.

Bei sensiblen Daten sollte man deshalb Technologien wie SSL (Secured Socket Layer) anwenden.

Verwendet man ein Formular, so kann der Übertragungstyp über das Attribut „method“ festgelegt werden. „Action“ gibt das verarbeitende Skript an.

Für PHP macht es im Grunde keinen Unterschied, ob GET oder POST verwendet wurde. Alle Variablen stehen im globalen Bereich des Skripts unter dem Namen zur Verfügung, den die HTML-Objekte zur Eingabe trugen.

So könnte ein einfaches Formular aussehen:

```
<form action="form2.php" method="post">
  <input type="text" name="formtext">
  <input type="submit" value="Abschicken">
</form>
```

Im Skript kann der eingegebene Text dann folgendermaßen ausgelesen werden:

```
print $formtext;
```

Es können für „name“ im Formular auch Array-Bezeichnungen wie „formvars[text]“ verwendet werden (allerdings nur assoziative, indizierte erhält man durch mehrmaliges verwenden von „formvars[]“, das Array beginnt dann bei 0).

Ein Spezialfall sind Image-Submits, wo ein Bild als „Abschicken“-Button verwendet wird. Dabei stehen zwei zusätzliche Variablen zur Verfügung, welche den angeklickten Punkt auf dem Bild angeben. Im Formular wird das Bild so eingebaut:

```
<input type="image" src="image.gif" name="sub">
```

Im verarbeitenden Skript heißen die Positionsvariablen dann \$sub_x und \$sub_y.

3. Konzepte und Features anhand von Beispielen

3.1. Counter

Ein häufig angewandtes und relativ einfaches PHP-Skript zählt jeden Seitenaufruf und speichert die Anzahl in einer Datei. Da die Datei zuvor eingelesen wird (und nicht extra geprüft werden soll, ob sie vorhanden ist, um das Beispiel auch wirklich einfach zu halten) muss sie zu Beginn mit dem schlichten Inhalt „0“ vorhanden sein. Die Datei soll „counter.txt“ heißen und im gleichen Verzeichnis wie das Skript liegen. Es ist zu beachten, dass PHP Schreibrechte auf der Datei haben muss (in Linux/Unix am einfachsten mit „chmod 666 counter.txt“ oder in Windows NT/2000 mit Berechtigung für den Internetuser).

```
<?
$filename = "counter.txt";
$fp = fopen($filename, "r+");
$count = fread($fp, filesize($filename));

$count++;
print "Schon $count Besucher waren auf meiner Seite!";

rewind($fp);
fwrite($fp, $count);

fclose($fp);
?>
```

Das Beispiel soll Schritt für Schritt näher betrachtet und die verwendeten Dateifunktionen erklärt werden:

Mit `fopen()` wird die angegebene Datei geöffnet und ein Dateizeiger (File-pointer) für weitere Zugriffe zurückgegeben. Schlägt das Öffnen fehl, wird *false* zurückgegeben. Der zweite Parameter gibt den Modus an, in welchem die Datei geöffnet wird, „r+“ steht dabei für „Lesen und Schreiben“, wobei der interne Dateizeiger auf den Anfang der Datei positioniert wird. Beim Schreiben werden vorhandene Daten einfach überschrieben.

`fread()` liest eine bestimmte Anzahl von Bytes vom Dateizeiger und gibt diese als String zurück. Um die gesamte Datei in einen String zu lesen, kann die Funktion `filesize()` mit dem Dateinamen als Parameter verwendet werden, welche die Länge in Bytes zurückgibt.

Obwohl `$count` eine String-Variable ist, wird die Operation `++` darauf angewandt. Diese kann aber auch mit Strings umgehen und die Variable behält auch ihren Typ (anders als bei „`$count = $count + 1;`“, wobei das hier keinen Unterschied machen würde).

Nach dem Einlesen ist der interne Dateizeiger am Ende der Datei, weshalb er vor dem Überschreiben wieder mit `rewind()` an den Anfang gesetzt werden muss. Anschließend wird mit `fwrite()` in die geöffnete Datei geschrieben.

Ein anständiger Counter sollte Seitenbesuche jedoch nur zählen, wenn die Besucher auch wirklich von einer anderen Seite kommen. Wird nur ein Unterverzeichnis besucht und dann auf die Startseite zurückgegangen, ist dies ja deshalb kein neuer Besuch. Um solche „interne Bewegungen“ abzufangen, kann der bei jedem HTTP-Request mitgelieferte „Referer“ ausgelesen werden, der angibt, wo man den Link zur beantragten Seite angeklickt hat (dieser ist leer, wenn man die URL z.B. per Hand eingibt). Der Referer ist in PHP als Umgebungsvariable `$HTTP_REFERER` verfügbar. Da diese Variable innerhalb des Skripts auch überschrieben werden kann, ist es u.U. sicherer, direkt über die `getenv()` Funktion darauf zuzugreifen:

```
$referer = getenv("HTTP_REFERER");
```

Der Referer wird dann mit dem Verzeichnis verglichen, in dem sich unsere Site befindet. Dies sei z.B. „`http://www.mydomain.com/myweb/`“. D.h. wenn man innerhalb dieser Site auf die Counterseite surft, wird nichts gezählt:

```
<?
$filename = "counter.txt";
$fp = fopen($filename, "r+");
$count = (int) fread($fp, filesize($filename));

$site = "http://www.mydomain.com/myweb/";

if ($site != substr($HTTP_REFERER, 0, strlen($site))) {
    $count++;

    rewind($fp);
    fwrite($fp, $count);
}

print "Schon $count Besucher waren auf meiner Seite!";

fclose($fp);
?>
```

Hier werden zwei neue Funktionen verwendet, die zu den Stringfunktionen gehören. `substr()` gibt einen Teil eines Strings zurück, dessen Anfangspunkt und (optional) Länge angegeben wird. `strlen()` liefert die Länge eines Strings. Dass bei diesem Aufruf die Länge von `$site` auch größer als die Stringlänge des Referers sein kann, stört die `substr()` Funktion nicht.

