PHP

Stand: 15.08.2022

PHP als Script-Sprache

Dieses Kapitel richtet sich an Betreiber von Websites, die einen Bedarf an serverseitiger Programmierung haben und sich dabei für eine Lösung mit **PHP** entscheiden. Ein solcher Bedarf kann schnell entstehen:

- Daten aus Feedback-Formularen auf Webseiten sollen verarbeitet werden, z.B. an den Site-Betreiber gemailt oder auf dem Server gespeichert werden.
- Das Projekt besteht aus sehr vielen Seiten, die leichter wartbar sind, wenn mit HTML-Templates gearbeitet wird statt mit statischen HTML-Dateien.
- Es sollen Daten in Seiten eingebunden werden, die in anderer Form gespeichert sind, z.B. in Textdateien, in XML-Dateien oder in Datenbanken.
- Bestimmte Bereiche im Webangebot sollen nur registrierten Anwendern zugänglich sein. Dabei soll jedoch eine eigene Benutzerverwaltung und Zugriffsüberwachung realisiert werden.
- Der Anwender soll nicht nur Informationen abrufen, sondern auch Aktionen durchführen können, wie z.B. in einem elektronischen Shop einkaufen, Kommentare zu Inhalten verfassen, in Foren diskutieren oder sich in einem Gästebuch eintragen.
- Der Anwender soll für ihn individualisierte Seiten angeboten bekommen.

Die Liste ließe sich beliebig weiter fortführen.

PHP ist für all diese Anwendungsfälle und für viele andere gerüstet. Keine andere Webprogrammiersprache bietet so viele vordefinierte Funktionen an, um mit möglichst wenig eigenem Programmieraufwand möglichst viel zu erreichen. Nicht einmal, wenn Sie dynamisch PDF-Dokumente oder Grafiken oder Flash-Movies erzeugen wollen, lässt sie PHP im Stich, und erst recht nicht, wenn es um Datenbankanbindung, Mailversand, Socketverbindungen für beliebige Anwendungsprotokolle, XML oder DOM geht. Vom einfachen Form-Mailer über einen anwenderfreundlichen Shop bis hin zum Content-Management-System lässt sich PHP für so ziemlich alle Bereiche einsetzen.

> PHP ist so konzipiert, dass der Einstieg leicht fällt.

Erste Erfolge sind meist schnell erzielt. Vor allem macht PHP den Umstieg von statischem HTML sehr einfach, da sich PHP-Code auch einfach mitten im ansonsten statischen HTML-Code notieren lässt.

Gleichzeitig ist die Sprache aber auch flexibel genug, um den Anforderungen professioneller Softwareentwicklung gerecht zu werden. Von objektorientierter Programmierung bis hin zu einem ausgeklügelten Include-Automatismus steht fast alles zur Verfügung, was jenseits von „Quick & Dirty-Programmierung benötigt wird.

PHP ist zwar vom Ansatz her leicht erlernbar, kann aber auch sehr komplex werden. Das betrifft weniger die syntaktischen Konstrukte – diese sind in der Regel besser lesbar als etwa bei der `Shell`-Programmierung unter Unix oder auch bei manchen Konstrukten in `Perl`. Schwer überschaubar ist aber dagegen die unglaubliche Menge an vordefinierten Funktionen, von denen viele ihre Besonderheiten haben.

Vieles ist in Programmiersprachen, die letztlich auf der Welt der Programmiersprache C basieren, syntaktisch gleich oder ähnlich gelöst, beispielsweise die verfügbaren Variablentypen, die möglichen Konstrukte bei Kontrollstrukturen, das Prinzip von Funktionen, Parametern und Rückgabewerten und ein Basis-Set an Funktionen für verschiedene Zwecke. PHP gehört ebenso wie `JavaScript` zu diesen Sprachen. Wenn Sie bereits die Kapitel über `JavaScript` und `DOM` durchgearbeitet haben, werden Sie in dieser Hinsicht in PHP einiges Bekanntes wiederfinden und können von dem bereits erworbenen Know-how profitieren.

Auf den meisten Root-Servern, die sich heute bei Hosting-Providern anmieten lassen, ist **PHP** längst vorinstalliert. Da die Sprache jedoch weiterentwickelt wird und in zahlreichen neuen Zwischenversionen erscheint, sind hin und wieder Updates sinnvoll. Allerdings muss dabei auch vor Unvorsichtigkeit gewarnt werden: PHP sorgte schon in manch einer Zwischenversion für böse Überraschungen. Scripts liefen plötzlich nicht mehr oder falsch, weil bestimmte Funktionen andere Parameter erwarteten oder eine andere Art von Rückgabewert erzeugten. Vor einem `PHP-Upgrade` in einer Produktivumgebung ist daher stets erst ein Upgrade in einer möglichst ähnlichen Testumgebung zu empfehlen.

Auf jeden Fall sollten Sie sich PHP auch in Ihrer lokalen Entwicklungsumgebung, also auf Ihrem PC installieren. Nachdem dort nach Lektüre der vorangegangenen Kapitel auch schon der Apache Webserver installiert sein sollte, ist die Installation von PHP nur der nächste logische Schritt.

Download und Versionen

PHP können Sie von php.net downloaden. Derzeit werden sowohl die 4er-Versionsreihe als auch die 5er-Versionsreihe weitergepflegt. Für Neuinstallationen und neu zu erstellende Webprojekte spricht nichts dagegen, gleich eine 5er-Version zu nehmen.

Für MS Windows stehen fertig kompilierte Fassungen zur Verfügung. Am einfachsten ist es, eine „Installer"-Variante herunterzuladen. Damit können Sie PHP wie andere Windows-Anwendungen auch bequem über ein Setup-Programm installieren. Für Unix/Linux-Umgebungen stehen die üblichen `tar.gz`– oder `tar.bz2`-Formate zur Verfügung. Auch für andere Betriebssysteme, wie Macintosh OS oder AS/400 werden Versionen angeboten.

> PHP ist Open Source.

Sie können es also ohne Lizenzgebühren downloaden und nutzen, auch für kommerzielle Webanwendungen.

Voraussetzung für den Betrieb von PHP als Script-Sprache für Webseiten ist eine funktionierende

Webserver-Umgebung. Da PHP mit dem Apache Webserver optimal zusammenarbeitet, ist dieser Webserver wärmstens zu empfehlen.

Installation unter Unix/Linux

Zur Installation benötigen Sie `root`-Rechte. Gehen wir von der Annahme aus, dass Sie eine `tar.gz`-Datei von PHP auf den Rechner geladen haben, auf dem es installiert werden soll. Zunächst müssen Sie die Datei dekomprimieren:

```
gunzip php-xxx.tar.gz
```

Dabei steht xxx für die Versionsnummer. Achten Sie also auf den genauen Dateinamen.

Als Nächstes entpacken Sie die entsprechende Archivdatei:

```
tar -xvf php-xxx.tar
```

Stellen Sie vor der Installation sicher, dass ein `Ansi-C-Compiler` auf dem System installiert ist. Das Standardprodukt auf Linux-Distributionen ist der **GNU C-Compiler GCC**. Über die Shell können Sie beispielsweise Folgendes eingeben:

```
gcc --help
```

Wenn dann Versionsinformationen und Aufrufmöglichkeiten zu GCC erscheinen, ist alles in Ordnung.

Als Nächstes sind grundsätzliche Konfigurationen für die Installation nötig. Das betrifft vor allem den Ablageort der endgültigen Installation sowie den Ablageort der zentralen PHP-Konfigurationsdatei **php.ini**. Dazu dient das .configure-Script. Zahlreiche weitere Einstellungen, die mit diesem Script durchführbar sind, können jedoch später in Ruhe auch in der **php.ini** oder durch erneuten Aufruf des Scripts vorgenommen werden. Daher genügt zunächst eine Grundkonfiguration wie diese:

```
./configure --prefix=/usr/local/bin/php --with-config-file-path=/etc
```

In diesem Beispiel wird PHP im Verzeichnis /usr/local/bin/php installiert. Die wichtige Konfigurationsdatei **php.ini** wird im zentralen Konfigurationsverzeichnis /etc abgelegt. Dies ist übrigens durchaus der übliche Ablageort dieser Datei. Anschließend kann der Kompilierungsvorgang gestartet werden:

```
make
```

Nachdem dieser Vorgang abgeschlossen ist, kann die Installation gestartet werden:

```
make install
```

Die zentrale Konfigurationsdatei **php.ini** existiert nach der Installation noch nicht. Bei der Distribution ist lediglich eine Beispieldatei unter dem Namen **php.ini-dist** dabei. Kopieren Sie diese aus dem Installationsverzeichnis ins Verzeichnis /etc unter dem endgültigen Namen:

```
cp php.ini-dist /etc/php.ini
```

Installation unter MS Windows

Wenn Sie wie empfohlen eine Windows-Installer-Version benutzen, können Sie sich auf das Setup verlassen. Dabei können Sie auch das Verzeichnis für den installierten Zustand festlegen.

Auch für die Windows-Distribution gilt: Die wichtige Konfigurationsdatei **php.ini** existiert zunächst noch nicht. Sie muss erst erstellt werden. Außerdem muss sie im Pfad liegen. Kopieren Sie dazu aus dem endgültigen Ablageverzeichnis von PHP die dort vorhandene Datei **php.ini-dist** in Ihr Windows-Verzeichnis, also z.B. in c:\windows oder c:\winnt. Benennen Sie die Datei dort um in **php.ini**.

Ferner wird empfohlen, aus dem endgültigen Ablageverzeichnis von PHP die dort vorhandene Datei php4ts.dll bzw. php5ts.dll in Ihr Windows-Systemverzeichnis zu kopieren, also z.B. in c:\windows\system32 oder c:\winnt\system32.

Es gibt zwei Möglichkeiten, **PHP-Scripts** von Apache ausführen zu lassen: entweder als Apache-Modul oder als CGI-Scripts. Die Lösung als Apache-Modul ist dann vorzuziehen, wenn PHP intensiv eingesetzt werden soll und auch Priorität gegenüber anderen Script-Sprachen auf dem Server haben soll. Außerdem entfallen bei der Variante „PHP als Modul“ die typischen CGI-Merkmale wie Beschränkung auf bestimmte Verzeichnisse usw. Nachteil ist jedoch, dass Apache damit insgesamt mehr Arbeitsspeicher pro Prozess benötigt, und zwar auch dann, wenn gar keine PHP-Scripts ausgeführt werden müssen. Wenn PHP als Apache-Modul läuft, laufen PHP-Scripts auf Unix-/Linux-Umgebungen auch unter dem Benutzernamen von Apache, sie haben also die gleichen Rechte wie dieser.

> Auf den meisten öffentlichen Server-Rechnern ist PHP als Apache-Modul konfiguriert.

Einbindung unter Unix/Linux

Um PHP als Apache-Modul zu konfigurieren, müssen Sie die Installation von PHP korrigieren, indem Sie in dem Verzeichnis, in dem Sie PHP dekomprimiert und entpackt haben, das .configure-Script erneut aufrufen mit:

```
./configure --with-apxs=[Pfad/zu/apxs]
```

Dabei ist apxs eine ausführbare Datei von Apache. Wo sie genau liegt, können Sie herausbekommen durch Eingabe von:

```
find / -name apxs
```

Nachdem das `.configure`-Script durchgelaufen ist, müssen Sie erneut `make` und `make install` ausführen.

Wechseln Sie nun in das Verzeichnis, in dem der Apache Webserver installiert ist, also z.B. in /usr/local/bin/apache, und dort ins Unterverzeichnis libexec. Dort sollte sich nun eine Datei libphp4.so oder libphp5.so befinden, je nachdem, ob Sie PHP 4.x oder PHP 5.x installiert haben.

Sind diese Voraussetzungen erfüllt, müssen Sie in der **httpd.conf**, also in der zentralen Konfigurationsdatei des Apache Webservers, folgende Einträge vornehmen:

- Das Apache-Modul von PHP muss geladen werden. Das geschieht für PHP 4.x durch folgenden Eintrag:

```
LoadModule php4_module  libexec/libphp4.so
```

Und für PHP 5.x durch folgenden Eintrag:

```
LoadModule php5_module  libexec/libphp5.so
```

Notieren Sie die Einträge innerhalb der httpd.conf dort, wo `LoadModule`-Einträge für dynamische Module (DSO) stehen sollten.

- Suchen Sie als Nächstes in der httpd.conf nach AddType. Notieren Sie zusätzlich zu eventuell vorhandenen Einträgen dieser Art den folgenden:

```
AddType application/x-httpd-php .php
```

- Bei der Direktive **DirectoryIndex** sollte ein Dateiname des Typs .php mit aufgenommen werden, in den meisten Fällen index.php. Das gilt auch für eventuell definierte virtuelle Hosts.

Möchten Sie dagegen PHP nicht als Apache-Modul einbinden, sondern einfach als möglichen Handler für CGI-Scripts, sind folgende Änderungen in der httpd.conf erforderlich:

- Überprüfen Sie, ob das CGI-Modul des Apache geladen wird. Dazu muss folgende Zeile ohne Kommentarzeichen davor existieren:

```
LoadModule cgi_module modules/mod_cgi.so
```

- Stellen Sie sicher, dass mindestens eine geeignete `ScriptAlias`-Direktive existiert, z. B.: ScriptAlias `/cgi-bin/ "/var/www/cgi-bin/"`
- Suchen Sie nach folgender Zeile: `AddHandler cgi-script .cgi`. Notieren Sie daran anschließend folgende beiden Zeilen:

```
Action php-script /cgi-bin
AddHandler php-script .php
```

Nachdem Sie die httpd.conf geändert haben, speichern Sie die Datei und führen Sie einen Neustart des Apache Webservers durch

Einbindung unter MS Windows

Unter MS Windows sollten Sie zunächst die Dateien php4apache.dll bzw. php4apache2.dll (je nach Apache-Version) aus dem PHP-Unterverzeichnis `sapi` in Ihr Windows-Systemverzeichnis kopieren, also z.B. in c:\windows\system32 oder c:\winnt\system32. Dadurch liegen diese Dateien zusätzlich im Pfad und mögliche Probleme des Apache Webservers beim Laden dieser Bibliotheken werden vermieden.

Auch unter Windows können Sie PHP wahlweise als Apache-Modul oder als Handler für **CGI-Scripts** einbinden. In beiden Fällen müssen Sie die zentrale httpd.conf des installierten Apache Webservers bearbeiten.

Um PHP als Apache-Modul zu installieren, sind folgende Einträge erforderlich:

- Das Apache-Modul von PHP muss geladen werden. Das geschieht für PHP 4.x durch folgenden Eintrag:

```
LoadModule php4_module  [Pfad zu PHP]/sapi/php4apache*.dll
```

Und für PHP 5.x durch folgenden Eintrag:

```
LoadModule php5_module  [Pfad zu PHP]/sapi/php5apache*.dll
```

Dabei ist `[Pfad zu PHP]` der vollständige Laufwerkspfad zum Programmverzeichnis von PHP, also etwa `c:/php` oder `c:/Programme/php`. Benutzen Sie einfache Schrägstriche zur Verzeichnistrennung, nicht

den unter Windows üblichen Backslash. Welche Datei Sie genau adressieren müssen, hängt von der PHP- und der Apache-Version ab. Bei PHP 4.x und Apache 1.3.x heißt sie php4apache.dll, bei PHP 4.x und Apache 2.0.x php4apache2.dll, bei PHP 5.x und Apache 1.3.x php5apache.dll und bei PHP 5.x und Apache 2.0.x php5apache2.dll.

Notieren Sie die Einträge innerhalb der httpd.conf dort, wo `LoadModule`-Einträge für dynamische Module (DSO) stehen sollten.

- Suchen Sie als Nächstes in der httpd.conf nach AddType. Notieren Sie zusätzlich zu eventuell vorhandenen Einträgen dieser Art den folgenden:

```
AddType application/x-httpd-php .php
```

- Bei der Direktive DirectoryIndex sollte ein Dateiname des Typs .php mit aufgenommen werden, in den meisten Fällen index.php. Das gilt auch für eventuell definierte virtuelle Hosts.