3.2. Gästebuch

Das hier vorgestellte, sehr einfache Gästebuch soll noch ohne Datenbanken auskommen. Der Besucher kann auf der Seite „gbentry.htm“ seinen Namen, EMail und eine kurze Bemerkung eintragen, nach Abschicken wird das Skript „gbsave.php“ aufgerufen, das diese Daten in einer Datei „gbdata.inc“ ablegt (neue Einträge werden einfach vorn hinzugefügt). Schließlich gibt es noch die Gästebuch-Hauptseite „gb.php“, welche „gbdata.txt“ einbindet und einen Link auf „gbentry.htm“ bietet.

Die Textfelder beim Formular heißen „Name“, „EMail“ und „Bemerkung“, die Action wird auf „gbsave.php“ gesetzt.

Hier das Skript „gbsave.php“:

```
<?
// vom Formular werden $Name, $EMail und $Bemerkung übergeben
$datafile = "gbdata.txt";

// prüfen ob ein Name und eine Bemerkung eingegeben wurde
if (($Name == "") or ($Bemerkung == "")) {
    ?>Bitte Name und Bemerkung nicht vergessen.<?
}
else {
    $data = ""; // Variable initialisieren

    // Datenfile zum Lesen öffnen, Fehler wenn noch nicht existiert
    $fp = @fopen($datafile, "r");
    if ($fp != false) { // existiert?
        $data = fread($fp, filesize($datafile));
        fclose($fp);
    }

    // Datenfile zum Schreiben öffnen
    // erstellen wenn noch nicht vorhanden
    $fp = fopen($datafile, "w");

    $entry = "<p><b>Von: </b> ";
    if ($EMail != "") {
        $entry .= "<a href=\"mailto: " . htmlentities($EMail) . "\"> " .
        htmlentities($Name) . "</a>";
    } else {
        $entry .= $Name;
    }
    $entry .= "\n<br><b>Bemerkung: </b><br>" . nl2br(htmlentities(
    $Bemerkung)) . "</p><hr>\n\n";

    // neuen Eintrag und schon vorhandene schreiben
    fwrite($fp, $entry);
```

```

fwrite($fp, $data);

fclose($fp);

?>Eintrag gespeichert.<br><a href="gb.php">Zum Gästebuch...</a><?
}
?>

```

Zu beachten sind die Funktionen `nl2br()` und `htmlentities()` bei den String-Variablen (außer `$EMail`), die in die Datei geschrieben werden. Ersteres sorgt für eine Umwandlung aller Zeilenumbrüche in „`
`“, da nur diese in HTML auch als Zeilenumbruch dargestellt werden. Die zweite Funktion ersetzt alle Sonderzeichen (wie „`<`“, „`&`“, aber auch „`ä`“) mit den entsprechenden HTML-Codes. Um nur die HTML-Steuerzeichen („`<`“, „`>`“, „`&`“, „`\"`“, „`'`“) zu ersetzen, kann `htmlspecialchars()` verwendet werden. Wichtig ist beim Aufruf die Reihenfolge, da die eingefügten „`
`“ natürlich nicht wieder in HTML-Codes umgewandelt werden sollten. `$EMail` wurde nicht umgewandelt, da es ohnehin keine Sonderzeichen enthalten darf.

„gb.php“ selbst enthält nur folgende Zeile:

```
<? @include("gbdata.txt"); ?>
```

Sollte die Datei nicht existieren, wird wegen des `@`-Operators eine Fehlermeldung unterdrückt. Das gleiche wurde beim Öffnen zum Einlesen gemacht, um den Fehler selbst behandeln zu können, ohne dass deshalb eine Meldung an den Besucher erfolgt.

3.3. Lotto

Zufällig 6 Zahlen zwischen 1 und 45 auszuwählen und anzuzeigen klingt trivialer als es eigentlich ist. Da die Kugeln bei Lotto ja gezogen und nicht wieder zurückgelegt werden, kann eine Zahl nur einmal gezogen werden. Dies wird gelöst, indem bei jeder Ermittlung mit allen bereits gezogenen Zahlen verglichen wird. Kommt eine Zahl doppelt vor, wird noch einmal gezogen.

Natürlich kann PHP keine richtigen Zufallszahlen liefern. Die Funktion `rand()` gibt eine Ganzzahl aus einem angegebenen Bereich wieder. Dabei wird allerdings bei jeder Skriptausführung die gleiche Sequenz generiert. Man sollte daher zuvor mit `srand()` einen internen Startwert (seed) für den Zufalls-generator festlegen. Es ist üblich, dafür die aktuelle Systemzeit zu verwenden, die sich mit Sicherheit ständig ändert. Das führt dann zu immer anderen Sequenzen. Als Zeitgeber wird die Funktion `microtime()` herangezogen, welche die vergangenen Sekunden seit Beginn der Unix-Epoche (1.1.1970, 0:00 GMT) mit vorangestelltem Nachkommateil liefert.

```

<?
// Lotto - 6 aus 45

```

```

// seed für Zufallszahlen setzen
srand ((double) microtime()*1000000);

$i = 0;
$z = array(); // damit foreach funktioniert
do {
    $r = rand(1, 45);
    // prüfen ob schon gezogen
    foreach ($z as $einz) {
        if ($einz == $r) {
            // neu ermitteln
            continue 2;
        }
    }
    // ins Array aufnehmen
    $z[] = $r;
    $i++;
} while ($i < 6);

// sortieren
sort($z);
print implode(", ", $z);
?>

```

Im Beispiel wurde die Array-Funktion `array()` (wobei diese keine ‚richtige‘ Funktion ist, sondern ein Sprachkonstrukt) benutzt, um ein leeres Array zu generieren. `array()` wird normalerweise dazu benutzt auf schnellem Weg ein Array mit Daten zu füllen, z.B.:

```
$arr = array(1, 2, 3, 4, 5);
```

Wichtig ist die Zeile, in der mit `continue` aus den beiden Schleifen `foreach` und `do` herausgesprungen und der nächste Durchlauf von `do` gestartet wird. Da es sich um zwei Schleifen handelt, darf man die Angabe von „2“ nicht vergessen.