Möchten Sie dagegen PHP nicht als Apache-Modul einbinden, sondern einfach als möglichen Handler für CGI-Scripts, sind folgende Änderungen in der httpd.conf erforderlich:

- Überprüfen Sie, ob das CGI-Modul des Apache geladen wird. Dazu muss folgende Zeile ohne Kommentarzeichen davor existieren:

```
LoadModule cgi_module modules/mod_cgi.so
```

- Stellen Sie sicher, dass mindestens eine geeignete `ScriptAlias`-Direktive existiert, z. B.: ScriptAlias `/cgi-bin/ "c:/webdocs/cgi-bin/"`
- Suchen Sie nach folgender Zeile: `AddHandler cgi-script .cgi`. Notieren Sie daran anschließend folgende beiden Zeilen:

```
Action php-script /cgi-bin
AddHandler php-script .php
```

Nachdem Sie die httpd.conf geändert haben, speichern Sie die Datei und führen Sie einen Neustart des Apache Webservers durch.

Austesten von PHP unter Apache: ein erstes Script

Wenn Sie alles richtig installiert haben, sollten PHP-Scripts nun ausführbar sein. Wir nehmen in unseren nachfolgenden Beispielen an, dass PHP als **Apache-Modul** installiert ist. PHP-Dateien können dann überall unterhalb der Document Root, also unterhalb des Startverzeichnisses für Webdokumente, abgelegt werden. Erstellen und bearbeiten lassen sie sich mit jedem Texteditor. Bestens geeignet sind auch code-

basierte HTML-Editoren. Viele davon unterstützen von Haus aus Syntax-Highlighting für PHP-Code, was in der Praxis sehr hilfreich ist.

Folgendes Listing eignet sich, unter einem Dateinamen wie php-test.php irgendwo unterhalb der Document Root abgespeichert, zu einem ersten Test, ob PHP korrekt funktioniert:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>PHP-Test</title>
</head>
<body>
<h1><?php echo "Hier ist PHP!" ?></h1>
</body>
</html>
```

Wenn es in Ihrem Browser so aussieht wie abgebildet, ist PHP korrekt installiert und in Apache eingebunden.

Das Beispiel zeigt auch schon, wie simpel PHP sein kann. Unser Listing sieht aus wie eine gewöhnliche HTML-Datei. Nur an einer Stelle, nämlich im Elementinhalt der `h1`-Überschrift, ist ein Bereich notiert, der mit `<?php` beginnt und mit `?>` endet. Innerhalb davon kann PHP-Code stehen. In unserem Beispiel wird einfach nur etwas Text ausgegeben.

Wenn Sie sich im Browser den **HTML-Quelltext** ansehen („`view source`„), dann werden Sie feststellen, dass von dem PHP-Script-Bereich nichts mehr zu sehen ist. Dort steht innerhalb von `<h1>...</h1>` nur noch der von PHP ausgegebene Text.

> Genau das ist das Prinzip serverseitiger Scripts.

Die Scripts werden ausgeführt, bevor die Daten zum Browser übertragen werden. Ein serverseitiges Script muss dafür sorgen, dass sinnvolle Daten zum Browser gelangen, also z.B. HTML-Code. Beim Browser selbst kommt dann nur noch der ausgegebene HTML-Code an, aber nichts mehr vom Programmcode des Scripts. Der Browser weiß nicht einmal, dass ein Script den HTML-Code erzeugt hat, bzw. es ist ihm völlig egal. Er erhält seine Daten vom Webserver genauso, als würde es sich um eine statische HTML-Datei handeln.

Umgekehrt ist es Aufgabe des Webservers, richtig zu reagieren. Versuchen wir zu verstehen, was genau passiert:

- Der Webserver erhält vom Browser eine Anfrage, in der dieser mit `GET` oder `POST` dazu auffordert,

ihm die Daten einer Datenressource namens php-test.php zu senden.
- Der Webserver erkennt an der Endung, dass in diesem Fall der PHP-Interpreter ausgeführt werden muss. Die Endung ist ihm durch seine Konfiguration bekannt.
- Zunächst muss der Webserver prüfen, ob er PHP-Scripts im angeforderten Verzeichnis überhaupt ausführen darf. Angenommen, PHP ist als Modul konfiguriert, dann darf es in allen Webverzeichnissen ausgeführt werden.
- Der Webserver startet den als Modul geladenen **PHP-Interpreter** mit der angeforderten Scriptdatei. Der PHP-Interpreter führt das Script aus und reicht den vom Script erzeugten Datenoutput an den Webserver durch.
- Der Webserver sendet den Datenoutput an den anfragenden Browser.

Die Scripts testen

Während Sie beim Erstellen von **HTML-Seiten** diese direkt im Browser betrachten können, benötigen Sie zum Testen von PHP-Dateien einen Server, der **PHP** unterstützt. Arbeiten Sie an Ihrem heimischen Computer und nicht am Server, müssen Sie die Scriptdateien also unter Umständen auf den Server hochladen. Sie benutzen dazu irgendein Standard-FTP-Programm, z.B. WS_FTP.

Info
FTP (File Transfer Protocol) ist ein Dateiübertragungsprotokoll, mit dem Sie Texte oder binäre Dateien zwischen Ihrem Computer und einem FTP-Computer im Internet austauschen können. In den meisten Fällen benötigen Sie für den FTP-Zugang eine bestimmte Zugangsberechtigung; es gibt jedoch auch so genannte anonyme FTP-Server, die jedem Benutzer Zugang gewähren, um beispielsweise kostenlose Software herunterzuladen oder Dokumente zu beziehen.

Sobald Sie eine Verbindung zum FTP-Server aufgebaut haben, können Sie Dateien mehr oder minder so, wie im Windows-Explorer üblich, in ein entsprechendes Verzeichnis auf den Server kopieren.

Das Grundgerüst eines PHP-Scripts

Sie brauchen zum Schreiben eines PHP-Scripts, wie Sie es von HTML gewohnt sind, einen Texteditor. Ein reines **PHP-Script** ohne HTML-Elemente sieht dann in seiner Struktur zunächst ganz einfach aus: Die ausführbaren PHP-Codes werden in die folgenden Anfangs- und Endezeichen eingeschlossen:

```
<?php
PHP-Code;
?>
```

Wie Sie noch sehen werden, spielt die Groß- und Kleinschreibung vielfach eine Rolle, allerdings nicht beim Anfangszeichen. Es könnte auch <?PHP heißen. Eventuell können Sie auch bei Ihrem Provider nachfragen (bzw. in der Tabelle mit Informationen über die spezifische Installationsumgebung nachsehen, die im nächsten Abschnitt vorgestellt wird), inwieweit Sie kurze Tags benutzen können, also <? und ?> an Stelle von <?php und ?>, oder ob ASP-Tags akzeptiert werden: <% und %>. Manche Programme, beispielsweise

Dreamweaver, funktionieren mit den ASP-Tags besser als mit den PHP-Tags.

> Jede PHP-Anweisung wird mit einem Semikolon beendet. Damit wird PHP quasi mitgeteilt,

wo der Befehl, der ausgeführt werden soll, zu Ende ist. Es bietet sich wegen der Übersichtlichkeit an, danach jeweils in eine neue Zeile zu springen (auch wenn dies von der Sache her nicht unbedingt erforderlich ist). Ein Semikolon nach einem Code zu vergessen, ist eine oft vorkommende Fehlerquelle, versuchen Sie also immer daran zu denken. Manche Programmierer schreiben das Semikolon grundsätzlich zuerst und dann die anderen Eingaben der Programmzeile. Dies ist ein Trick, der sicherlich dafür sorgt, das Semikolon weniger häufig zu vergessen!

Zusammen mit HTML enthält ein Script die entsprechenden HTML-Tags und den **PHP-Code**, der in den Body-Container geschrieben und abgegrenzt wird. Das sieht dann als Grundgerüst so aus:

```
<html>
<head>
<title>HTML_PHP</title>
</head>
<body>
<?php
PHP-Code;
?>
</body>
</html>
```

Sie können beliebig viele PHP-Codes in ein Dokument einfügen und mit den jeweils benötigten HTML-Tags kombinieren. Sie müssen nur jedes Mal auf das Startzeichen <?php und das Endezeichen ?> achten. Oftmals ist auch eine andere Reihenfolge als die hier vorgestellte sinnvoll, z.B. das Script mit <?php zu beginnen, anstatt mit dem üblichen HTML-Header.

Grundsätzlich arbeitet der PHP-Prozessor die Datei der Reihe nach von oben bis unten ab; wann Sie welche Anweisung wohin schreiben, spielt also eine große Rolle. Reines HTML wird unverändert zurückgegeben, die PHP-Anweisungen werden ausgewertet und ausgeführt.

Speichern

HTML-Seiten, die PHP-Elemente enthalten bzw. reine PHP-Scripts sind, speichern Sie mit der Erweiterung .php. Dies ist die Standarderweiterung für PHP4, und deswegen werden wir in diesem Kapitel diese Erweiterung benutzen. Arbeiten Sie mit einem Server, auf dem PHP3 installiert ist, benötigen Sie gegebenenfalls die Erweiterung .php3. Sofern Sie nicht direkt am Server arbeiten, muss das Dokument mithilfe eines FTP-Programms auf den Server hochgeladen werden. Davon war weiter oben schon die

Rede. Bei den folgenden Beschreibungen gehen wir davon aus, dass Sie am Server arbeiten, sodass Sie die Scripts testen können, indem Sie die Dateien einfach im Browser aufrufen. (Folglich heißt es bei uns dann lediglich: Speichern Sie das Script und testen Sie es im Browser oder so ähnlich.)

Ausgabe

Damit Sie so bald wie möglich ein Script im Browser testen können, lernen Sie hier (und in den meisten Büchern über PHP) als Erstes den **PHP-Befehl** `echo` kennen. Echo gibt alle Zeichenketten im Browser aus. Unter einer Zeichenkette, auch als String bezeichnet, versteht man in PHP eine Reihe von Zeichen, die aus einer beliebigen Kombination von Buchstaben, Zahlen, Symbolen und Leerstellen (und Variablen) bestehen kann und in einfache oder doppelte Anführungszeichen eingeschlossen wird.

Schauen Sie sich das einmal an:

- 1. Öffnen Sie ein leeres Dokument und schreiben Sie:

```
<?php
echo "ich lerne PHP!";
?>
```

- 2. Speichern Sie das Dokument als Datei mit der Erweiterung .php.
- 3. Öffnen Sie nun diese Datei im Browser. Dort müssten Sie nun lesen:

```
ich lerne PHP!
```

Um eine bestimmte Formatierung (und/oder Seitengestaltung) zu erreichen, können Sie PHP und HTML auch mischen, indem Sie zwischen PHP und HTML hin und her wechseln. Eine Möglichkeit könnte so aussehen:

```
<html>
<head>
<title>HTML_PHP</title>
</head>
<body>
<b>
<?php
echo "ich lerne PHP";
?>
</b>
<p><b>hier kommt normale HTML-Ausgabe</b></p>
<?php
echo "hier steht ein neuer PHP-Code";
```

```
?>
</body>
</html>
```

Bei dieser Methode beenden Sie also PHP, schreiben den HTML-Teil mit den jeweils benötigten Tags und beginnen PHP erneut.

Es ist aber auch möglich, HTML als PHP auszugeben. Dies ist für PHP-Einsteiger mitunter etwas verwirrend. Sie müssen sich klar machen, dass in dem Fall alles, was der Browser anzeigen soll, Teil des PHP-Codes sein und ausgegeben werden muss, auch Formatierungen, Zeilenumbrüche und Ähnliches. Ein Scriptfragment würde dann etwa so aussehen:

```
<?php
echo "<b>ich lerne PHP</b><br>";
echo "<h2>eine Überschrift</h2>";
echo "<hr>";
echo "neuer PHP-Code";
?>
```

Der Befehl `echo` funktioniert durch die Verwendung des HTML-Tags für einen Zeilenumbruch auch über mehrere Zeilen. Wenn Sie schreiben:

```
echo "mein Hut der hat vier Ecken <br> vier Ecken hat mein Hut!";
```

wird als Ausgabe im Browser zu lesen sein:

```
Mein Hut der hat vier Ecken
vier Ecken hat mein Hut!
```

Info
Leerzeilen im Script haben keinerlei Einfluss auf das Erscheinungsbild der Seite, sie können aber das Script selbst mitunter übersichtlicher machen. Auch per Tabulator eingerückte Teile erhöhen die Lesbarkeit eines Scripts und sicherlich werden Sie feststellen, dass es sinnvoll ist, **HTML** und **PHP** optisch zu trennen.

PHP-Info

Es gibt eine gute Möglichkeit bzw. eine spezielle Funktion, sich grundlegende Informationen über PHP einzuholen. Diese Funktion lautet:

```
phpinfo()
```

Mit dieser Funktion wird eine Tabelle an den Browser gesendet, die Informationen über die spezifische PHP-Installation auf dem fraglichen Server enthält. Sie können durch den Einsatz dieser Funktion eine ganze Menge über die Server-Umgebung und die Konfiguration erfahren, z.B. welche Erweiterungen benutzt werden können, welche Datenbank verwendet wird etc. Vor allem, wenn Sie nicht am Server arbeiten und PHP nicht selbst installiert haben, bietet es sich an, einen Blick auf diese Tabelle zu werfen. Sie brauchen ein ganz einfaches „Script“:

```
<?php
echo phpinfo();
?>
```

Der Browser zeigt daraufhin die erwähnte Tabelle mit den Informationen über PHP. Riskieren Sie ruhig einen Blick.

In der Vergangenheit war die Installation eines **LAMP-Systems** (Linux + Apache + MySQL + PHP/Perl) ein Kinderspiel. Alle gängigen Distributionen lieferten die erforderlichen Pakete gleich mit. Im Wesentlichen bestand die ganze Arbeit darin, den Paketmanager zu starten, die Pakete zu installieren und schließlich Apache und MySQL zu starten. Je nach Distribution sehen die Startkommandos so aus:

```
root# /etc/init.d/httpd start     # Apache bei Red Hat, Fedora
root# /etc/init.d/apache2 start   # Apache bei SUSE
root# /etc/init.d/mysql start     # MySQL bei Red Hat, Fedora und SUSE
```

Damit die Server in Zukunft immer automatisch gestartet werden, führen Sie die folgenden Kommandos aus:

```
root# chkconfig --add httpd   # Apache bei Red Hat, Fedora
root# chkconfig --add mysql   # MySQL bei Red Hat, Fedora
root# insserv apache2         # Apache bei SUSE
root# insserv mysql           # MySQL bei SUSE
```

Info
Bei Red Hat bzw. beim von Red Hat geleiteten Fedora-Team ist man der Ansicht, dass die Lizenz von **MySQL** ab Version 4.0 eine Auslieferung nicht zulässt. Der strittige Punkt sind die Client-Bibliotheken, die seit MySQL 4.0 der GPL unterstehen. Diese relativ strenge Open-Source-Lizenz ist inkompatibel mit einigen anderen Open-Source-Projekten, die eine liberalere Lizenz verwenden, unter anderem PHP.

MySQL hat deswegen eine Ausnahmeregel (FOSS) für die Client-Bibliotheken definiert, die die gemeinsame Weitergabe mit anderen Open-Source-Projekten explizit erlaubt. Aber offensichtlich hat auch das Red Hat nicht überzeugen können. Vielleicht sollte man noch erwähnen, dass Red Hat in der Vergangenheit sehr stark auf PostgreSQL gesetzt hat und daher möglicherweise kein ganz neutrales Verhältnis zu MySQL hat

...

Aktuelle Versionen von PHP und MySQL installieren

Welche Möglichkeiten bestehen nun, aktuelle Versionen von PHP und MySQL auf einer gängigen Linux-Distribution zu installieren?

- Die bequemste Lösung lautet XAMPP für Linux. Dabei handelt es sich um ein Komplettpaket, das aus Apache, PHP, MySQL und einigen weiteren Paketen besteht. Die Installation besteht aus nur einem `tar`-Kommando. Anschließend ist das Komplettpaket, das aus Apache, PHP, MySQL und zahlreichen weiteren Programmen besteht, bereit für den Testeinsatz.

Wenn Sie selbst die Kontrolle über jede Software-Komponente haben möchten, müssen Sie die aktuellen Versionen von PHP und MySQL selbst installieren. Bei MySQL ist das relativ einfach – unter MySQL finden Sie kompilierte Pakete für Linux.

Info
Im Folgenden wird vorausgesetzt, dass die von Ihrer Distribution mitgelieferten Apache-, PHP- und MySQL-Pakete nicht installiert sind!

XAMPP

XAMPP installieren

XAMPP für Linux besteht aus einem einzigen, sehr großen `tar`-Archiv, das Sie hier zum Download finden:

Die Installation ist wirklich denkbar einfach – ein einziges `tar`-Kommando reicht!

```
root# tar xvfz xampp-linux-1.4.9a.tar.gz -C /opt
```

Anschließend befinden sich sämtliche XAMPP-Komponenten im Verzeichnis /opt/lampp. Ein weiteres Kommando startet das System:

```
root# /opt/lampp/lampp start
```

Indem Sie die Seite **http://localhost** mit einem Webbrowser öffnen, können Sie sich davon überzeugen, dass alles funktioniert hat.

XAMPP absichern

Die Defaultinstallation von XAMPP ist nicht durch Passwörter abgesichert und daher unsicher. Das sollten Sie mit dem folgenden Kommando ändern:

```
root# /opt/lampp/lampp security
```

Sie können nun verschiedene Aspekte von XAMPP absichern:

- **XAMPP pages**: Damit sichern Sie den Zugang zu den von Apache erzeugten Webseiten zur XAMPP-Administration ab. Unabhängig davon, ob die Seiten vom lokalen Rechner oder extern aufgerufen werden, müssen Sie sich nun zuerst mit dem Benutzernamen lampp und dem von Ihnen angegebenen Passwort anmelden. (Die Absicherung erfolgt durch zwei .htaccess-Dateien in den Verzeichnissen /opt/lampp/xampp und /opt/lampp/phpmyadmin.)
- **MySQL network access**: Wenn Sie ausschließlich mit PHP auf den MySQL-Server zugreifen, erfolgt dieser Zugriff über eine so genannte Socket-Datei (nicht über ein Netzwerkprotokoll). Aus Sicherheitsgründen ist es empfehlenswert, die Netzwerkfunktionen des MySQL-Servers zu deaktivieren.
- **phpMyAdmin-Passwort**: Zur Administration des MySQL-Servers hat XAMPP **phpMyAdmin** installiert. Für einige Zusatzfunktionen von phpMyAdmin hat XAMPP einen eigenen MySQL-Benutzer eingerichtet, für den Sie nun ein Passwort definieren können. Das Passwort wird allerdings im Klartext in die Datei /opt/lampp/phpmyadmin/config.inc.php eingetragen (sonst funktioniert phpMyAdmin nicht). Verwenden Sie also nicht dasselbe Passwort, mit dem beispielsweise Ihr Login abgesichert ist!
- **MySQL-root-Passwort**: Die MySQL-Administration erfolgt durch den Benutzer `root`. Sie können nun für `root` ein Passwort angeben. Dieses Passwort gilt auch für phpMyAdmin. Dort müssen Sie sich in Zukunft mit dem Benutzernamen `root` und dem dazu passenden Passwort anmelden.
- **FTP-Passwort**: Als Teil von XAMPP wurde auch ein FTP-Server eingerichtet. Dieser ist mit dem Passwort lampp abgesichert. Wenn Sie ein anderes Passwort benutzen möchten, können Sie es jetzt angeben.

XAMPP verwenden

Nun können Sie mit der Entwicklung Ihrer ersten **PHP-Testprogramme** beginnen. Ihre eigenen PHP-Dateien speichern Sie beispielsweise in /opt/lampp/htdocs/meinebeispiele. Dabei müssen Sie darauf achten, dass die Dateien von Apache gelesen werden können, der unter dem Account `nobody` ausgeführt wird.

```
root# mkdir /opt/lampp/htdocs/meinebeispiele
root# chown benutzername /opt/lampp/htdocs/meinebeispiele
root# chgrp nobody /opt/lampp/htdocs/meinebeispiele
```

Nach diesen Vorbereitungsarbeiten sollten Sie aus Sicherheitsgründen unter ihrem normalen Account weiterarbeiten. (Anders als unter Windows ist es unter Linux verpönt, als `root` zu arbeiten, wenn dies nicht zu Administrationszwecken unbedingt erforderlich ist.)

XAMPP beenden

Bevor Sie den Rechner herunterfahren, sollten Sie XAMPP stoppen. Das ist insbesondere deswegen wichtig, damit der MySQL-Server alle offenen Datenbankdateien ordnungsgemäß schließen kann.

```
root# /opt/lampp/lampp start
```

XAMPP automatisch starten und stoppen

Jedes Mal, wenn Sie Ihren Rechner starten, müssen Sie auch XAMPP neu starten. Beim Ausschalten müssen Sie daran denken, XAMPP zu beenden. Wenn Sie dazu keine Lust haben, können Sie das auch automatisch erledigen. Dazu müssen Sie in das Init-V-System Ihrer Linux-Distribution zwei Links einfügen. (Das Init-V-System steuert, wann welche Systemdienste gestartet bzw. gestoppt werden.)

Die folgenden Kommandos zeigen die Vorgehensweise für SUSE Linux. Andere Distributionen verwenden ein wenig abweichende Verzeichnisnamen (z.B. Red Hat und Fedora /etc/rc5.d).

```
root# cd /etc/init.d/rc5.d
root# ln -s /opt/lampp/lampp S99lampp
root# ln -s /opt/lampp/lampp K01lampp
```

XAMPP deinstallieren

Ebenso einfach wie die Installation ist die Deinstallation:

```
root# rm -rf /opt/lampp/
```

Beachten Sie, dass Sie dadurch auch alle Ihre PHP-Dateien, MySQL-Datenbanken etc. verlieren. Gegebenenfalls sollten Sie vorher ein Backup machen. Die PHP-Dateien kopieren Sie einfach von /opt/lampp/htdocs/beispiele/ in ein anderes Verzeichnis (z.B. Ihr Heimatverzeichnis). **MySQL-Datenbanken** sichern Sie am bequemsten mit phpMyAdmin. Alternativ können Sie auch das Kommando `mysqldump` zuhilfe nehmen.

```
user$ cp -a /opt/lampp/htdocs/meinebeispiele ~
user$ /opt/lampp/binmysqldump -u root -p \ --default-characterset=latin1 \
meinedatenbank > ~/meinedatenbank.sql
Password: ********
```

MySQL

MySQL 4.1 oder 5.0 installieren

Vorweg ein kurzer Überblick: Dieser und der folgende Abschnitt beschreiben, wie Sie zuerst die vorkompilierten MySQL-Pakete von mysql.com installieren und dann Apache und PHP kompilieren. Für die ungewöhnliche Reihenfolge (zuerst MySQL) gibt es natürlich einen Grund: PHP kann nur dann mit MySQL-

Unterstützung kompiliert werden, wenn MySQL bereits installiert ist.

MySQL-Freaks können natürlich auch MySQL selbst kompilieren (mehr dazu im übernächsten Abschnitt). Das ist allerdings nur in Ausnahmefällen notwendig, weil es für MySQL (anders als für PHP!) vorkompilierte Pakete gibt.

MySQL deinstallieren

Dieser Abschnitt setzt voraus, dass die von Ihrer Distribution mitgelieferten **MySQL-Pakete** nicht installiert sind. Bei den meisten Distributionen können Sie das mit dem folgenden Kommando überprüfen:

```
user$ rpm -qa | grep -i mysql
```

Das Kommando listet alle installierten Pakete auf, in deren Namen mysql vorkommt. Diese Liste sollte leer sein. Wenn das nicht der Fall ist, verwenden Sie `rpm -e` oder den Paketmanager Ihrer Distribution, um die MySQL-Pakete zu deinstallieren.

MySQL-Pakete herunterladen

Im Folgenden wird vorausgesetzt, dass Sie eine Distribution verwenden, die die RPM-Paketverwaltung einsetzt (z.B. Red Hat, Fedora, Mandrakelinux oder SUSE). Vorkompilierte MySQL-Pakete für beinahe alle Betriebssysteme und für alle aktuellen Entwicklungszweige finden Sie unter MySQL. Sie benötigen die in Tabelle aufgelisteten Pakete:

Dateiname	Bedeutung
`MySQL-server-version.rpm`	Der eigentliche MySQL-Server
`MySQL-client-version.rpm`	Client-Werkzeuge (mysql, mysqldump etc.)
`MySQL-shared-compat-version .rpm`	Client-Bibliotheken (inklusive älterer Versionen dieser Bibliotheken zur Kompatibilität mit älteren Programmen, die auf den MySQL-Server zugreifen möchten)
`MySQL-devel-version .rpm`	Dateien zur Entwicklung/Kompilierung eigener MySQL-Clients (wichtig für die PHP-Kompilierung)

Die Installation ist denkbar einfach (und unabhängig davon, ob Sie sich für MySQL 4.1 oder 5.0 entscheiden):

```
root# rpm -i MySQL-*.rpm
```

MySQL-Server starten/beenden

Um den MySQL-Server zu starten, führen Sie das folgende Kommando aus:

```
root# /etc/init.d/mysql start
```

Um sich zu überzeugen, ob alles funktioniert hat, starten Sie mit dem Kommando `mysql` den MySQL-Kommandozeileninterpreter und führen dort das Kommando `status` aus. Mit `exit` oder Strg+D beenden Sie das Programm.

Ganz ähnlich wie das Startkommando sieht auch das Kommando aus, um den Server zu stoppen:

```
root# /etc/init.d/mysql stop
```

Falls Sie möchten, dass der Server in Zukunft automatisch gestartet bzw. beendet wird, wenn der Rechner hoch- bzw. heruntergefahren wird, führen Sie eines der beiden folgenden Kommandos aus:

```
root# insserv mysql         (für SUSE)
root# chkconfig --add mysql (für Red Hat, Fedora, Mandrakelinux etc.)
```

Info
Der MySQL-Server ist nach einer Neuinstallation nicht abgesichert. Aus Sicherheitsgründen sollten Sie die beiden Benutzer `root` mit einem Passwort ausstatten (einmal für den Netzwerkzugang und einmal für die Kommunikation über eine Socket-Datei) und die beiden anderen Defaultbenutzer löschen.

MySQL Administrator und MySQL Query Browser installieren

Der **MySQL Administrator** und der **MySQL Query Browser** erleichtern die lokale Administration sowie das Testen von SQL-Kommandos und SPs ungemein. Deswegen sollten Sie auch diese beiden Programme installieren.

Nachdem Sie die `tar`-Archivdateien von http://dev.mysql.com heruntergeladen haben, entpacken Sie die Archive in das Verzeichnis /usr/local:

```
root# cd /usr/local
root# tar xzf mysql-query-browser-<version>-linux.tar.gz
root# tar xzf mysql-administrator-<version>-linux.tar.gz
```

Anschließend ändern Sie in den Script-Dateien zum Start der beiden Programme die Variable MYPATH:

```
# Änderung in /usr/local/mysql-query-browser/bin/mysql-query-browser
...
MYPATH=/usr/local/mysql-query-browser/bin

# Änderung in /usr/local/mysql-administrator/bin/mysql-administrator
...
MYPATH=/usr/local/mysql-administrator/bin
```

Zu guter Letzt richten Sie noch zwei Links ein, damit jeder Benutzer des Systems die Programme bequem starten kann:

```
root# cd /usr/bin/X11
root# ln -s /usr/local/mysql-administrator/bin/mysql-administrator .
root# ln -s /usr/local/mysql-query-browser/bin/mysql-query-browser .
```

Apache 2 und PHP 5 kompilieren

Dieser Abschnitt beschreibt, wie Sie **Apache** und **PHP 5** selbst kompilieren. Dazu müssen einige Voraussetzungen erfüllt sein:

- Der Abschnitt setzt voraus, dass vorher alle mit der Distribution mitgelieferten Apache- und PHP-Pakete deinstalliert worden sind. Davon überzeugen Sie sich mit den beiden folgenden Kommandos, die beide kein Ergebnis liefern dürfen.

```
user$ rpm -qa | grep -i apache
user$ rpm -qa | grep -i php
```

- Damit die folgenden Kommandos funktionieren, müssen die üblichen Entwicklungswerkzeuge installiert sein (C-Compiler, make etc.). Falls Sie mit SUSE Linux arbeiten, installieren Sie einfach mit dem YaST-Modul SOFTWARE|SOFTWARE INSTALLIEREN ODER LÖSCHEN die Selektion C/C++ COMPILER UND WERKZEUGE.
- Weiters müssen die Entwicklungsdateien für diverse Bibliotheken installiert sein (Paketname libxxx-devel). Erforderlich sind unter anderem: libxml2, libexif, libjpeg und zlib.

Als Verzeichnis für alle weiteren Arbeiten wird /usr/local/src verwendet. Damit Sie dort als gewöhnlicher Benutzer arbeiten können, ändern Sie als `root` den Besitzer dieses Verzeichnisses:

```
root# chown benutzername.users /usr/local/src
```

Info
Fast alle Linux-Distributionen sehen einen Update-Service vor, der es sehr einfach macht, bei bekannten

Sicherheitsmängeln die betroffenen Programme zu aktualisieren. Dieser Update-Service gilt aber natürlich nur für die von der Distribution mitgelieferten Pakete, nicht für selbst kompilierte Programme. Für deren Aktualisierung bei Sicherheitsmängeln sind Sie selbst verantwortlich.

Bei einem Entwicklungsrechner ist das nicht so wichtig, aber falls Sie selbst einen **root-Server** betreiben, sollten Sie größte Vorsicht walten lassen! Sofort nach Bekanntwerden von Sicherheitsmängeln müssen Sie den neuesten Quellcode herunterladen, kompilieren, installieren und neu starten.

Apache 2 kompilieren

Den aktuellen Quellcode von Apache.

Nach dem Download extrahieren Sie das `tar`-Archiv, bereiten die Kompilation durch `configure` vor, kompilieren mit make und installieren die resultierenden Dateien schließlich mit `make install`. (Beachten Sie, dass `make install` von `root` ausgeführt werden muss. Falls Ihr `tar`-Archiv die Dateiendung .bz2 statt .gz hat, lauten die richtigen tar-Optionen xjf statt xzf.)

Das `configure`-Kommando ist hier nur aus Platzgründen auf zwei Zeilen verteilt. Bei einer Eingabe in einer Zeile entfällt das `\`-Zeichen. Die `configure`-Option `--prefix` gibt an, wohin Apache installiert werden soll. `--enable-so` gibt an, dass Apache DSO-Erweiterungen (dynamic shared objects) unterstützen soll. Eine derartige Erweiterung wird PHP sein. `--with-mpm=prefork` bestimmt das Threading-Modell, also die Art und Weise, wie der Webserver auf mehrere gleichzeitige Anfragen reagiert. `prefork` ist das Modell, das für die Zusammenarbeit mit PHP am besten geeignet ist. Es gibt unzählige weitere Optionen, die für unsere Standardinstallation aber nicht relevant sind. `configure --help` liefert die schier endlose Liste von Möglichkeiten.

```
user$ cd /usr/local/src
user$ tar xzf httpd-2.0.52.tar.gz
user$ cd httpd-2.0.52/
user$ ./configure --prefix=/usr/local/apache2 --enable-so \ --with-mpm=prefork
user$ make
root# make install
```

Damit Sie später als gewöhnlicher Benutzer Dateien in /usr/local/apache2/htdocs ändern dürfen, sollten Sie gleich die Zugriffsrechte für dieses Verzeichnis ändern.