Ganz am Ende sind die sechs Zufallszahlen in einem Array zusammengefasst, das mit der Array-Funktion `sort()` sortiert wird. Um sich die Schleife zum Ausgeben der Daten zu ersparen, wird das Array mit der String-Funktion `implode()` und einem mitgelieferten Trennzeichen zu einem String zusammengesetzt. Das Gegenstück zu `implode()` ist `explode()`, das aus einem String und einem bestimmten Trennzeichen ein Array macht.

Einfacher ließe sich mit der Funktion `in_array()` überprüfen, ob eine Zahl schon gezogen wurde. Damit könnte man sich die `foreach`-Schleife ersparen.

3.4. Mail-Versenden

Es kann in vielen Situationen interessant sein, vom Skript ein Mail zu generieren, das z.B. den Webmaster über neue Gästebucheinträge informiert, einen Forum-Autor über eine neue Reaktion auf seinen Beitrag unterrichtet etc. PHP bietet mit der Funktion `mail()` eine sehr einfache Realisierungsmöglichkeit, die allerdings voraussetzt, dass der Webserver Mail verschicken kann. Dazu muss in Windows ein entsprechender SMTP-Server für PHP angegeben werden (in der „`php.ini`“), in Linux/Unix wird das Mail einfach an `sendmail` weitergegeben. Der Syntax sieht folgendermaßen aus:

```
mail($to, $subject, $message [, $additional_headers])
```

Die `$additional_headers` können z.B. verwendet werden, um einen anderen Absender als den in der „`php.ini`“ definierten Standardabsender zu verwenden. Mehrere zusätzliche Header-Einträge werden durch „`\n`“ getrennt:

```
$additional_headers = "From: foo@bar.com\nReply-To: bar@foo.com"
```

Im obigen Beispiel wird auch gezeigt, wie man eine alternative Reply-To-Adresse setzen kann. Achtung: Bei Windows-Systemen gilt „`\n`“ nicht als vollwertiger Zeilenumbruch und es kommt deshalb zu einem Fehlverhalten. Werden die Mails also über einen Windows-Server geschickt, sollte man stattdessen „`\r\n`“ einfügen (was auch bei Unix/Linux-Servern nichts schadet).

Um Mailkonten abzufragen stellt PHP eine umfangreiche IMAP-Bibliothek zur Verfügung, die neben IMAP auch POP3 und NNTP unterstützt.

3.5. Download-Liste

Hierbei ist die Anforderung, den Inhalt eines Verzeichnisses auszulesen und die Dateien als Links für den Download bereitzustellen. Sollte man also neue Dateien im Download-Verzeichnis ablegen, muss man die entsprechende Downloadseite nicht updaten.

Es wird die `dir()` Funktion verwendet, die vom PHP-Manual als „pseudo-objektorientierter Mechanismus zum Lesen eines Verzeichnisses“ bezeichnet wird. Diese gibt dann eine Objektreferenz zurück. Im folgenden Beispiel wird zum ersten mal gezeigt, wie man auf die Methoden eines Objekts zugreift: dies geschieht durch den Namen der Objektvariable und der Methode selbst, die durch „`->`“ verbunden werden.

```
<?
$d = dir("download");
while ($entry = $d->read()) {
    echo "<a href=\"download/$entry\">$entry</a><br>\n";
}
$d->close();
?>
```

Dabei ist insbesondere zu beachten, dass die Methoden von Klammern () abgeschlossen werden (in denen die Parameter stehen, wenn welche verwendet werden). Das ist deshalb notwendig, damit PHP die Methoden des Objekts von seinen Eigenschaften (ohne Klammern) unterscheiden kann. Eigenschaften wären „handle“ und „path“. Ersteres ist ein Verweis auf einen Eintrag in einer internen Tabelle zur Verwaltung von Objekten und kann in Zusammenhang mit anderen Verzeichnisfunktionen (mit denen sich das Beispiel genauso umsetzen ließe) verwendet werden. Der „path“ ist einfach die Pfadangabe des geöffneten Verzeichnisses:

```
print $d->path;
```

Der Vorteil dieser Objektstruktur ist, dass alle Funktionen (hier Methoden genannt) unter einem Namen (im Beispiel \$d) zusammengefasst sind und sich dann jeweils auf das „Verzeichnis“ \$d beziehen. Andernfalls müsste man für jedes geöffnete Verzeichnis einen Handle verwalten, der den Funktionen dann mitgeliefert wird.

Als zusätzliche Methode ist noch rewind() verfügbar, wodurch man wieder zum Anfang der Liste (also zu „.“) zurückkehrt.

Im Beispiel ist (\$entry = \$d->read()) die Bedingung für den Durchlauf der while-Schleife. Diese Zuweisung gibt *false* zurück, wenn sie nicht erfolgreich ist, d.h. wenn das Ende der Dateiliste für das Verzeichnis erreicht ist. Wenn „.“ und „..“ nicht in der Download-Liste enthalten sein sollen, können sich einfach mit einer if-Abfrage übersprungen werden.

3.6. Datei-Upload

PHP ermöglicht Dateiuploads über HTTP auf eine sehr einfache Weise: Um die Datei auszuwählen muss in einem Formular ein File-Input-Feld eingebaut werden. Das verarbeitende Skript bekommt dann vier Variablen übergeben, welche die Position der Datei auf dem Server nach dem Hochladen, den ursprünglichen Dateinamen, die Größe in Bytes und den MIME-Typ angeben. Da der Prozess des Uploads bei größeren Dateien natürlich einige Zeit in Anspruch nehmen kann, sollte man nicht vergessen, den Besucher entsprechend zu informieren, damit dieser die Übertragung nicht abbricht. Das Formular könnte folgendermaßen aussehen:

```
<form method="POST" action="upload.php" enctype="multipart/form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
  <input type="file" name="Datei" size="30"><br>
  <input type="submit" value="Abschicken">
</form>
```

Das Hidden-Input-Feld setzt dabei die maximale akzeptierte Größe von Dateien in Bytes. Die in der „php.ini“ angegebene Maximalgröße beschränkt dies jedoch immer noch zusätzlich. Standardmäßig werden die Dateien am Server im entsprechenden PHP-Uploadverzeichnis abgelegt (ebenfalls in

„php.ini“ festgelegt) und wieder entfernt, wenn das verarbeitende Skript abgeschlossen ist. D.h. es obliegt immer dem Skript-Programmierer, die Datei an ihren vorbestimmten Platz zu verschieben. Der Zugriff erfolgt in diesem Beispiel über die Variable `$Datei` (immer der Name des File-Input-Feldes; enthält den vollen Pfad). `$Datei_name` gibt den ursprünglichen Namen auf dem System des Absenders an, `$Datei_size` die Größe und `$Datei_type` den MIME-Typ.