```
root# chmod a+rw /usr/local/apache2/htdocs
```

PHP 5 kompilieren

Den Quellcode für offizielle PHP-5-Versionen finden Sie auf der ersten der beiden folgenden Seiten. Falls Sie stattdessen eine ganz aktuelle Entwicklerversion installieren möchten, finden Sie alle zwei Stunden (!) die neueste Version auf der zweiten Seite. (Vorsicht: Bei diesen Snapshots ist die Wahrscheinlichkeit von Fehlern recht groß.)PHP DownloadPHP Snapshots

Nach dem Download wiederholen sich die schon bekannten Kommandos `tar`, `.configure`, `make` und `make install`. Das `configure`-Kommando ist aus Platzgründen auf mehrere Zeilen verteilt. Bei einer Eingabe in einer Zeile entfallen die `\`-Zeichen.

Einige Worte zu den eingesetzten `configure`-Optionen: `--prefix` gibt an, wohin PHP 5 installiert werden soll. `--with-apxs2` bedeutet, dass PHP 5 als Erweiterungsmodul für Apache 2 kompiliert werden soll. Das nachfolgende Verzeichnis gibt den vorgesehenen Ort für Apache-Module an. `--with-libxml-dir` gibt an, wo sich die **XML-Bibliotheken** befinden. Analog gibt `--with-zlib-dir` den Pfad zur `zlib`-Bibliothek an.

`--with-mysql` gibt an, dass PHP mit der traditionellen `mysql`-Schnittstelle kompiliert werden soll. Der nachfolgende Pfad zeigt zum MySQL-Installationsverzeichnis, das bei den vorkompilierten MySQL-Paketen von dev.mysql.com einfach `/usr` lautet.

`--with-mysqli` gibt an, dass in PHP auch die neue `mysqli`-Schnittstelle integriert werden soll. Die nachfolgende Datei ist Teil des MySQL-devel-Pakets. Es handelt sich dabei um ein Script, das Informationen über die installierte MySQL-Version und deren Installationsorte gibt.

Die weiteren `--with-xxx` bzw. `--enable-xxx`-Optionen aktivieren diverse Zusatzfunktionen von PHP. Eine endlose Liste mit weiteren Optionen liefert das Kommando `./configure --help`.

```
user$ cd /usr/local/src
user$ tar xzf php-5.0.2.tar.gz
user$ cd php-5.0.2
user$ ./configure --prefix=/usr/local/php5 \
--with-apxs2=/usr/local/apache2/bin/apxs \
--with-libxml-dir=/usr/lib \
--with-zlib --with-zlib-dir=/usr/lib \
--with-mysql=/usr --with-mysqli=/usr/bin/mysql_config \
--with-jpeg-dir=/usr --enable-exif \
--with-gd --enable-soap --enable-sockets \
--enable-force-cgi-redirect
user$ make
root# make install
```

Info
Wenn beim Kompilieren unzählige multiple-defined-Fehler für die Bibliothek `libmysql` auftreten, hat `.configure` einen Fehler in das Makefile eingebaut. (Das ist bei unseren Tests manchmal passiert; es ist allerdings unklar geblieben, warum.)

Die Lösung besteht darin, vor der Ausführung von `make` die Datei **Makefile** in einen Editor zu laden. Dort sichen Sie die Zeile EXTRA_-LIBS = ... In dieser Zeile ist `-lmysqlclient` zweimal enthalten. Löschen Sie `-lmysqlclient` einmal, und speichern Sie die Datei.

Wenn Sie `./configure` mit veränderten Optionen neu ausführen, müssen Sie vor `make` das Kommando `make clean` ausführen. Es entfernt die Ergebnisse der vorherigen Kompilation.

PHP 5 wurde damit in Verzeichnis /usr/local/php5 installiert. Der Ort für die Konfigurationsdatei php.ini ist /usr/local/php5/lib/. Per Default existiert diese Datei allerdings noch nicht, d.h., PHP 5 läuft mit den Defaulteinstellungen (die einstweilen ausreichen). Muster für *.ini-Dateien finden Sie in /usr/local/src/php-5.n.

Apache-Konfiguration ändern

Der nächste Schritt besteht darin, die Apache-Konfigurationsdatei /usr/local /apache2/conf/httpd.conf so zu ändern, dass Apache das PHP-5-Modul verwendet. Dazu laden Sie die Datei in einen Editor und fügen die fett markierte Zeilen hinzu:

```
# Änderungen in /usr/local/apache2/conf/httpd.conf
...
LoadModule php5_module modules/libphp5.so
AddType application/x-httpd-php .php
...
### Section 2: ...
```

Apache starten, testen und beenden

Um Apache samt PHP zu starten, führen Sie als `root` das folgende Kommando aus:

```
root# /usr/local/apache2/bin/apachectl start
```

Um zu testen, ob Apache tatsächlich läuft, sehen Sie sich die Seite **http://localhost** auf Ihrem Webbrowser an. Um zu testen, ob auch PHP 5 funktioniert, erzeugen Sie die Testdatei /usr/local/apache2/htdocs/phptest.php. Diese Datei muss vom Apache-Account `nobody` lesbar sein.

```
<? phpinfo();
?>
```

Sehen Sie sich nun die Seite **http://localhost/testphp.php** auf Ihrem Webbrowser an. Das Ergebnis sollte wie in Abbildung aussehen.

Mit dem folgenden Kommando beenden Sie Apache wieder:

```
root# /usr/local/apache2/bin/apachectl stop
```

Denken Sie daran, dass Änderungen an **httpd.conf** oder php.ini erst wirksam werden, nachdem Sie Apache neu gestartet haben!

Apache automatisch starten und beenden

Jetzt wäre es noch wünschenswert, dass Apache automatisch gestartet wird, wenn Sie Ihren Rechner einschalten, und automatisch beendet wird, wenn Sie den Rechner herunterfahren. Dazu müssen Sie die Datei `apachectl` in den Init-V-Prozess Ihrer Distribution einbauen. (Das Init-V-System steuert, wann welche Systemdienste gestartet bzw. gestoppt werden.)

Die folgenden Kommandos zeigen die Vorgehensweise für SUSE Linux:

```
root# cp /usr/local/apache2/bin/apachectl /etc/init.d/apache
root# insserv apache
```

Bei Red Hat und Fedora gehen Sie so vor:

```
root# cp /usr/local/apache2/bin/apachectl /etc/rc.d/init.d/apache
root# chkconfig --add apache
root# chkconfig --level 35 apache on
```

`mysql`-Datenbank zur Verwaltung der Zugriffsrechte einrichten

Bevor der MySQL-Server erstmals gestartet werden kann, muss die Datenbank `mysql` zur Verwaltung der **MySQL-Zugriffsrechte** erstellt werden. Die folgenden Zeilen gehen davon aus, dass die Datenbankdateien in /usr/local/mysql gespeichert werden sollen und dass der MySQL-Server den Account `mysql` nutzt. (Falls auf Ihrem Rechner bereits bisher MySQL gelaufen ist, können Sie dessen Datenbanken weiterverwenden. In diesem Fall verzichten Sie auf die beiden folgenden Kommandos.)

```
root# ./scripts/mysql_install_db --ldata=/usr/local/mysql
root# chown -R mysql /usr/local/mysql
```

Konfigurationsdateien und Init-V-Script einrichten

Zum Start des MySQL-Servers benötigen Sie ein so genanntes Init-V-Script. In den MySQL-Quelldateien finden Sie mit support-files/mysql.server eine passende Vorlage. Zur Konfiguration verschiedener MySQL-Parameter dient die Datei /etc /my.cnf. Auch hierfür enthält das support-files-Verzeichnis eine Vorlage.

```
root# cp support-files/mysql.server /etc/init.d/mysql
root# cp support-files/my-medium.cnf /etc/my.cnf
```

Jetzt müssen Sie in den beiden Dateien noch kleine Änderungen vornehmen, die in den folgenden Zeilen fett hervorgehoben sind. /etc/my.cnf steuert, welche Socket-Datei Client und Server für ihre lokale Kommunikation verwenden sollen. Per Default verwendet der MySQL-Server /tmp/mysql.sock, PHP ist aber meist auf /var/lib/mysql/mysql.sock eingestellt.

```
# Änderungen in /etc/my.cnf
...
[client]
socket = /var/lib/mysql/mysql.sock
...
[mysqld]
socket = /var/lib/mysql/mysql.sock
...
```

/etc/init.d/mysql ist für den Start des MySQL-Servers zuständig. Dort geben Sie an, wo sich die MySQL-Binärdateien und wo sich die **MySQL-Datenbankdateien** befinden. (Falls auf Ihrem Rechner bereits bisher MySQL gelaufen ist, können Sie dessen Datenbanken weiterverwenden. In diesem Fall stellen Sie `datadir` so ein, dass es auf das bisherige Datenbankverzeichnis zeigt. Im Regelfall ist das /var/lib /mysql.)

```
# Änderungen in /etc/init.d/mysql
...
basedir=/usr/local
datadir=/usr/local/mysql
...
```

MySQL-Server starten

Um den MySQL-Server nun tatsächlich zu starten, führen Sie das folgende Kommando aus:

```
root# /etc/init.d/mysql start
```

Wenn Sie möchten, dass der Server in Zukunft automatisch gestartet wird, wenn der Rechner hochgefahren wird, führen Sie eines der beiden folgenden Kommandos aus:

```
root# insserv mysql         (für SUSE)
root# chkconfig --add mysql (für Red Hat, Fedora, Mandrakelinux etc.)
```

Die folgenden Abschnitte beschreiben die Einzelinstallation von Apache 2, PHP 5 und MySQL 4.1. Die Einzelinstallation hat den Vorteil, dass Sie die Version jeder Komponente selbst auswählen können und eher verstehen, welche Programme auf Ihrem Rechner installiert sind und wie sie konfiguriert werden. Das erleichtert auch die spätere Wartung bzw. das Update einer bestimmten Komponente.

Info

Noch bequemer ist die Installation, wenn Sie ein Komplettpaket verwenden, das alle drei Komponenten in sich vereint. Die beiden bekanntesten derartigen Pakete sind WAMP (TSW) sowie XAMPP:

Apache 2.0 installieren IIS deaktivieren

Falls Sie auf einer Windows-Server-Version arbeiten, sollten Sie als Erstes sicherstellen, dass der Microsoft Internet Information Server (kurz IIS) nicht läuft. Die Installation von **Apache** in der Standardeinstellung (Port 80) scheitert, wenn IIS parallel läuft (wozu es auch selten einen Grund gibt).

Zur Kontrolle bzw. zur Deinstallation des IIS führen Sie EINSTELLUNGEN|SYSTEM-STEUERUNG|SOFTWARE|WINDOWS-KOMPONENTEN aus. Gegebenenfalls deaktivieren Sie im ASSISTENT FÜR WINDOWS-KOMPONENTEN die Option IIS, woraufhin das Programm gestoppt und deinstalliert wird.

Apache installieren

Apache steht bei (bzw. auf diversen Mirrors) als Installationsprogramm für Windows zur Verfügung. Laden Sie die apache_2.n_xxx.msi-Datei auf Ihren Rechner, und starten Sie das Installationsprogramm im Windows-Explorer durch einen Doppelklick. Es erscheint ein Dialog zur Grundkonfiguration. Darin müssen Sie den Domain-Namen und den Server-Namen angeben. Diese Informationen werden auch dann benötigt, wenn Ihr Rechner nur als einzelner Testrechner läuft. Im Regelfall werden die richtigen Werte automatisch in die Dialogfelder eingetragen. Wenn Sie unsicher sind, wie diese Namen bei Ihrem Rechner lauten, können Sie im Dialog EINSTELLUNGEN| SYSTEMSTEUERUNG|SYSTEM|COMPUTERNAME nachsehen. Des Weiteren müssen Sie angeben, unter welcher E-Mail-Adresse der Administrator des Webservers (also Sie) erreichbar ist und ob der Webserver den Port 80 verwenden soll.

Nach der Installation wird Apache unter Windows NT/2000/XP automatisch als Service (Dienst) eingerichtet und sofort gestartet. Im rechten Teil der Taskleiste erscheint ein kleines Icon, das den Zustand von Apache anzeigt (ein grüner Pfeil, wenn das Programm läuft). Über dieses Icon können Sie Apache starten und stoppen.

Apache starten und stoppen

Sie können Apache mit den Menükommandos PROGRAMME|APACHE HTTP SERVER| CONTROL APACHE SERVER|START bzw. |STOP starten bzw. beenden. PROGRAMME| APACHE HTTP SERVER|CONFIGURE APACHE SERVER|EDIT HTTPD.CONF startet den Editor Notepad und zeigt darin die Apache-Konfigurationsdatei an.

Als ersten Test, ob Apache tatsächlich läuft, öffnen Sie in Ihrem Webbrowser die Seite **http://localhost**. Sie sollten die Defaultstartseite des Webbrowsers zu sehen bekommen. Abschließend noch zwei Verzeichnisangaben, die für eine Defaultinstallation gelten:

Konfigurationsdatei: C:\Programme\Apache Group\Apache2\conf\httpd.conf

Webdateien: C:\Programme\Apache Group\Apache2\htdocs

PHP 5.0 installieren

Auf der Website von PHP finden Sie die aktuellste Version von **PHP** in zwei Varianten: als Zip-Archiv oder als Installationsprogramm. Da das Installationsprogramm auf den Internet Information Server abgestimmt ist, in diesem Artikel als Webserver aber Apache verwendet wird, sollten Sie sich für die Zip-Variante entscheiden.

Die Installation umfasst die folgenden Schritte:

- Extrahieren Sie den Inhalt des Zip-Archivs in ein beliebiges Verzeichnis. (Für die folgenden Beispiele nehmen wir an, dass das Installationsverzeichnis C:\php5 lautet.)
- Kopieren Sie die Datei C:\php-430\php4ts.dll in das sapi-Verzeichnis Ihrer PHP-Installation (C:\php-430\sapi).
- Fügen Sie die fett markierten Zeilen in die Apache-Konfigurationsdatei httpd.conf ein. Der beste Ort innerhalb von httpd.conf befindet sich bei den anderen LoadModule-Anweisungen, von denen es normalerweise eine ganze Menge gibt.

```
# Änderungen in C:\Programme\Apache Group\Apache2\conf\httpd.conf
...
LoadModule php5_module "c:/php5/php5apache2.dll"
AddType application/x-httpd-php .php
PHPIniDir "C:/php5"
...
```

- Erstellen Sie eine Kopie von C:\php5\php.ini-recommended mit dem neuen Namen C:\php5\php.ini.
- Starten Sie Apache neu (Kommando PROGRAMME|APACHE HTTP SERVER|CONTROL APACHE SERVER|RESTART). Ein Apache-Neustart ist übrigens auch nach jeder Änderung von php.ini oder httpd.conf erforderlich!

PHP ist nun als Erweiterung zu Apache installiert und wird zusammen mit Apache gestartet. Um zu testen, ob die Installation funktioniert hat, erstellen Sie im Verzeichnis C:\Programme\Apache Group\Apache2\htdocs mit einem Editor die Datei phptest.php mit dem folgenden Inhalt:

```
<?php phpinfo();
?>
```

Anschließend öffnen Sie mit einem Webbrowser die folgenden Seite **http://localhost/phptest.php**. Das Ergebnis sollte wie in Abbildung 1 aussehen.

MySQL 4.1 oder 5.0 installieren

MySQL 4.1 installieren

Die MySQL-Installationsdatei für MySQL 4.1 finden Sie als ZIP-Archiv unter MySQL. Das Archiv enthält als einzige Datei setup.exe. Nach dessen Start erscheint ein komfortabler Installationsassistent, in dem Sie

normalerweise nur den Installationstyp (SETUP TYPE) auf TYPICAL stellen.

Nach der Installation des Servers in das Verzeichnis C:\Programme\MySQL\MySQL Server 4.1 fragt das Installationsprogramm, ob Sie einen Account bei mysql.com einrichten möchten. Diesen optionalen Schritt überspringen Sie im Regelfall einfach mit SKIP SIGN-UP.

MySQL-Konfigurationsassistent

Wesentlich wertvoller ist da schon der Konfigurationsassistent, der nach der Installation automatisch ausgeführt wird. Bei Bedarf können Sie diesen Assistenten später mit PROGRAMME|MYSQL|MYSQL SERVER 4.1|MYSQL SERVER INSTANCE CONFIG WIZARD neuerlich starten. Der Assistent erfüllt zwei Aufgaben: Er richtet die MySQL-Konfigurationsdatei ein und sichert den **MySQL-Server** durch ein Passwort ab. Die folgende Aufzählung beschreibt die wichtigsten Schritte des Assistenten, wenn Sie sich für die DETAILED CONFIGURATION entscheiden. (Sie machen nichts falsch, wenn Sie im Zweifelsfall immer den Defaultvorgaben des Assistenten folgen.)

- **SERVER TYPE**: Hier können Sie zwischen DEVELOPER, SERVER oder DEDICATED SERVER wählen. Die Option beeinflusst, welchen Teil des Hauptspeichers der MySQL-Server für sich zu reservieren versucht. Je größer der Anteil ist, desto schneller läuft der Server, aber desto stärker sind auch alle anderen Programme beeinträchtigt. Für Webentwickler ist DEVELOPER die optimale Einstellung. (Falls Sie Benchmark-Tests durchführen, erzielen Sie bei großen Datenbanken mit SERVER oder DEDICATED SERVER wesentlich bessere Ergebnisse.)
- **DATABASE USAGE**: Die drei Optionen MULTIFUNCTIONAL, TRANSACTIONAL ONLY und NON-TRANSACTIONAL ONLY entscheiden darüber, für welche Tabellentypen der MySQL-Server konfiguriert wird (für Experten: MyISAM, InnoDB oder beide). Die Defaulteinstellung lautet MULTIFUNCTIONAL, und dabei sollten Sie es auch belassen.
- **INNODB TABLESPACE SETTING**: In diesem Punkt geben Sie an, wo **InnoDB**-Datenbankdateien gespeichert werden sollen (per Default einfach im Installationsverzeichnis).
- **CONCURRENT CONNECTIONS**: Dieser Punkt bestimmt die Anzahl der Datenbankverbindungen, die üblicherweise gleichzeitig offen sein werden. Solange Sie MySQL auf einem Entwicklungsrechner laufen lassen, wird die Anzahl ziemlich niedrig sein. Wählen Sie daher die Option DECISION SUPPORT (20 Verbindungen). Wenn der MySQL-Server dagegen im Vollbetrieb in Kombination mit einer gut frequentierten Webseite läuft, wird die Anzahl der Verbindungen wesentlich höher sein (Option ONLINE TRANSACTION PROCESSING).
- **ENABLE TCP/IP NETWORKING**: Unter Windows bestehen zwei Möglichkeiten, wie der MySQL-Server mit den Anwendungen kommuniziert: über so genannte Named Pipes oder über das Netzwerkprotokoll TCP/IP. Der Assistent schlägt vor, TCP/IP zu aktivieren und die Port-Nummer 3306 zu verwenden. Auch hier sollten Sie dem Vorschlag des Assistenten folgen, wenn es nicht wirklich triftige Gründe gibt, die dagegen sprechen.
- **DEFAULT CHARACTER SET**: MySQL unterstützt alle möglichen Zeichensätze zur Speicherung von Texten in der Datenbank. In diesem Punkt legen Sie den Defaultzeichensatz fest. Dieser gilt nur, wenn beim Datenbankdesign nicht explizit ein anderer Zeichensatz gewählt wird (d.h. die Einstellung schränkt Sie später in keiner Weise ein).
- **INSTALL ALS WINDOWS SERVICE**: Der MySQL-Server kann wahlweise als *.exe-Datei oder als Windows-Dienst gestartet werden. Die zweite Variante ist komfortabler und sicherer. Behalten Sie

einfach die Defaulteinstellungen des Assistenten bei.
- **SECURITY OPTIONS**: In diesem wahrscheinlich wichtigsten Punkt des Assistenten legen Sie fest, wer eine Verbindung zum MySQL-Server herstellen kann und ob dabei ein Passwort erforderlich ist. Die sicherste und von den Autoren empfohlene Einstellung kurz zusammengefasst: Als einziger Benutzer wird root eingerichtet und durch ein Passwort abgesichert. root gilt als Datenbankadministrator und darf sich nur vom lokalen Rechner aus anmelden (nicht von einem anderen Rechner im Netzwerk). Darüber hinaus gibt es keine anonymen Accounts (Verbindungsaufbau ohne Passwort).

Selbstverständlich können Sie später weitere Benutzer einrichten, die auf MySQL bzw. auf bestimmte **MySQL-Datenbanken** zugreifen dürfen.

MySQL 5.0 installieren

Vermutlich wird die Installation von MySQL 5.0 ab Version 5.0.2 genauso verlaufen wie eine 4.1-Installation. Die zuletzt verfügbare Version 5.0.1 verwendet allerdings noch ein älteres Installationsprogramm. Die wesentlichen Unterschiede:

- Die Installation erfolgt per Default in das Verzeichnis C:\mysql.
- Um MySQL zu starten, müssen Sie das Programm C:\mysql\bin\winmysqladmin.exe starten. Unter Windows NT/2000/XP installiert dieses Programm den MySQL-Server als Windows-Dienst, der in Zukunft immer automatisch gestartet wird.
- Die MySQL-Installation wird nicht automatisch durch ein Passwort abgesichert. Sie müssen diesen Schritt selbst erledigen, z.B. mit dem MySQL Administrator oder mit phpMyAdmin, wo auch das Prinzip der **MySQL-Zugriffsrechte** beschrieben wird).

Testen, ob MySQL läuft

Um zu testen, ob die **MySQL-Installation** erfolgreich war, starten Sie den MySQL-Kommandozeileninterpreter mysql.exe. Wenn Sie das neue Installationsprogramm eingesetzt haben (also ab MySQL 4.1.5/5.0.2), starten Sie mysql.exe einfach durch PROGRAMME|MYSQL|MYSQL SERVER 4.1|MYSQL COMMAND LINE CLIENT. Wenn Sie dagegen das alte Installationsprogramm verwendet haben (MySQL 4.0.n, MySQL 4.1.4 und früher, MySQL 5.0.1 und früher), öffnen Sie mit PROGRAMME|ZUBEHÖR|EINGABEAUFFORDERUNG ein Eingabeaufforderungsfenster (DOS-Fenster) und führen darin die folgenden Kommandos aus:

```
> CD "c:\Programme\MySQL\MySQL Server 4.1\bin"
> mysql -u root -p
Enter password: ********
```

Dabei geben Sie dasselbe Passwort an, das Sie während der MySQL-Konfiguration verwendet haben. Wenn alles klappt, erscheint im Eingabefenster nun die MySQL-Eingabeaufforderung. Geben Sie als erstes Kommando `status` ein. Damit werden die wichtigsten Verbindungsparameter angezeigt. Mit `exit` beenden Sie den Kommandozeileninterpreter.

MySQL Administrator und Query Browser installieren

Grundsätzlich können Sie mit dem gerade vorgestellten Programm mysql.exe und einigen ähnlichen Werkzeugen (mysqldump.exe und mysqladmin.exe, um die beiden wichtigsten zu nennen) die gesamte Administration des MySQL-Servers durchführen. Besonders praktisch ist das aber nicht. Wesentlich mehr Komfort bieten die Programme **MySQL Administrator** und **Query Browser**.

Sie finden diese Programme unter MYSQL kostenlos zum Download. Zur Installation reicht ein Doppelklick auf die *.msi-Dateien aus. Die Programme können anschließend mit START|PROGRAMME|MYSQL|MYSQL ADMINISTRATOR bzw. |MYSQL QUERY BROWSER gestartet werden.

Ein weiteres Administrationswerkzeug ist phpMyAdmin. Sein größter Vorteil gegenüber den hier erwähnten Programmen besteht darin, dass es über einen Webbrowser bedient wird und sich daher auch zur Administration des MySQL-Servers eignet, der auf einem anderen Rechner läuft (z.B. beim ISP).

MySQL-Extensions für PHP konfigurieren

Als letzter Schritt muss nun PHP so konfiguriert werden, dass es auf den MySQL-Server zugreifen kann. Hierfür ist es entscheidend, dass PHP zwei DLL-Dateien findet, die mit PHP mitgeliefert werden. Sie müssen die folgenden Schritte ausführen:

- Kopieren Sie die Dateien C:\php5\libmysql.dll und C:\php5\libmysqli.dll in das Windows-Verzeichnis (üblicherweise C:\Windows oder C:\Winnt). Ab PHP 5.0.3 ist eine DLL, nämlich libmysql.dll, ausreichend.
- Öffnen Sie die Datei C:\php5\php.ini mit einem Editor (z.B. Notepad), und ändern Sie die im Folgenden fett hervorgehobenen Zeilen bzw. fügen Sie sie neu ein:

```
; Änderungen in c:\php5\php.ini
...
extension_dir = "c:/php5/ext"
extension = php_mysql.dll
extension = php_mysqli.dll
```

Um die richtigen Zeilen zu finden, verwenden Sie am besten BEARBEITEN|SUCHEN. Achten Sie darauf, dass als Trennzeichen für Verzeichnisse / verwendet wird, nicht wie unter Windows üblich \. Vor den Zeilen `extension=...` darf kein Strichpunkt stehen – andernfalls gilt die Zeile als Kommentar und wird ignoriert.

- Damit die Änderungen wirksam werden, müssen Sie Apache neu starten (Kommando PROGRAMME|APACHE HTTP SERVER|CONTROL APACHE SERVER|RESTART).

Sollten Probleme auftreten, laden Sie in Ihrem Webbrowser nochmals die **PHP-Testseite**. Kontrollieren Sie, ob der Dateiname in der Zeile Configuration Path (php.ini) File (fünfte Zeile) tatsächlich

C:\php5\php.ini lautet. Wenn das nicht der Fall ist, kopieren Sie Ihre php.ini-Datei an den Ort, den die Testseite angibt. (Der wahrscheinlichste Konfigurationsfehler besteht erfahrungsgemäß darin, dass PHP die Konfigurationsdatei an einem anderen Ort erwartet als Sie. Das ist dann der Grund, weswegen Ihre Änderungen in php.ini einfach ignoriert werden ...)

So lange man nur ein paar kleine Probescripts schreibt, fällt es schwer, sich vorzustellen, wie komplex – und mitunter ausufernd lang – ein Script sein kann. Wenn dann noch die nicht ungewöhnliche Situation auftritt, dass man sich Monate nach der ursprünglichen Programmierung erneut an den Programmcode setzt, um irgendetwas zu ändern oder zu verbessern, wäre man ohne erklärende **Kommentare** im Script mitunter verloren. Haben Sie hingegen mit aussagekräftigen Kommentaren gearbeitet, können Sie sehr viel einfacher rekonstruieren, was jeweils angewiesen wurde und warum Sie den Code so und nicht anders geschrieben haben.

Kommentare im **PHP-Code** werden nicht übertragen. Der Parser (der die Befehle ausführt) ignoriert die Zeile(n) einfach. Sie können Kommentare also auch getrost als „Notizzettel“, die nur für Sie selbst gedacht sind, nutzen. Es gibt zwei Methoden, eine Programmzeile im Script zu kommentieren. Entweder Sie schreiben `//` oder `#` an den Anfang der Zeile, die lediglich ein Kommentar sein soll. Das sieht beispielsweise so aus:

```
//Fehlermeldung zusammenbauen
If(!$name){$fehler="Bitte geben Sie einen Namen ein <br>";}
```

> Um eine Programmzeile direkt zu kommentieren, können Sie den Kommentar auch an das Ende der Zeile schreiben:

```
$name=""; // löscht den eingegebenen Wert
```

Benutzen Sie am Anfang des Kommentarteils das Zeichen `/*` und am Ende `*/`, wird der **PHP-Prozessor** alles ignorieren, was zwischen diesen Zeichen steht, ob nur eine Zeile oder auch mehrere:

```
<?php
/*
echo "<b>Heute ist Montag, willkommen auf unserer Seite</b>";
*/
?>
```

Würden Sie diesen **Code** speichern und die Datei im Browser aufrufen, würden Sie eine leere Seite sehen, weil ja nichts ausgegeben wird. Deswegen ersparen wir Ihnen hier ein Bild!

Info

Manche Editoren verwenden für Kommentare eine andere Farbe als die, die sonst im Script benutzt wird:

Dies ist, wie Sie sich denken können, vor allem bei langen Scripts sehr hilfreich.

Bevor wir uns genauer mit der PHP-Programmierung beschäftigen, ist es nötig, einen Blick in die zentrale Konfigurationsdatei von PHP, **php.ini**, zu werfen. Jeder PHP-Programmierer sollte zumindest mit den wichtigsten Einträgen in dieser Datei vertraut sein. Denn sie beeinflussen das Verhalten von PHP zum Teil erheblich.

Auf Linux-Systemen wird die php.ini gerne unter /etc abgelegt. Auf Windows-Systemen sollte sie sich im Windows-Verzeichnis befinden. Bearbeitet werden kann sie mit jedem Texteditor.

Die Optionen in der php.ini haben das Format Bezeichner = Wert. Alle Zeilen, die mit einem Semikolon beginnen, sind Kommentarzeilen. Wenn Sie die php.ini wie empfohlen durch Kopieren der **php.ini-dist** erzeugt haben, dann finden Sie eine reichhaltig auskommentierte Datei vor. Die Kommentare unterstützen beim Verständnis der einzelnen Konfigurationseinstellungen.

Änderungsberechtigungen

Es gibt Möglichkeiten, die Einträge in der php.ini zu überschreiben, und zwar:

- In der Konfiguration des Apache Webservers, insbesondere in der dezentralen Konfiguration für bestimmte Verzeichnisse, also etwa in `.htaccess`-Dateien. Dann gilt die dort festgelegte Einstellung im entsprechenden Verzeichnis und in seinen eventuellen Unterverzeichnissen. Außerhalb des Verzeichnisses gilt weiterhin die Einstellung der php.ini.
- In PHP-Scripts. Dann gilt die dort programmierte Einstellung für das Script, in dem sie notiert ist. Ansonsten gilt weiterhin die Einstellung der php.ini.

Für jede Option in der php.ini ist daher in PHP festgelegt,

ob und wo der zugewiesene Wert überschreiben darf. Dafür gibt es folgende vier Berechtigungsklassen:

- **PHP_INI_USER**: Optionen mit dieser Klasse dürfen durch ein PHP-Script überschrieben werden.
- **PHP_INI_PERDIR**: Optionen mit dieser Klasse dürfen durch ein PHP-Script überschrieben werden. Ferner dürfen sie durch entsprechende Einträge in der verzeichnisspezifischen Apache-Konfiguration überschrieben werden, also z.B. in einer `.htaccess`-Datei.
- **PHP_INI_SYSTEM**: Optionen mit dieser Klasse dürfen durch ein PHP-Script überschrieben werden. Ferner dürfen sie in der zentralen Apache-Konfiguration (nicht verzeichnisspezifisch) überschrieben werden.
- **PHP_INI_ALL**: Optionen mit dieser Klasse dürfen durch ein PHP-Script oder auf jede andere Art und Weise überschrieben werden.

Wenn PHP im Apache Webserver als Modul installiert ist, können Sie PHP-Optionen in der Apache-Konfiguration mithilfe der Direktive `php_value` notieren. Beispiel:

```
php_value variables_order "EPGCS"
```

Info
Diese Direktive, in einer `.htaccess`-Datei notiert, bewirkt unter anderem, dass PHP in dem entsprechenden Verzeichnis die Daten über den POST-Kanal schneller ermittelt als die über den GET-Kanal.

Für die Praxis sind die Änderungsberechtigungen nur dann von Bedeutung, wenn Sie solche Feinjustierungen in der Apache-Konfiguration vornehmen möchten. PHP-Scripts dürfen alle Optionen überschreiben. **PHP_INI_USER** ist also die Mindestberechtigung aller Optionen.

Optionen für Sprachverhalten

Diese Optionen regeln allgemeine Einstellungen zu PHP. In der php.ini finden Sie die Einstellungen unterhalb von:

;;;;;;;;;;;;;;;;;;; ; Language Options ; ;;;;;;;;;;;;;;;;;;;Nachfolgende Tabelle beschreibt einige wichtige Optionen:

Option	Default-Wert	Bedeutung	Berechtigung
`engine`	On	Nur bei On funktioniert die Einbindung von PHP unter Apache.	
`short_open_tag`	On	Wenn On, dürfen Sie PHP-Code in <? ... ?> einschließen. Wenn Off, müssen Sie <?php ... ?> notieren. Letzteres ist auf jeden Fall vorzuziehen, wenn Sie mit PHP XML verarbeiten wollen.	`PERDIR, SYSTEM`
`asp_tags`	Off	Wenn On, kann PHP so wie ASP innerhalb von <% ... %> notiert werden.	`PERDIR, SYSTEM`
`precision`	14	Anzahl der ausgegebenen Stellen bei Fließkommazahlen.	`ALL`
`y2k_compliance`	On	Jahr-2000-Kompatibilität.	`ALL`

Optionen für Ressourcenbelegung

Diese Optionen regeln, wie stark PHP das System, auf dem es läuft, belasten darf. In der php.ini finden Sie die Einstellungen unterhalb von:

;;;;;;;;;;;;;;;;;;; ; Resource Limits ; ;;;;;;;;;;;;;;;;;;;Nachfolgende Tabelle beschreibt diese Optionen:

Option	Default-Wert	Bedeutung	Berechtigung
max_execution_time	30	Wenn ein Script zur Ausführung mehr Sekunden benötigt als angegeben, wird es vom Parser abgebrochen.	
max_input_time	60	Wenn ein Script mehr Sekunden als angegeben keinen Input z.B. über GET oder POST erhält, wird es vom Parser abgebrochen.	
memory_limits	8M	Arbeitsspeicher, den ein Script maximal belegen darf. Integer-Wert mit M dahinter für Megabyte, oder -1 für „keine Begrenzung“.	ALL

Optionen für Fehlerbehandlung

Diese Optionen regeln, wie PHP bei Syntaxfehlern oder problematischem Code reagieren soll. Gerade während der Entwicklungsphase von größeren Scripts ist es sinnvoll, die Fehleranzeige und das Fehler-Logging etwas großzügiger einzustellen. In Produktivumgebungen dagegen sollten Fehler überhaupt nicht angezeigt werden. In der php.ini finden Sie die Einstellungen unterhalb von:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ; Error handling and logging ; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Nachfolgende Tabelle beschreibt die wichtigsten dieser Optionen:

Option	Default-Wert	Bedeutung	Berechtigung
error_reporting	E_ALL & ~E_NOTICE	Bestimmt, welche Vorkommnisse behandelt werden. Mögliche Werte und Beispiele siehe Kommentare in php.ini.	ALL
display_errors	On	Gibt Fehler im Browser aus. Sollte bei Produktivumgebungen auf Off gestellt werden.	ALL
display_startup_errors	Off	Gibt bei On Fehler beim Interpreter-Start aus.	ALL
log_errors	Off	Protokolliert Fehler in der Logdatei statt sie im Browser auszugeben. Empfehlenswert für Produktivumgebungen. Die Logdatei kann bei error_log festgelegt werden.	ALL
log_errors_max_len	1024	Länge für Fehlermeldungen begrenzen (Anzahl Byte). 0 bedeutet: keine Begrenzung.	ALL
ignore_repeated_errors	Off	Wenn eine PHP-Datei eine Meldung mehrmals produziert, wird sie nur einmal protokolliert.	ALL

Option	Default-Wert	Bedeutung	Berechtigung
ignore_repeated_source	Off	Wenn eine PHP-Datei eine Meldung mehrmals produziert, wird die betroffene Quelltextumgebung nur einmal protokolliert.	ALL
report_memleaks	On	Fehler bei der Speicherfreigabe werden ausgegeben bzw. protokolliert.	ALL
track_errors	Off	Die jeweils letzte Meldung steht dem Script selbst in einer Variablen namens $php_errormsg zur Verfügung.	ALL
html_errors	On	HTML-Markup in Meldungsausgaben verwenden.	ALL
error_log		Absoluter Pfadname der gewünschten Logdatei. Der Webserver muss für diese Datei Schreibrechte haben.	ALL

Optionen für Datenbehandlung

Diese Optionen regeln wichtige Voreinstellungen, wie PHP mit Daten umgeht. In der php.ini finden Sie die Einstellungen unterhalb von:

;;;;;;;;;;;;;;;;;; ; Data Handling ; ;;;;;;;;;;;;;;;;;;Nachfolgende Tabelle beschreibt die wichtigsten dieser Optionen:

Option	Default-Wert	Bedeutung	Berechtigung
track-vars	On	Wenn On, stehen GET- und POST-Daten, Umgebungsvariablen, Servervariablen und Cookies in so genannten Superglobal-Arrays zur Verfügung. Sollte unbedingt auf On stehen und die Superglobals sollten heutzutage anstelle älterer Lösungen für entsprechende Datenzugriffe verwendet werden.	
variables_order	„EGPCS“	Reihenfolge der Variablen beim Parsen: E = Umgebungsvariablen, G = GET-Daten, P = POST-Daten, C = Cookie-Daten, S = Servervariablen.	ALL
register_globals	Off	Wenn On, stehen GET- und POST-Daten, Umgebungsvariablen, Servervariablen und Cookies in bestimmten, normalen Variablen zur Verfügung. Sollte heutzutage nicht mehr verwendet werden, also auf Off stehen.	PERDIR, SYSTEM

Option	Default-Wert	Bedeutung	Berechtigung
post_max_size	8M	Integer-Wert mit M dahinter für Megabyte. Maximaler Umfang von Daten, die mittels POST-Methode an ein Script übermittelt werden können. Interessant vor allem, wenn Datei-Uploads verarbeitet werden müssen.	PERDIR, SYSTEM

Diverse Optionen

Die **php.ini** enthält zahlreiche weitere Optionen, die zum Teil das Zusammenspiel zwischen PHP und bestimmten anderen Softwareprodukten regeln. Viele davon werden in der Praxis nur sehr selten angefasst. Es gibt jedoch auch Optionen, die Sie möglicherweise ändern oder zumindest kennen sollten.

Nachfolgend eine Liste solcher Optionen:

Option	Default-Wert	Bedeutung	Berechtigung
doc_root		Nur von Bedeutung, wenn PHP im CGI-Modus läuft. In diesem Fall das Stammverzeichnis für PHP-Scripts als absolute Pfadangabe.	SYSTEM
file_uploads	On	Wenn On, werden Inhalte aus File-Upload-Feldern in HTML an PHP-Scripts übertragen, wenn Off, dann nicht.	PERDIR, SYSTEM
upload_max_filesize	2M	Integer-Wert mit M dahinter für Megabyte. Wenn file_uploads auf On steht, kann hier die maximale Größe notiert werden, die hochgeladene Dateien haben dürfen.	PERDIR, SYSTEM
allow_url_fopen	On	Wenn On, dürfen Scripts die Methoden zum Öffnen von Dateien auch auf URIs anwenden. PHP stellt dann im Hintergrund eine Socketverbindung her und lädt die URI-Inhalte wie normale Dateien.	SYSTEM
SMTP		Bei lokalen PHP-Installationen unter Windows von Bedeutung. IP-Adresse des Servers für ausgehende Mails. Wird für Mailversand aus PHP benötigt.	ALL
session.save_handler	files	Name einer Funktion, die Session-Dateien speichern kann. Sollte auf files stehen bleiben.	ALL

Option	Default-Wert	Bedeutung	Berechtigung
session.save_path	„/tmp“	Absoluter Pfadname des Verzeichnisses, unter dem Session-Dateien gespeichert werden. Es sollte sich um ein typisches Verzeichnis für temporäre Daten handeln. Muss unter MS Windows in der Regel geändert werden, da dort /tmp üblicherweise nicht existiert.	ALL
session.name	„PHPSESSID“	Default-Name für Session-Cookies.	ALL
session.use_cookies	On	Wenn On, versucht PHP für Session-Daten ein Session-Cookie an den Browser zu senden.	ALL
session.use_ only_cookies	Off	Wenn On, funktionieren Sessions nur, wenn der Anwender Session-Cookies akzeptiert. Scripts laufen dann möglicherweise nicht wie gewünscht, wenn der Anwender keine Cookies akzeptiert. Dafür wird aber dem Ausspähen von Session-Daten vorgebeugt, was auch im Interesse des Anwenders ist.	ALL
session.cache_expire	180	Anzahl Minuten, die Session-Seiten im Cache-Speicher bleiben.	ALL
session.use_trans_sid	Off	Die Session-ID wird an den GET-String angehängt, wodurch sich Sessions via URI ansprechen lassen. Sollte normalerweise auf Off stehen, da es das Ausspähen von Session-Daten erleichtert.	ALL

Für einige der genannten Optionen, vor allem diejenigen zu Sessions, sind Kenntnisse in PHP-Sessions erforderlich. Wir werden dieses wichtige Thema im vorliegenden Kapitel behandeln.

Die Funktion phpinfo()

```
<?php phpinfo(); ?>
```

Rufen Sie diese Datei via HTTP im Browser auf.

PHP ist eine **Programmier- und Script-Sprache**. Bis einschließlich Version 4.x gehört **PHP** zu den prozeduralen Programmiersprachen, in denen eine Anweisungsreihenfolge existiert, die abgearbeitet wird. Ab Version 5.0 kann PHP zusätzlich auch zu den objektorientierten Sprachen gezählt werden, die regulierbare Datenkapselung bei Objekten und Vererbung beherrscht. PHP bietet beide Formen der Programmierung an.

PHP-Code wird im üblichen Betrieb jedoch nicht in statische, für das Betriebssystem ausführbare Dateien

übersetzt, sondern von einem Interpreter zur Laufzeit „on the fly“ kompiliert und ausgeführt. Insofern gehört es zur Familie der Script-Sprachen. Für den Programmierer entfällt also das zeitraubende Neukompilieren vor jedem neuen Austesten.

> PHP entbindet den Programmierer wie alle Script-Sprachen auch davon,

sich um die Arbeitsspeicherverwaltung selbst zu kümmern. Die einzige Schnittstelle in dieser Hinsicht bietet PHP in seinen Konfigurationseinstellungen an. Dort lässt sich angeben, wie viel Speicher PHP für die Ausführung eines Scripts maximal reservieren darf.

PHP ist zwar mittlerweile auch ähnlich wie `Perl` als Script-Sprache auf Konsolenebene einsetzbar. Doch der eigentliche und konsequent verfolgte Einsatzzweck ist der als Script-Umgebung eines Webservers. PHP kann in HTML eingebettet werden und PHP-Dateien können wie HTML-Dateien in allen Webverzeichnissen abgelegt und ausgeführt werden (sofern es im Webserver als Modul eingebunden wird und nicht nur für CGI-Scripts erlaubt ist). PHP hat eine besonders enge Anbindung an die Gegebenheiten von HTML und HTTP. Viele vordefinierte Funktionen übernehmen Aufgaben, die speziell der Webprogrammierung entgegen kommen.

Geschichte und heutige Bedeutung von PHP

Die Geschichte von PHP reicht bis ins Jahr 1995 zurück. Der Aufstieg von einem kleinen Toolset zu einer ernst zu nehmenden Technologie begann 1998 auf Basis der Version 3.0. Damit wuchs allerdings auch der Druck auf Stabilität und Features der Script-Sprache.

Für die Sprachversion 4.0 wurde der Script-Interpreter von Grund auf neu entwickelt. Sein neuer Kern war die so genannte **Zend Engine** (ein Akronym aus den Vornamen der Chef-Entwickler Zeev Suraski und Andi Gutmans). Schon in der langen Beta-Phase zog der neue Script-Interpreter immer mehr interessierte Webprogrammierer in seinen Bann. Nachdem PHP 4.0 im Mai 2000 offiziell freigegeben wurde, sorgte eine für ein Open-Source-Projekt wirklich ungewöhnliche PR-Maschine dafür, dass PHP in Sachen Webprogrammierung alsbald zum Maß aller Dinge wurde und zuvor dominierende Sprachen wie `Perl` aus diesem Bereich weitgehend verdrängte.

> Auch Microsoft hat PHP bis heute keine Konkurrenz machen können.

Das `.NET-Framework`, Nachfolger der Active Server Pages (ASP), ist den meisten Webprogrammierern zu komplex und zu proprietär. Andere Konkurrenten, wie etwa die Java Server Pages (JSP), haben sich bestimmte Nischen gesichert, vor allem bei Banken und Großunternehmen. Dort wird `Java` auch für andere Zwecke verwendet und es haben immer noch Entscheider das Sagen, die einem kommerziellem Anbieter (Java wird von Sun Microsystems vertrieben) mehr vertrauen als einem Open-Source-Projekt wie PHP.
Laut php.net war PHP im Dezember 2007 auf 20,9 Mio. virtuellen Hosts (Domains) bzw. auf 1,2 Mio. echten

Hosts (IP-Adressen) mit öffentlichen Webservern installiert. Das entspricht etwa einem Drittel aller im öffentlichen Web erreichbaren Hosts. Der Anteil wächst weiter rapide. Ab der Produktversion 5.0 befriedigt PHP auch die Bedürfnisse von Großprojekten, die schon aus organisatorischen Gründen streng objektorientiert arbeiten. Davon ist ein weiterer Ausbreitungsschub zu erwarten.

Zeichenketten werden – wie Sie gesehen haben – in Anführungszeichen gesetzt. Das bringt offensichtlich Probleme mit sich, wenn ein Textstück tatsächlich in Anführungszeichen ausgegeben werden soll. Als Lösung können Sie zwei Methoden anwenden. Entweder Sie verwenden doppelte und einfache Anführungszeichen, beispielsweise doppelte Anführungszeichen für die Zeichenkette und einfache für das Textstück (oder vice versa):

```
"ich lerne 'PHP'"; oder 'ich lerne "PHP"';
```

oder Sie benutzen als ein **Maskierungszeichen** den Backslash. Dieser wird vor die Anführungszeichen für das Textstück gesetzt. Damit sagen Sie **PHP**, dass die Anführungszeichen ausgegeben, aber nicht als Beginn oder Ende einer Anweisung interpretiert werden sollen (deswegen: Maskierung):

```
"ich lerne \"PHP\"";
```

Wenn Sie mit **HTML** vertraut sind, wissen Sie, dass hier oft Anführungszeichen gesetzt werden/gesetzt werden sollen, denn Sie schreiben ja beispielsweise `<font color="red">`. Diese Anführungszeichen müssen natürlich auch maskiert werden, wenn sie Teil einer PHP-Anweisung sind. Das sieht dann so aus:

```
echo "<font color=\"red\">";
```

Wenn nun tatsächlich ein Backslash angezeigt werden soll, brauchen Sie einen zweiten:

```
echo "c:\\programme";
```

Auch für Zeilenumbrüche in PHP (nicht zu verwechseln mit dem `<br>`-HTML-Tag) gibt es ein Zeichen mit einem Backslash: `\n`.

Die folgende Tabelle gibt einen Überblick über die **Zeichen**, die Sie verwenden können, damit bestimmte Zeichen ausgegeben werden.

Zeichenfolge	Effekt
\n	neue Zeile
\r	Wagenrücklauf
\t	horizontaler Tabulator
\\	Backflash
\$	Dollarsymbol
\"	doppelte Anführungszeichen

Mit **PHP** war ursprünglich einmal „Personal Homepage“ gemeint. Das war im Jahr 1994, als PHP von Rasmus Lerdorf kreiert wurde. Über die Folgejahre wuchsen die Fähigkeiten von PHP und damit änderte sich auch die Bedeutung dieses Kürzels – passenderweise ohne dass ein Buchstabe unbedingt geändert werden musste – in die klangvolle Bezeichnung Hypertext Preprocessor.

Darin steckt schon im Ansatz die „Arbeit“ von PHP: Es verarbeitet Daten, bevor sie vom Webserver – üblicherweise in **HTML** (HyperText Markup Language) ausgegeben – zum Browser des Clients wandern.

Spezielle Merkmale von PHP

- Im Gegensatz zu diversen anderen Programmiersprachen, die clientseitig, also auf dem Rechner des Users ablaufen, führt PHP seine Anweisungen serverseitig, d.h. direkt auf dem Server aus. Die Ergebnisse der Verarbeitung werden – in der Regel – im Browser angezeigt. Wie PHP arbeitet, betrachten wir weiter hinten noch etwas genauer.
- PHP ist plattformübergreifend. Dies heißt, es ist auf allen gängigen Linux- und Windows-Versionen lauffähig und funktioniert auch mit Macintosh und OS/2. Angenehm dabei ist, dass die Scripts im Gegensatz zu den meisten anderen Programmiersprachen ohne große Mühe an eine andere Plattform angepasst werden können, sollte dies notwendig sein. Die Plattformunabhängigkeit bedeutet beim Einsatz im Internet nicht, dass Sie die Scripts für die Clients, die die **PHP-Webseiten** im Browser anschauen, je nach Betriebssystem ändern müssen, sondern dass geringe Änderungen notwendig sind, wenn auf dem Webserver das Betriebssystem gewechselt wird.
- Der PHP-Code kann direkt in **HTML-Seiten** integriert werden. Allein durch die Endung .php oder .php3, mit der die Dokumente gespeichert werden, kann der Server erkennen, dass es sich um ein PHP-Script handelt. Die offizielle PHP-Website spricht von PHP als einer „HTML embedded scripting language“. Der Begriff embedded, der ja mit eingebettet übersetzt wird, macht recht schön deutlich, wie die HTLM-Codes und PHP-Codes zusammengehen.

- Eine weitere Besonderheit von PHP ist, dass es, wie eben schon zitiert, eine Scripting Language ist und keine Programmiersprache im eigentlichen Sinn. Dies bedeutet, PHP wird nur „aktiv“, wenn ein Ereignis eingetreten ist, z.B. wenn eine Webseite (ein URL) aufgerufen wird oder wenn auf einer Webseite ein Formular von einem User ausgefüllt und dieses Formular abgeschickt wurde. Andere Programmiersprachen funktionieren im Gegensatz dazu stand-alone (Compiler), d.h., sie agieren von sich aus, teilweise das Web involvierend, teilweise nicht. JavaScript hingegen ist ebenfalls eine Scripting Language und von daher in gewisser Weise mit PHP vergleichbar, obwohl es meist nicht serverseitig, sondern clientseitig eingesetzt wird.
- Es gibt kaum Unterschiede zwischen PHP3 und PHP4, obwohl in PHP4 durchaus einige nützliche Funktionen hinzugekommen sind und PHP etwas leistungsstärker geworden ist.

Diese und weitere Informationen zu PHP finden Sie unter der Adresse php.net.

Was kann und macht PHP?

Die erste Möglichkeit, die in den Sinn kommt, wenn man über die Erstellung von Webseiten nachdenkt, ist HTML. HTML ermöglicht jedoch keine Flexibilität und/oder Dynamik auf einer Seite. Besucher einer HTML-Seite kommen weder in den Genuss, spezifische Reaktionen hervorrufen zu können, noch kann die Seite in irgendeiner Form angepasst werden. Durch diese Beschränkung ist reines HTML keine Alternative zu PHP, mit dem Sie genau das machen können – wobei Sie auch beim Einsatz von PHP nicht auf HTML verzichten können, um Ausgaben an den Browser zu senden. Mit PHP erstellen Sie dynamische Webseiten, auf denen Sie alle möglichen speziellen Gegebenheiten und Interaktionen mit dem User einbauen, auf Eingaben reagieren können, für regelmäßige Updates sorgen etc. Außerdem – und dies ist ein besonders hervorzuhebendes Feature – arbeitet PHP sehr gut mit einer Vielzahl von Datenbanken, mit dem Datei- und Verzeichnissystem und E-Mails zusammen, sodass sich Informationen speichern und weiterverarbeiten lassen (was ein enorm wichtiger Aspekt bei der Verwendung von Webseiten – für welche Zwecke auch immer – ist, denn es geht ja vielfach um die Generierung von Daten und Informationen, z.B. Adressen, Bestelldaten etc).

> Abgesehen von HTML, das auf Grund seiner Grenzen keine Konkurrenz zu PHP ist,

gibt es natürlich Alternativen, denn die Webadministratoren und Fachleute wissen ja schon längst, dass sich mit HTML keine dynamischen Seiten basteln lassen. Daher haben in den letzten Jahren auch andere serverseitige Technologien an Bedeutung gewonnen und sind mehr und mehr eingesetzt worden. Hier ist beispielsweise Perl zu erwähnen oder ASP (Active Server Pages) und (mit Einschränkungen) auch JavaScript.

Im Vergleich bietet PHP gegenüber den erwähnten Programmiersprachen einige deutliche Vorteile, die schnell auf den Punkt gebracht sind:

- PHP ist schneller zu programmieren und schneller bei der Ausführung von Scripts.
- Der zweite gewichtige Unterschied, der vor allem für den Einsteiger (in die Welt des Programmierens) von Interesse ist, ist der, dass es leichter zu lernen ist und keine formalen Programmiervorkenntnisse erfordert. Für ASP sind beispielsweise Erfahrungen mit VB-Scripts mehr

oder minder zwingend, Perl und C erfordern relativ lange Einarbeitungszeiten. Dabei handelt es sich um komplexe Sprachen, mit denen man sich nicht mal eben vertraut machen kann! Dies gilt zwar auch für PHP, aber nicht allzu komplizierte Aufgaben lernt man relativ schnell per PHP zu bewältigen; dies werden Sie feststellen, sobald Sie mithilfe dieses Kapitels Ihre ersten Gehversuche unternommen haben. Eine kleine dynamische Seite, mit der Sie beispielsweise Webseitenbesucher je nach Tageszeit oder Wochentag mit wechselnden Texten begrüßen, ist schnell erstellt. Und das ist doch schon etwas!
- Nicht zuletzt spielt es eine Rolle, dass im Prinzip nur PHP speziell für die Programmierung dynamischer Webseiten entwickelt wurde. Dies bedeutet, dass PHP bestimmte Aufgaben besonders gut und schneller lösen kann als die Alternativen, nicht alle, aber insbesondere die, für die es geschrieben wurde. Dies macht die Sprache nicht unbedingt besser als ASP oder Perl, aber in mancher Hinsicht einfach adäquater. Überdies kann man PHP mit anderen Sprachen kombinieren, was seine Funktionalität natürlich erhöht.

Funktionsweise von PHP

Wir wollen nicht in die Einzelheiten gehen! Es ist aber nicht verkehrt, eine gewisse Vorstellung davon zu haben, in welcher Form PHP arbeitet.

Voraussetzung ist das Zusammenspiel von Client, Server und PHP. PHP arbeitet – das wurde eingangs bereits erwähnt – serverseitig. Dies heißt vereinfacht: Alles, was PHP macht, geschieht auf dem Server. Wenn Sie eine PHP-Datei erstellen und auf den Server hochladen (bzw. selbst am Server arbeiten), erkennt der Server anhand der Endung, dass es sich um ein PHP-Dokument handelt und schickt es an den PHP-Interpreter. Der führt die Anweisungen aus und sendet das Ergebnis, also die passenden Informationen (HTML-Ausgaben des Scripts), zurück an den Webserver und dieser schickt es an den Browser.

Dies unterscheidet sich von HTML-generierten Seiten, wo der Client lediglich eine URL-Anfrage an den Server stellt und der Server die Anfrage direkt beantwortet, indem die Seite an den Browser des Clients geschickt wird. Bei diesem Prozess ist keinerlei serverseitige Interpretation involviert. Für den User, der eine Seite aufgerufen hat und sie im Browser betrachtet, ist im Prinzip nicht ersichtlich, ob ein **PHP-Script** ausgeführt wurde oder nicht.

Dieser Artikel geht auf die Frage ein, welchen Zeichensatz Sie für die Entwicklung Ihrer Webanwendungen verwenden sollen. Diese Frage ist relativ elementar, und sie sollte beantwortet werden, bevor Sie mit der Entwicklung eines größeren Projekts beginnen. (Eine nachträgliche Änderung des Zeichensatzes ist mühsam.)

Um das Thema hier nicht ausarten zu lassen, beschränkt sich der Abschnitt auf die beiden in unserem Sprachraum wichtigsten Zeichensätze, nämlich auf **latin1** alias ISO-8859-1 und auf Unicode **UTF-8** alias ISO-10646.

Vor den vielen Details als eine Art Kurzfassung des Abschnitts zwei Tipps:

- Wenn für Ihre Anwendung der `latin1`-Zeichensatz ausreicht (und das wird für den Großteil der im deutschen Sprachraum entwickelten Anwendungen der Fall sein), verwenden Sie diesen

Zeichensatz. Er verursacht bei weitem die geringsten Probleme.
- Wenn Sie `Unicode` einsetzen (müssen), gehen Sie dabei möglichst konsequent vor. Nutzen Sie alle zur Verfügung stehenden Unicode-Möglichkeiten sämtlicher Glieder der Entwicklungskette, also Betriebssystem, Webserver, PHP, Datenbank, Editor bzw. Entwicklungswerkzeuge etc.

Theoretisch ist es auch möglich, in unterschiedlichen Ebenen der Entwicklung bzw. Datenverwaltung unterschiedliche Zeichensätze einzusetzen. Die Wahrscheinlichkeit, dass dann an irgendeinem Punkt beim Übergang von `latin1` zu `Unicode` (oder umgekehrt) Probleme auftreten, beträgt aber nahezu 100 Prozent.

Zeichensatzgrundlagen

Zeichensätze bestimmen, welche Codes zur Darstellung von Zeichen verwendet werden. Bei den 128 US-ASCII-Zeichen sind sich die meisten **Zeichensätze** einig (z.B. Code 65 für den Buchstaben A). Problematischer ist die Darstellung internationaler Zeichen.

latin-Zeichensätze:

In der Vergangenheit wurden je nach Sprachraum verschiedene 1-Byte-Zeichensätze entwickelt, von denen die `latin`-Zeichensätze die größte Verbreitung gefunden haben: `latin1` alias ISO-8859-1 enthält alle in Westeuropa üblichen Zeichen (äöüßáàå etc.)., `latin2` alias ISO-8859-2 Zeichen aus Zentral- und Osteuropa etc. `latin0` alias latin9 alias ISO-8859-15 entspricht `latin1`, enthält aber zusätzlich das Euro-Zeichen.

Die `latin`-Zeichensätze werden sowohl von Unix/Linux als auch von aktuellen Windows-Versionen unterstützt. Bei älteren Windows-Versionen kann ersatzweise die `codepage` 1252 verwendet werden (kurz CP 1252, manchmal auch ANSI-Zeichensatz genannt), die bis auf wenige Abweichungen dem `latin1`-Zeichensatz entspricht.

Das Problem bei diesen Zeichensätzen ist offensichtlich: Ihre Anwendung kommt nie mit allen Zeichen aus ganz Europa zurecht, weil jeder `latin`-Zeichensatz nur eine Teilmenge der Zeichen enthält.

Unicode-Varianten

Als Lösung wurde der 2-Byte-Zeichensatz **Unicode** entwickelt. Mit 65.535 möglichen Zeichen deckt er nicht nur alle Zeichen ganz Europas ab, sondern darüber hinaus auch noch die der meisten asiatischen Sprachen.

Unicode regelt allerdings nur, welcher Code welchem Zeichen zugeordnet ist, nicht, wie die Codes tatsächlich gespeichert werden. Hierfür bestehen wieder mehrere Varianten, von denen UCS-2 und UTF-8 die beiden wichtigsten sind. (UTF steht für Unicode Transfer Format, UCS für Universal Character Set.)

- **UCS-2 alias UTF-16**: Die einfachste Lösung scheint auf den ersten Blick darin zu bestehen, jedes Zeichen einfach durch 2 Byte (also 16 Bit) darzustellen. Diese Formatierung wird UTF-16 oder UCS-2

genannt. Fast alle Betriebssystemfunktionen von Microsoft Windows verwenden diese Darstellung. Sie hat allerdings zwei Nachteile: Erstens verdoppelt sich der Speicherbedarf, und zwar auch in solchen Fällen, wo überwiegend europäische Zeichen oder gar nur US-ASCII-Zeichen gespeichert werden sollen. Zweitens tritt der Bytecode 0 an beliebigen Stellen in Unicode-Zeichenketten auf. Bei Texten mit US-ASCII-Zeichen ist sogar jedes 2. Byte 0. Viele C-Programme, E-Mail-Server etc. setzen aber voraus, dass das Byte 0 das Ende einer Zeichenkette markiert.

- **UTF-8**: Die bei weitem populärste Alternative zu UTF-16 ist UTF-8. Dabei werden die US-ASCII-Zeichen (7 Bit) wie bisher durch ein Byte dargestellt, deren oberstes Bit 0 ist. Alle anderen Unicode-Zeichen werden durch zwei bis vier Byte lange Byte-Ketten dargestellt.
  Der offensichtliche Nachteil dieses Formats besteht darin, dass es keinen unmittelbaren Zusammenhang zwischen der Byteanzahl und der Anzahl der Zeichen eines Dokuments gibt. Wegen der größeren Kompatibilität zu existierenden Programmen und einer Reihe anderer Vorteile hat sich UTF-8 unter Unix/Linux und bei den meisten für die Webentwicklung wichtigen Komponenten als Standard etabliert. Wenn von Unicode die Rede ist, ist in Zukunft immer Unicode im UTF-8-Format gemeint.

Trotz der offensichtlichen Vorteile von Unicode – egal, in welcher Darstellung – gibt es zwei Gründe, die gegen den sofortigen Umstieg sprechen: Zum einen ist der Unicode-Zeichensatz inkompatibel mit den bekannten 1-Byte-Zeichensätzen; vorhandene Datenbestände und Codedateien müssen also konvertiert werden. Zum anderen ist die Unicode-Unterstützung der Komponenten, die bei der Webentwicklung zum Einsatz kommen, noch alles andere als perfekt.

Zeichensatzunterstützung in Apache, PHP und MySQL

Dieser Abschnitt behandelt im Folgenden die gesamte Kette der Werkzeuge und Programme, vom Apache-Server bis zum Webbrowser auf dem Client. Der Abschnitt geht auch auf das Format HTML und das Protokoll HTTP sowie auf die **Zeichensatzkonfiguration** unter Windows und Linux ein.

HTML

Laut HTML-Standard gilt für alle Dokumente ohne explizite Zeichensatzangabe der Zeichensatz iso-8859-1 (also `latin1`). Den verwendeten Zeichensatz können Sie am Beginn von HTML-Dokumenten angeben. Im HTML-Code sieht das so aus:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
...
```

Zulässige `charset`-Einstellungen sind utf-8, iso-8859-1, iso-8859-15 etc. Das Problem besteht allerdings darin, dass viele Webbrowser diese Information schlicht ignorieren und sich stattdessen auf die HTTP-

Header-Daten verlassen (siehe unten).

HTML-Codes für Sonderzeichen

Wenn Sie möchten, dass internationale **Sonderzeichen** auch dann richtig dargestellt werden, wenn der Webbrowser die Zeichensatzinformation nicht oder falsch auswertet, können Sie für viele Sonderzeichen spezielle HTML-Codes einsetzen, beispielsweise `&auml;` statt `a`. Das macht Texte in HTML-Dokumenten allerdings schwer zu lesen. Bei Ausgaben, die Sie mit PHP durchführen, setzen Sie einfach die Funktion `htmlentities` ein. (`htmlentities('aäb')` liefert die Zeichenkette `a&auml;b`).
Info
Wenn Sie als Zeichensatz `latin1` bzw. iso-8859-1 nutzen, können Sie zwar Sonderzeichen wie äöüß direkt im HTML-Code verwenden, nicht aber das Euro-Zeichen. Um das Euro-Zeichen dennoch fehlerfrei auszugeben, verwenden Sie einfach den HTML-Code €!

Formulare

Normalerweise werden in ein HTML-Formular eingegebene Texte in dem Zeichensatz versendet, in dem die HTML-Seite dargestellt wird. Hat also beispielsweise die Erkennung des **Unicodes-Zeichensatzes** geklappt, dann sollten auch die eingegebenen Zeichen als Unicode versandt werden.

Was aber, wenn dies bei alten Browsern nicht zuverlässig funktioniert, wenn die Zeichensatzerkennung im Browser fehlgeschlagen ist oder im Browser ein bestimmter Zeichensatz fest eingestellt ist? Dann werden die Texteingaben aus dem Formular in einem anderen Zeichensatz zurückgeliefert und in der Folge falsch gespeichert.

Das folgende PDF-Dokument (die Zusammenfassung eines Vortrags über Internationalisierung und MySQL) schlägt vor, in Formulare ein `hidden`-Textfeld mit einer vorgegebenen Unicode-Zeichenkette einzubauen und vor der Formularverarbeitung zu kontrollieren, ob diese Zeichenkette korrekt übermittelt worden ist. Nach unseren Tests bleibt aber auch diese Kontrolle wirkungslos, wenn im Browser ein falscher Zeichensatz aktiv ist. In diesem Fall wird der Inhalt des `hidden`-Textfelds korrekt übertragen, die restlichen Eingabefelder enthalten aber dennoch falsch codierte Texte.