Ein verarbeitendes Skript könnte so aussehen:

```
<?
if (!move_uploaded_file($Datei, "uploadir/$Datei_name")) {
    print "Datei ungültig oder zu groß.";
}
?>
```

Die Funktion `move_uploaded_file()` prüft, ob `$Datei` eine gültige, mit HTTP-POST upgeladete Datei ist, ansonsten wird *false* zurückgegeben. Da diese Funktion nur in neueren PHP-Versionen verfügbar ist, kann alternativ auch auf `copy()` zurückgegriffen werden, da die temporären Dateien ohnehin nach Beendigung des Skripts gelöscht werden.

3.7. Datum und Uhrzeit

PHP liefert natürlich auch Funktionen zum ermitteln des aktuellen Datums und der Uhrzeit. Entweder man verwendet `time()`, was den aktuellen Unix-Timestamp liefert (die seit Beginn der Unix-Epoche vergangenen Sekunden, vgl. „3.3 Lotto“) oder `date()` das entweder die aktuelle Zeit oder einen mitgelieferten Timestamp beliebig formatiert. Hier einige Beispiele:

```
<?
print time() . "<br>"; // UNIX-Timestamp
print date("d.m.y H:i:s") . "<br>"; // Datum/Zeit kurz
print date("D., j. F Y") . "<br>"; // Datum lang
print date("d.m.Y H:i:s", 0) . "<br>"; // Beginn der Unix-Epoche
?>
```

Des Weiteren kann man über `getdate()`, welches ein assoziatives Array liefert, einzelne Zeitangaben zu einer bestimmten Timestamp ermitteln (z.B. Tag des Monats). Mit `mktime()` und `strtotime()` ist es möglich, einen Unix-Timestamp zu errechnen, was z.B. für das Addieren einer bestimmten Anzahl von Tagen zu einem Datum gebraucht wird. Schließlich prüft `checkdate()`, ob ein angegebene Datum nach dem gregorianischen Kalender gültig ist.

3.8. Cookies

Cookies bieten die Möglichkeit, kleine Datenmengen am Client abzuspeichern und bei späteren Besuchen auf der Seite wieder abzufragen. Die Cookies

werden immer mit dem Servernamen abgespeichert, weshalb es unmöglich ist, dass ein Server die Cookies eines anderen abfragt. Vom Server werden Cookies über den HTTP-Header geschickt (weshalb der entsprechende PHP-Befehl vor jeder normalen Ausgabe gegeben werden muss). Bei jeder Anfrage des Clients an den Server werden zu diesem Server gehörende Cookies dann wieder zurückgesendet.

Ein Cookie besteht im Wesentlichen aus drei Werten: dem Cookie-Namen, dem Cookie-Wert und einer Ablaufzeit (als Unix-Timestamp). Der Befehl zum Setzen lautet `setcookie()`:

```
setcookie("CookieName", "CookieWert", time()+3600*24*30);
```

Im diesem Beispiel wird ein Cookie mit einer Ablaufzeit von 30 Tagen gesetzt. Um die Cookies, die einem Skript offen stehen, abzufragen, greift man auf das assoziative Array `$HTTP_COOKIE_VARS` oder einfach auf den Cookie-Namen als Variable zu:

```
print $HTTP_COOKIE_VARS["CookieName"];  
print $CookieName;
```

Es ist auch möglich ein Array an Cookies zu setzen und dann unter einem Namen wieder auszulesen:

```
setcookie("CookieArray[0]", "Array 0", time()+3600);  
setcookie("CookieArray[1]", "Array 1", time()+3600);  
setcookie("CookieArray[2]", "Array 2", time()+3600);
```

```
print $CookieArray[0];  
print $CookieArray[1];  
print $CookieArray[2];
```

Um ein Cookie zu löschen, muss es nur auf einen Leerstring als Wert und eine Verfallszeit in der Vergangenheit gesetzt werden:

```
setcookie("CookieName", "", time()-3600);
```

Um zu überprüfen, ob ein Cookie überhaupt gesetzt ist (und somit als Variable zur Verfügung steht), muss man nur die Existenz der Variable mit der Funktion `isset()` überprüfen.

3.9. HTTP-Authentifizierung

HTTP stellt einen sehr praktischen Mechanismus zur Verfügung, den Zugriff auf Webseiten zu beschränken. Dabei wird bei einer Seitenanforderung statt der eigentlichen Seite im HTTP-Header „HTTP/1.0 401 Unauthorized“ zurückgegeben, was den Browser dazu veranlasst, nach Username und Passwort zu fragen. PHP unterstützt diesen Mechanismus, indem die User-Daten als `$PHP_AUTH_USER` und `$PHP_AUTH_PW` bereitgestellt werden.

Das Skript kann die Daten dann prüfen und wenn sie stimmen die normale Seite ausgeben. Stimmen sie nicht, wird wieder über die Funktion header() ein „401“ an den Client zurückgeschickt und der User wird noch einmal um Eingabe seiner Zugangsdaten gebeten (nach dem dritten Mal erscheint die Seite, die als ‚Ersatz‘ geboten wird). Voraussetzung ist wie bei den Cookies, dass die Funktion header() vor jeder normalen HTML-Ausgabe aufgerufen wird, da der Header ansonsten schon abgeschickt wurde.

Die Authentifizierung ist immer an eine Zone („realm“) gebunden. Ist man erst einmal angemeldet, übernimmt der Browser die Registrierung beim nächsten Seitenaufruf selbst und zwar so lange, bis das Browserprogramm geschlossen und wieder gestartet wird. Damit wird es möglich, ohne Variablenübergabe zwischen den Skripten mehrere Seiten mit einem Passwort zu sichern.

Hier ein Beispiel für so ein einfaches Authentifizierungsskript:

```
<?
function SendAuth() {
    header("WWW-Authenticate: Basic realm=\"Verbotene Zone\"");
    header("HTTP/1.0 401 Unauthorized");

    // Ersatzseite
    ?>
    <html><body>Zugangsdaten waren falsch.</body></html>
    <?
}

if (!isset($PHP_AUTH_USER)) { // Variable initialisiert?
    SendAuth();
    exit(); // Skript beenden
} else {
    if (!(($PHP_AUTH_USER == "User") and ($PHP_AUTH_PW ==
"Password"))) {
        // Authentifizierung fehlgeschlagen!
        SendAuth();
        exit(); // Skript beenden
    }
}

// geschützte Seite:
?>
<html><body>Willkommen im geschützten Bereich.</body></html>
```

Leider funktioniert die HTTP-Authentifizierung nur, wenn PHP als Apache-Modul läuft. Die Eingabe ist übrigens case-sensitive und es ist zu beachten, dass die Daten standardmäßig nicht verschlüsselt werden (dazu sollte man auf SSL zurückgreifen).

3.10. News-Grabber

Da man stets aktuelle Inhalte auf seiner Seite präsentieren aber sich nicht selbst darum kümmern will, kann man auf einen sogenannten „News-Grabber“ zurückgreifen. Das Prinzip dahinter ist sehr einfach: Beim Aufruf der Seite holt sich PHP den Inhalt einer anderen Seite (wo man sich um die News kümmert) und zeigt sie auf der eigenen Seite an. Dabei kann mit etwas String-Parsen sowohl der ausgewählte Bereich als auch das Layout angepasst werden. Das Beispiel greift auf die News von heise.de zu und wird auf der Startseite von inter.at eingesetzt:

```
<?
// Heise-Newsticker einlesen
$url = "http://www.heise.de/newsticker/";
$fp = fopen($url, "r");
$news = fread($fp, 100000);
fclose ($fp);

$news = strstr($news, "</HEISETEXT>"); // start von news
$end = strpos($news, "Fehlt eine wichtige Nachricht?"); // ende
$news = substr($news, 16, $end - 56);

// Links reparieren
$news = str_replace("\", "\http://www.heise.de/", $news);

print $news;
?>
```

Wie man sieht erfolgt der Zugriff auf Remote-Dateien fast gleich wie bei lokalen, bis auf die Tatsache, dass der Dateiname mit „http://“ eingeleitet wird. Alle relativen Links auf der Originalseite müssen abgeändert werden, um auch von der neuen Position aus zu funktionieren. Da das Skript sehr viele seitenspezifische Angaben enthält, muss der Grabber mit jeder Änderung der „Quelle“ natürlich neu angepasst werden.

4. Datenbankanbindung (an MySQL)

4.1. PHP, Datenbanken und SQL

Datenbankanwendungen sind überall dort vonnöten, wo größere, strukturierte Datenmengen bereitgestellt werden oder gespeichert werden müssen. Sie bieten sich bei Produktkatalogen, Personenlisten, Foren oder News-Seiten an. PHP ermöglicht eine sehr einfache Anbindung an Datenbanken und unterstützt die meisten frei verfügbaren und viele kommerzielle Datenbanksysteme. Alle wichtigen Funktionen im Zusammenhang mit Datenbanken sind fix integriert oder müssen (als Extensions) nur noch aktiviert werden.

Um auf die Datenbanken zugreifen oder sie manipulieren zu können, wird von den meisten Datenbanksystemen eine eigene standardisierte Sprache, genannt SQL (Structured Query Language), verwendet. Zwischen verschiedenen Datenbanksystemen kann es dabei zu „Dialektunterschieden“ kommen, der generelle Sprachumfang ist jedoch immer derselbe. Eine Einleitung erhält man z.B. unter:

<http://w3.one.net/~jhoffman/sqltut.htm>

Für einfache Anwendungen sind vor allem vier Prozesse wichtig:

- das Auswählen von bestimmten Datensätzen
- das Aktualisieren von Datensätzen
- das Erstellen von neuen Datensätzen
- das Löschen von Datensätzen

Für jede dieser Aufgaben existieren spezielle SQL-Befehle. Um einen Datensatz identifizieren zu können, wird in relationalen Datenbanken ein Index-Feld (ID) verwendet, das jedem Datensatz eine eindeutige Zahl zuweist. In vielen Datenbanksystemen kann die Vergabe dieser Zahl automatisch vorgenommen werden (man muss sich also nicht selbst darum sorgen, dass die Zahl auch wirklich eindeutig ist). Relationen werden dann über diesen Index hergestellt. Werden z.B. zu einer Produktkategorie (aus der Tabelle *ProduktKat*) mehrere Produkte (in der Tabelle *Produkte*) zugewiesen, enthalten die Felder *Kat_ID* der Produkte jeweils den Wert des Feldes *ID* der zugehörigen Kategorie (in einer n:1 Beziehung).

Hier einige Beispiele von SQL-Queries:

Auswahl:

```
SELECT * FROM Produkte WHERE Kat_ID = 4;
```

Aktualisieren:

```
UPDATE Produkte SET Name = 'Korbstuhl' WHERE ID = 67;
```

Erstellen eines neuen Datensatzes:

```
INSERT INTO Produkte (Name, Kat_ID) VALUES ('Klappstuhl', 4);  
(für ID wurde kein Wert gesetzt, er wird automatisch vergeben)
```

Löschen:

```
DELETE FROM Produkte WHERE Kat_ID = 3;  
(hier werden alle Produkte einer Kategorie gelöscht)
```

4.2. MySQL

MySQL ist ein schneller und stabiler SQL-Datenbankserver und im Rahmen der GPL völlig kostenlos verwendbar (unter www.mysql.org gibt es aktuelle Downloads). Für Windows stehen zudem ODBC-Treiber (MyODBC) zur Verfügung, die einen Zugriff von Windows-Programmen auf den MySQL-Server ermöglichen. PHP hat die Unterstützung dieses Servers fix eingebaut, alle MySQL-Funktionen tragen den Präfix „mysql“.

MySQL verwendet ein komplexes Privilegien-System, das gemeinsam mit den Benutzernamen und Passwörtern in der Datenbank „mysql“, die von Beginn an existiert, gespeichert wird. Änderungen an den Privilegien werden über diese Datenbank getätigt, wobei zu beachten ist, dass man MySQL danach neu starten muss, ehe die Änderungen wirksam werden. Privilegien können auf verschiedenen Ebenen definiert werden, die durch Tabellen in der „mysql“ Datenbank repräsentiert werden.

Die Tabelle „user“ legt die Benutzer und ihre Passwörter fest und bestimmt, von welchen Hosts sich die Benutzer anmelden können (spezielle Einträge hierfür sind „localhost“ für den Server selbst und „%“ für alle Hosts). Die einzelnen hier zu setzenden Privilegien (entweder „Y“ oder „N“) definieren die Superuser-Privilegien des entsprechenden Users – für den root-User ist alles auf „Y“, für andere Benutzer schaltet man für höchste Sicherheit alles auf „N“.

Die Tabelle „db“ legt fest, welche Benutzer von welchen Hosts aus auf welche Datenbanken zugreifen können. Zudem können die einzelnen Operationen auf den Datenbanken eingeschränkt werden. Wichtig ist hierbei v.a. das Grant-Privileg, das es einem User erlaubt, selbst seine Privilegien auch an andere weiterzugeben.

Auf der Kommandozeile des (Unix/Linux-)Servers kann mit folgendem Befehl der MySQL-Client gestartet werden:

```
mysql -u <username> -p<password>
```

(ohne die <, >; Achtung: nach -u folgt ein Leerzeichen, nach -p nicht. Lässt man das Passwort weg (nicht aber -p), kann man es „geheim“ eingeben.)

Nach Auswahl einer Datenbank kann man hier direkt mit SQL-Befehlen auf diese zugreifen.

4.3. phpMyAdmin

Der MySQL-Client auf der Kommandozeile erfordert eine recht umständliche Eingabe über SQL-Befehle und ist auch bei der Ausgabe von Tabellen nicht die beste Wahl. Zur Administration des MySQL-Servers ist ein sehr gutes Web-Interface verfügbar, das komplett in PHP geschrieben wurde. Es heißt phpMyAdmin und ist unter folgender Adresse verfügbar:

```
http://phpmyadmin.sourceforge.net
```

Nach der Installation (also dem Kopieren in ein Verzeichnis) muss phpMyAdmin über die Datei „config.inc.php“ eingerichtet werden. Hier ist die Angabe

eines oder mehrerer MySQL-Server möglich. Es genügt meist, einen Host, User und Password anzugeben. Wird nicht als „root“ auf den MySQL-Server zugegriffen, ist der Zugriff auf eine oder auch mehrere Datenbanken beschränkt (je nach gesetzten Privilegien), angezeigt werden allerdings immer alle Datenbanken.

Über die Datenbank „mysql“ ist es auch so möglich, Änderungen am Privilegiensystem vorzunehmen. Wichtig ist nur die Funktion „PASSWORD“, wenn man neue User einträgt oder deren Passwörter ändert. Damit werden die Passwörter nicht Klartext, sondern als Hash gespeichert.

Die Verwaltung von Datenbanken reduziert sich mit dem phpMyAdmin auf Mausklicks, es sind jedoch auch Aufgaben wie Importieren und Exportieren oder das direkte Ausführen von SQL-Befehlen möglich.

Natürlich sollte man darauf achten, wer Zugriff auf den phpMyAdmin hat, denn dieser ist standardmäßig nicht durch ein Passwort geschützt (die „advanced authorisation“ macht dies z.B. möglich, man kann aber auch selbst leicht das Verzeichnis schützen). Das in der „config.inc.php“ gespeicherte Passwort ist jedoch sicher, da die Datei sich durch die Endung „.php“ nicht direkt abrufen lässt.

4.4. Datenbankbindung in PHP

Wie schon erwähnt beginnen die MySQL-Funktionen von PHP alle mit „mysql_“. Um die Verbindung zum Server herzustellen, wird `mysql_connect()` mit Servername, Username und Passwort aufgerufen. Die Funktion liefert eine Verbindungsnummer zurück, die bei allen folgenden Operationen mit angegeben werden kann. Wird sie nicht angegeben, greift PHP auf die gerade aktive (bzw. letzte verwendete) Verbindung zu. Bei einem Misserfolg liefert die Funktion *false* zurück.

Ist die Verbindung zum Server hergestellt, wird mit `mysql_select_db()` eine Datenbank ausgewählt. Der angemeldete User muss natürlich die Berechtigung dafür haben. Sollte die Auswahl misslingen, wird auch hier *false* zurückgegeben, ansonsten *true*.

`mysql_query()` sendet einen SQL-Befehl zum Datenbankserver und gibt bei Erfolg die entsprechende Ergebnis-Kennung zurück, bei Misserfolg *false*. Mit `mysql_result()` kann das Ergebnis zeilen- und feldweise ausgelesen werden, `mysql_affected_rows()` liefert die Anzahl an zurückgegebenen Zeilen (können auch 0 sein) und `mysql_free_result()` gibt die belegten Ressourcen wieder frei.

Schließlich dient `mysql_close()` dazu, die Verbindung zur Datenbank wieder zu schließen.

Es steht noch eine Reihe von anderen, teilweise leistungsfähigeren Funktionen zur Verfügung, die Ergebnisse eines Datenbankaufrufes zurückzuliefern.

mysql_fetch_array() liefert z.B. eine Zeile aus dem angegebenen Ergebnis als assoziatives und zugleich indiziertes Array oder *false*, wenn kein Datensatz gelesen werden konnte. Jeder weiterer Aufruf der Funktion gibt eine neue Zeile zurück.

4.5. Datenbankzugriff in einem Beispiel

In einem einfachen Beispiel sollen die vorgestellten Funktionen vorgestellt werden. Es soll ein einfacher Produktkatalog angezeigt werden, die Felder der Tabelle *Produkte* sind: *ID*, *Name* und *Preis*.

```
<table>
<tr><td><b>Name</b></td><td><b>Preis</b></td></tr>
<?
$link = mysql_connect("my.sqlserver.com", "user", "passwd");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM Produkte", $link);
if ($result != false) {
    $n = mysql_affected_rows($link);
    for ($i=0; $i<$n; $i++) {
        print "<tr><td>";
        print mysql_result($result, $i, "Name");
        print "</td><td>";
        print mysql_result($result, $i, "Preis");
        print "</td></tr>\n";
    }
    mysql_free_result($result);
}
mysql_close($link);
?>
</table>
```

4.6. Bilder in Datenbanken ablegen

Mit dem BLOB-Datentyp bietet MySQL auch eine Möglichkeit, binäre Daten (wie Bilder es sind) in der Datenbank abzulegen. Das ist immer dann sinnvoll, wenn umfangreiche Bilddaten anderen Daten in der Datenbank zugeordnet werden (wie z.B. Beispielbilder in einer Produktdatenbank). Dabei wird im Grunde ganz einfach vorgegangen: Die Daten werden wie gewohnt mit über SQL in die Datenbank geschrieben und genauso wieder ausgelesen, als ob es sich um normale Felder handeln würde. Um die Bilder über ein PHP-Skript auch anzeigen zu können, wird dieses statt eines Bildes eingebunden. Voraussetzung ist, dass das PHP-Skript dann mit der header() Funktion einen entsprechenden Bild-Header und anschließend gültige Bilddaten liefert.

Die Einbindung eines PHP-Skripts als Bild könnte so aussehen:

```

```

Das zugehörige Skript greift in der Datenbank auf die Tabelle „bilder“ zu und gibt das (GIF-)Bild mit der angegebenen ID zurück:

```
<?
header("Content-type: image/gif");

$link = mysql_connect("my.sqlserver.com", "user", "passwd");
mysql_select_db("database", $link);

$result = mysql_query("SELECT Bild FROM bilder WHERE ID=$ID", $link);
if ($result != false) {
    if (mysql_affected_rows($link) > 0) {
        print mysql_result($result, 0, "Bild");
        mysql_free_result($result);
    }
}

mysql_close($link);
?>
```

Man könnte zum Speichern der Bilder eine Upload-Form verwenden, in welcher ein Bild ausgewählt und an ein Upload-Skript weitergegeben wird. Hierbei ist nur zu beachten, dass man die Funktion addslashes() nicht vergessen sollte, da innerhalb der scheinbar wirren Zeichenfolge, aus denen die Binärdaten bestehen, auch ein von MySQL reserviertes Zeichen enthalten sein kann.

```
<?
if (!is_uploaded_file($Bild)) {
    print "Datei ungültig oder zu groß.";
} else {
    // Bild einlesen
    $Bild = fread(fopen($Bild, "r"), $Bild_size);

    // als Kontrolle ausgeben
    header("Content-type: image/gif");
    print $Bild;

    // DB oeffnen
    $link = mysql_connect("my.sqlserver.com", "user", "passwd");
    mysql_select_db("database", $link);

    // Sonderzeichen escapen
    $Bild = addslashes($Bild);

    // in DB speichern
    $return = mysql_query("INSERT INTO bilder (Bild) VALUES('$Bild')",
    $link);
```

```
mysql_close($link);  
}  
?>
```

Diese Beispiele sind auf GIF-Bilder beschränkt, da der Header entsprechend gesetzt wird. Man kann den MIME-Typ aber auch ganz leicht aus der Dateierweiterung ermitteln bzw. in der Datenbank ablegen.

5. Programmierung eines Forums in PHP

5.1. Grundlagen

Das Forum kann als eine Art erweitertes Gästebuch angesehen werden. Einträge können sich dort auch auf bereits vorhandene beziehen und werden diesen dann in einer Baumstruktur (vergleichbar mit dem Dateisystem) untergeordnet. Jeder Eintrag besteht grundsätzlich aus fünf Feldern: *Autor*, *EMail*, *Zeit*, *Subject* und *Text*. Zur Verwaltung und Strukturierung der Einträge werden zwei weitere Felder benötigt: *ID* und *parent_ID*. Das Feld *parent_ID* bildet die Verknüpfung zu anderen Einträgen. Es enthält 0, wenn der Eintrag in der untersten Ebene gepostet wurde (als neues Thema / neuer Thread) bzw. die *ID* des Eintrags, auf den geantwortet wurde und dem der neue Eintrag untergeordnet wird.

Aus Gründen der Darstellbarkeit sollte die maximale Anzahl von Ebenen beschränkt werden (z.B. auf 10).

Folgende Seite werden für ein funktionstüchtiges Forum benötigt: Eine Seite, in welcher die Einträge samt Baumstruktur angezeigt werden („default.php“), eine Seite für neue Einträge („entry.php“) und eine Seite um neue Einträge zu speichern („entry_save.php“).

5.2. Darstellung der Baumstruktur

Da jeder Eintrag im Prinzip beliebig viele Untereinträge haben kann, muss man eine flexible Programmierung anwenden um alle Einträge und ihre Beziehung zueinander darzustellen. Dazu soll eine Funktion `show_subentries()` geschrieben werden, die als Argument die *ID* des Eintrags annehmen soll, dessen Untereinträge als Liste angezeigt werden. Für jeden ausgegebenen Eintrag wird die Funktion dann wieder aufgerufen, um auch dessen Untereinträge zu zeigen. Dieses Programmierkonzept wird als „rekursive Funktionsaufrufe“ bezeichnet und kommt v.a. bei Daten in Baumstruktur zur Anwendung.

Gestartet wird der gesamte Prozess durch einmaliges Aufrufen der Funktion mit dem Argument 0, also den Einträgen der untersten Ebene.

In `show_subentries()` wird zuerst überprüft, ob es Einträge mit der angegebenen *parent_ID* gibt. Sind solche vorhanden, wird eine einfache Tabelle mit zwei Spalten ausgegeben, deren erste Spalte nur als Einrückung dient, um die Baumstruktur darzustellen.

```
<?
$link = mysql_connect("my.sqlserver.com", "user", "passwd");
mysql_select_db("database", $link);

function show_subentries($parent_ID) {
    global $link;
```

```

    $sql = "SELECT * FROM forum WHERE parent_ID = $parent_ID ORDER BY
Zeit DESC";
    $result = mysql_query($sql, $link);
    if ($result != false) {
        $n = mysql_affected_rows($link);
        for ($i=0; $i<$n; $i++) {
            ?><table><tr><td width="20">&nbsp;&nbsp;&nbsp;</td><td><?

            print mysql_result($result, $i, "Subject")."<br>";

            // Funktion rekursiv aufrufen um Untereinträge zu zeigen
            show_subentries(mysql_result($result, $i, "ID"));

            ?></td></tr></table><?
        }
    }
}

// für Einträge unterster Ebene aufrufen
show_subentries(0);

mysql_close($link);
?>

```

Das obige Beispiel zeigt von den Einträgen zwar nur das *Subject*, es kann aber leicht um *Autor* und *E-Mail* ergänzt werden. Dann sollte ein Link von jedem Eintrag wieder auf „default.php“ zeigen, dem die *ID* des Eintrages übergeben wird. Der gewählte Eintrag wird dann angezeigt und in der Baumstruktur eventuell hervorgehoben.

5.3. Einträge speichern

Die Seite mit dem Eintragsformular übergibt die Daten an das Skript „entry_save.php“, wo diese in die Datenbank eingetragen werden. Um dem Benutzer eine Seite wie „Ihr Eintrag wurde gespeichert.“ zu ersparen, soll danach direkt zum Forum weitergeleitet werden. Dies geschieht aber nicht mit HTML-Methoden (wie JavaScript oder einem Meta-Tag; diese Methoden funktionieren zudem nicht überall) sondern indem der HTTP-Header mit PHP manipuliert und direkt über den Header auf eine andere Seite umgeleitet wird.

Wichtig beim Speichern des Eintrags ist jedoch, dass der Wert für das *parent_ID* Feld mitgeliefert wird. Dies kann entweder über GET im Skriptaufruf von „entry_save.php“ geschehen oder über ein verstecktes („hidden“) Input-Feld.

Da es auch in SQL Zeichen mit besonderer Bedeutung gibt (wie z.B. ' oder " um einen String in der SQL-Query zu markieren), müssen diese wie in PHP entsprechend ersetzt werden. Dies übernimmt die Funktion addslashes(). Beim

Anzeigen der Daten werden die „\“ von der addslashes() Funktion dagegen automatisch entfernt (manuell wäre dies mit stripslashes() möglich). Es sollten jedoch anschließend (wie beim Gästebuch) die Sonderzeichen von HTML codiert werden.

```
<?
$link = mysql_connect("my.sqlserver.com", "user", "passwd");
mysql_select_db("database", $link);

$Zeit = date("Y-m-d H:i:s");

$Autor = addslashes($Autor);
$EMail = addslashes($EMail);
$Subject = addslashes($Subject);
$Text = addslashes($Text);

$return = mysql_query("INSERT INTO forum (parent_ID, Autor, EMail,
Subject, Text, Zeit) VALUES ($parent_ID, '$Autor', '$EMail', '$Subject', '$Text',
'$Zeit')", $link);

if ($return != false) {
    // zurück zum Forum
    header("Location: default.php");
} else {
    print "Fehler beim Speichern.<br>";
    print mysql_error();
}

mysql_close($link);
?>
```

5.4. Weitere Features des Forums

Mit dem bisher vorgestellten Funktionsumfang wird das Forum zwar im Grunde funktionieren, es lässt aber auch mit einem schönen Layout viele Funktionen, die andere Forensysteme bereitstellen, vermissen.

Sehr praktisch ist zum Beispiel ein Administrationsmodus, in den man über eine Passworteingabe wechselt und der mit HTTP-Authentifizierung implementiert werden kann. Der Administrator kann dann Einträge bearbeiten oder löschen, ohne dass er direkten Zugang zur Datenbank hat. Beim Löschen von Einträgen ist zu beachten, dass auch alle Untereinträge (rekursiv) entfernt werden müssen.

Baut man eine Registrierung der Forumsbenutzer ein, könnten auch User ihre Einträge nachträglich noch bearbeiten.

Mit einem Cookie oder mit registrierten Benutzern kann man die Felder *Name* und *E-Mail* bei der Erstellung eines Eintrags auch schon von vornherein ausfüllen, da diese meist dieselben bleiben werden.

Damit man schnell erkennt, welche Einträge neu im Forum sind, kann man sich am einfachsten die Fähigkeit von Browsern zunutze machen, bereits besuchte Links farblich hervorzuheben. Trotzdem ist eine separate Seite mit den 10 (oder 20 etc.) aktuellsten Einträgen (nicht als Baum, sondern als Liste) recht praktisch.

Bei einem ausgewählten Eintrag ist es recht leicht ein kleines Navigationsmenü einzublenden, also z.B. Buttons für „eine Ebene hinauf“ und „zur untersten Ebene“. Zudem kann man die Zahl der Untereinträge ermitteln und bei jedem Eintrag anzeigen.

Mit einer Datenbank eigentlich nicht schwer zu realisieren ist eine Suche in allen Forumseinträgen. Die SQL-Query muss dazu etwa „SELECT * FROM forum WHERE Text LIKE '%\$Suchbegriff%'“ lauten. Die Kopfzeilen der gefundenen Einträge können dann als einfache Liste angezeigt werden.

Oft zu sehen ist auch eine Option „E-Mail-Benachrichtigung bei Antworten“, die man beim Erstellen eines Eintrags wählen kann. Der Server schickt dann dem Verfasser eines Beitrags ein E-Mail, wenn jemand auf dessen Eintrag reagiert, vorausgesetzt, eine E-Mail-Adresse wurde angegeben.

6. Quellenverzeichnis

- www.php.net – offizielle PHP-Seite
- PHP 4.0-Manual (von www.php.net)
- www.zend.com – offizielle Seite von Zend Technologies Ltd.
- php, Egon Schmid et al, 1999 Markt&Technik
- Netscape Cookie-Spezifikationen:
http://www.netscape.com/newsref/std/cookie_spec.html.