`http://mysql.binarycompass.org/Downloads/Presentations/practical-i18n-oscon2004.pdf`

HTTP-Protokoll

Das **HTTP-Protokoll** definiert, wie der Webbrowser und der Webserver miteinander kommunizieren, wie HTML-Dokumente und Formulardaten übertragen werden etc. Dieses Protokoll bietet neben dem HTML-`<meta>`-Tag die zweite Möglichkeit, den Zeichensatz einzustellen. Es wurde bereits erwähnt, dass sich die meisten Webbrowser auf die Information des HTTP-Protokolls verlassen, die Zeichensatzangabe im `<meta>`-Tag aber ignorieren.

Entscheidend für den Zeichensatz ist die Header-Information über den Dokumenttyp. Der HTTP-Header

wird vor dem eigentlichen HTML-Dokument übertragen und ist daher nicht Teil des HTML-Codes. Einige Webbrowser bieten aber ein Kommando zur Anzeige der Seiteninformationen oder -eigenschaften an, aus denen auch die Header-Informationen hervorgehen (siehe Abbildung 1).

Woher kommt nun die Header-Information über den Dokumenttyp samt Zeichensatz? Es bestehen folgende Möglichkeiten:

- Apache-Defaulteinstellung (`AddDefaultCharset utf-8` in httpd.conf)
- Apache-Verzeichniskonfiguration (`AddDefaultCharset` in .htaccess)
- PHP-Defaulteinstellung (`default_charset="utf-8"` in php.ini)
- PHP-Code der betreffenden Seite:

```
<?php Header("Content-Type: text/html; charset=utf-8");
?>
```

Beachten Sie, dass die Header-Funktion am Beginn der PHP-Seite und vor jeder HTML-Ausgabe erfolgen muss!

Details zu den Konfigurationsmöglichkeiten von Apache und PHP folgen in den weiteren Abschnitten.

Webbrowser

Wie bereits erwähnt, ist es die Aufgabe des Webbrowsers, den Zeichensatz des Dokuments zu erkennen und das Dokument entsprechend korrekt anzuzeigen. Der Webbrowser kann dazu zwei Informationen auswerten: die HTTP-Header-Daten und das HTML-`<meta>`-Tag. Die meisten Browser berücksichtigen allerdings nur die Header-Informationen und ignorieren das `<meta>`-Tag.

Über die korrekte Erkennung des Zeichensatzes hinaus bestimmen noch zwei Faktoren die korrekte Darstellung der Seite:

- Der Browser muss den Zeichensatz kennen. Das ist keineswegs immer der Fall. Sehr alte Browser haben beispielsweise keine Unicode-Unterstützung. Aber selbst moderne Browser können nicht alle Zeichensätze kennen – dafür gibt es viel zu viele. (`latin1` alias iso-8859-1 und `utf-8` bereitet natürlich keinem aktuellen Browser Probleme. Die meisten einigermaßen aktuellen Browser kommen auch mit `latin9` alias iso-8859-15 zurecht.)
- Der Browser muss Zugang zu Schriftarten haben, die die gewünschten Zeichen enthalten. Für die in Europa üblichen Zeichen ist das kein Problem. Es gibt aber nur sehr wenige Schriften, die alle in Unicode definierten Zeichen (also auch alle asiatischen Zeichen) enthalten.

Apache

Es bestehen mehrere Möglichkeiten, Apache mitzuteilen, welchen Zeichensatz er in der HTTP-Header-Information an den Browser übermitteln soll:

- **AddDefaultCharset Zeichensatz** in httpd.conf: Apache übermittelt den hier angegebenen Zeichensatz (z.B. `utf-8` oder iso-8859-1) für alle Seiten an den Browser. Die Einstellung gilt sowohl für .html- als auch für .php-Dateien. Das `<meta>`-Tag im HTML-Code wird ignoriert.
  Beachten Sie, dass einige Linux-Distributionen per Default `AddDefaultCharset utf-8` verwenden. Wenn also Ihre `latin1`-Seiten fehlerhaft angezeigt werden, ist das der wahrscheinlichste Grund.
- **AddDefaultCharset off** in httpd.conf: Apache wertet das `<meta>`-Tag aus und sendet den dort angegebenen Zeichensatz an den Browser. Es gilt also der Inhalt des `<meta>`-Tags.
- **AddCharset Zeichensatz .kennung** in httpd.conf: Damit werden ein Zeichensatz für Dateien einer bestimmten Kennung eingestellt. `AddCharset utf-8 .utf8` bewirkt also, dass für alle Dateien, deren Name auf `.utf8` endet, als Zeichensatz `utf-8` an den Browser gemeldet wird.
- **AddDefaultCharset** und **AddCharset** in .htaccess: Die beiden Schlüsselwörter sind auch in .htaccess zulässig und ermöglichen so eine verzeichnisspezifische Konfiguration. Das ist praktisch, wenn Sie (z.B. bei einem ISP) keinen Einfluss auf httpd.conf haben.
  Beachten Sie aber, dass die Einstellungen in .htaccess nur berücksichtigt werden, wenn httpd.conf für das Webverzeichnis Veränderungen durch lokale Konfigurationsdateien zulässt (`AllowOverride All` oder `FileInfo` in der betreffenden `<Directory>`-Gruppe). Vorsicht: Die Defaulteinstellung für `AllowOverride` lautet oft `None`. Die .htaccess-Datei wird in diesem Fall vollständig ignoriert.

PHP

Falls Sie einen anderen Zeichensatz als Unicode einsetzen, ist **PHP** momentan das schwächste Glied der Kette. Zwar ist PHP prinzipiell in der Lage, HTML-Dokumente in beinahe jedem Zeichensatz zu erzeugen. Funktionen wie `echo` oder `print` geben Zeichenketten einfach unverändert aus und kümmern sich dabei nicht um den Zeichensatz.

Zeichenkettenfunktionen

Das eigentliche Problem besteht aber darin, dass alle PHP-Zeichenkettenfunktionen davon ausgehen, dass die Zeichenketten den `latin1`-Zeichensatz verwenden. Wenn die Zeichenketten dagegen `UTF-8`-codiert sind,

- liefert `strlen` die Anzahl der Bytes statt der Anzahl der Zeichen,
- können Sie sich nicht auf die Ergebnisse der Kleiner- und Größer-Operatoren verlassen,
- können Sie Zeichenketten-Arrays nicht zuverlässig sortieren,
- können Sie sich auf die Ergebnisse wichtiger Funktionen wie `stripslashes` oder `htmlspecialchars` nicht mehr verlassen etc. Ganz auswegslos ist die Situation zum Glück auch nicht: PHP stellt die Funktionen `utf8_encode` und `utf8_decode` zur Verfügung, um Zeichenketten zwischen UTF-8 und ISO-8859-1 zu konvertieren. Falls PHP mit der `iconv`-Erweiterung kompiliert ist bzw. wenn diese Erweiterung als Modul aktiviert wird (unter Windows `extension=php_iconv.dll` in php.ini), stehen neben den Standardzeichenkettenfunktionen auch `iconv`-Varianten zur Verfügung. `iconv_strlen($s, "UTF-8")` ermittelt dann die tatsächliche Zeichenanzahl einer UTF-8-Zeichenkette. Noch mehr Zusatzfunktionen stellt die `mbstring`-Erweiterung zur Verfügung. Damit können Sie auch reguläre Ausdrücke auf Multibyte-Zeichenketten anwenden, E-Mails versenden etc.

Zeichensatz der PHP-Dateien

Die PHP-Dateien müssen in einem Zeichensatz vorliegen, der es dem **PHP-Interpreter** ermöglicht, die PHP-Sprachelemente korrekt zu erkennen. Das setzt voraus, dass zumindest die US-ASCII-Zeichen wie bei `latin1` codiert sind. Diese Voraussetzung ist für `UTF-8`-Dateien zum Glück erfüllt.

HTTP-Zeichensatz-Header in php.ini einstellen

Per Default gibt der PHP-Interpreter keine HTTP-Header-Informationen zum Zeichensatz an Apache weiter und überlässt es somit dem Webserver, ob und welche Header-Zeichensatzdaten dieser an den Browser sendet.

Durch die Einstellung `default_charset="utf-8"` in php.ini können Sie dieses Verhalten ändern. Der PHP-Interpreter gibt nun den angegebenen Zeichensatz an Apache weiter. Dieser verändert die Informationen nicht mehr. Das heißt, die von PHP stammenden Informationen haben Vorrang gegenüber denen aus der Apache-Konfiguration!

HTTP-Zeichensatz-Header dynamisch einstellen

Sie können den gewünschten Header-Zeichensatz auch durch die PHP-Funktion `header` erzeugen. Das ermöglicht eine dokumentenspezifische Zeichensatzeinstellung. Die `header`-Funktion muss am Beginn der PHP-Seite aufgerufen werden (vor jeder HTML-Ausgabe!).

```
<?php header("Content-Type: text/html; charset=utf-8");
?>
```

Die durch die `header`-Funktion gesendeten Daten haben Vorrang sowohl gegenüber der `default_charset`-Einstellung als auch gegenüber der Apache-Konfiguration.

MySQL

Geradezu vorbildlich ist die Zeichensatzunterstützung von **MySQL** – allerdings erst seit Version 4.1. Sie können nicht nur für den Server oder für eine Datenbank den gewünschten Zeichensatz einstellen, sondern auch für jede Tabelle und sogar für jede einzelne Spalte einer Tabelle. Es ist also möglich, in einer Tabelle eine Spalte mit latin1-Zeichenketten und eine zweite Spalte mit `utf-8`-Zeichenketten zu erzeugen.

MySQL unterstützt eine ganze Reihe verschiedener Zeichensätze, unter anderem ascii, binary, cp1250, greek, hebrew, latin1, latin2, latin5, latin7, ucs2 und utf8. In der Liste fehlt allerdings latin9. Das Euro-Zeichen können Sie daher nur in utf8- oder ucs2-Spalten speichern.

`utf-8`-Spalten weisen gegenüber `latin1`-Spalten momentan eine wesentliche Einschränkung auf: Die deutsche Sortierordnung wird nicht unterstützt. Die Sortierordnung `utf8_general_ci` ist zwar für viele

deutschsprachige Anwendungen ebenfalls ausreichend, aber wenn Sie gemäß dem DIN-1- oder DIN-2-Standard sortieren möchten, müssen Sie bis auf weiteres bei `latin1`-Spalten bleiben.

Konfiguration des Defaultzeichensatzes

In der MySQL-Konfigurationsdatei my.cnf bzw. my.ini (Windows) können Sie mit `default-character-set = latin1` den Defaultzeichensatz für den MySQL-Server einstellen (Optionsgruppe `[mysqld]`). Die Einstellung gilt für neue Datenbanken, Tabellen und Textspalten, soweit beim Erzeugen dieser Objekte nicht explizit ein anderer Zeichensatz angegeben wird.

Zeichensatztransformation durch die Client-Bibliotheken

Was nützt es, wenn der MySQL-Server selbst mit allen erdenklichen Zeichensätzen umgehen kann, der Client (also z.B. Ihr PHP-Programm oder der Kommandozeileninterpreter `mysql`) aber einen ganz bestimmten Zeichensatz voraussetzt? Die MySQL-Client-Bibliotheken wandeln daher automatisch alle **Zeichenketten** vom Zeichensatz des Clients in den Zeichensatz des Servers um (und umgekehrt).

Für welche Aspekte der Verbindung zwischen Client und Server welche Zeichensätze zum Einsatz kommen, wird durch mehrere MySQL-Systemvariablen gesteuert, die in der folgenden Tabelle zusammengefasst sind.

Variable	Bedeutung
`@@character_set_client`	Zeichensatz des Clients
`@@character_set_server`	Defaultzeichensatz des Servers
`@@character_set_connection`	Zeichensatz für die Verbindung zwischen Client und Server
`@@character_set_results`	Gewünschter Zeichensatz für SELECT-Ergebnisse
`@@character_set_database`	Defaultzeichensatz der Datenbank

Einige dieser Variablen können Sie in Ihrem Client-Programm verändern. Wenn Sie also beispielsweise **PHP-Code** schreiben, der UTF-8-Daten verarbeiten soll, sehen die entsprechenden Kommandos so aus (hier für die `mysqli`-Schnittstelle):

```
// mysql-Verbindung herstellen
$mysqli->query("SET @@session.character_set_client = 'utf8'");
$mysqli->query("SET @@session.character_set_results = 'utf8'");
```

```
$mysqli->query("SET @@character_set_connection = 'utf8'");
```

Nach diesen Vorarbeiten können Sie SQL-Kommandos mit UTF-8-Zeichenketten ausführen und erhalten SELECT-Ergebnisse ebenfalls im UTF-8-Format.

MySQL-Kommandos (`mysql`, `mysqldump`)

Die Kommandos `mysql` und `mysqldump` versuchen selbst zu erkennen, in welchem Zeichensatz Ein- und Ausgaben erfolgen sollen. Das gelingt manchmal, aber nicht immer. Manchmal müssen Sie daher mit der Option `--default-character-set=latin1` oder `=utf8` nachhelfen.

MySQL-Administrationswerkzeuge (phpMyAdmin, Query Browser)

phpMyAdmin

Dieses Programm kommt verblüffend gut mit allen erdenklichen Zeichensätzen zurecht. Das Beispiel ist gewissermaßen der Beweis dafür, dass man mit PHP durchaus Unicode-taugliche Programme entwicklen kann. Zwei Tipps zur Bedienung:

- Auf der Startseite von phpMyAdmin können Sie den Zeichensatz für die MySQL-Verbindung einstellen. Wählen Sie hier `utf8_general_ci`!
- Wenn Sie mit phpMyAdmin *.sql-Dateien importieren bzw. ausführen, müssen Sie den Zeichensatz dieser Dateien angeben. (Per Default nimmt phpMyAdmin an, dass es sich um UTF-8-Dateien handelt.)

MySQL Query Browser

Erwartungsgemäß zeigt auch dieses Programm wenig Zeichensatzprobleme. Bei der aktuellen Linux-Funktion treten allerdings manchmal Schwierigkeiten bei der korrekten Interpretation von Eingaben auf. Beispielsweise hatte ich mehrfach Probleme, ein Euro-Zeichen in eine Tabelle mit Unicode-Spalten einzugeben.

Linux

Unter Linux steuern die Umgebungsvariablen LANG sowie LC_xxx, welcher Zeichensatz per Default gilt. Die meisten Programme und insbesondere alle Textkommandos halten sich an diese Einstellung. Einige aktuelle Distributionen verwenden mittlerweile UTF-8 als Defaultzeichensatz (z.B. Fedora, Red Hat, SUSE, nicht aber Mandrakelinux)!

Wo diese Variablen eingestellt werden, hängt von der Distribution ab (bei Red Hat und Fedora /etc/sysconfig/i18n, bei SUSE /etc/sysconfig/language). Manche Distributionen stellen auch komfortable Konfigurationswerkzeuge zur Verfügung, um den Zeichensatz zu verändern (bei SUSE das YaST-Modul SYSTEM|SPRACHE).
Info

Der Defaultzeichensatz gilt auch für den Editor, mit dem Sie Ihre PHP-Dateien erstellen. Wenn Sie, wie in diesem Artikel empfohlen, `latin1` verwenden, sollten Sie auch den Linux-Defaultzeichensatz entsprechend einstellen. Andernfalls müssen Sie beim Aufruf des Editors immer darauf achten, dass alle *.php-Dateien im richtigen Zeichensatz gespeichert werden, was ebenso mühsam wie fehleranfällig ist.

Wenn Sie den Zeichensatz von Text- oder Codedateien nachträglich ändern möchten, helfen Ihnen dabei die Kommandos `recode` oder `iconv`. Die beiden folgenden Zeilen zeigen zwei alternative Möglichkeiten, um `latin1`-Dateien in `utf8`-Dateien umzuwandeln:

```
user$ recode latin1..u8 < latin1dat > utf8dat
user$ iconv -f latin1 -t utf-8 latin1dat >> utf8dat
```

Microsoft Windows

Unter Windows gibt es keine zentrale Einstellung für den Defaultzeichensatz. Das hat damit zu tun, dass reine Textdateien unter Windows eine viel geringere Rolle spielen als unter Unix/Linux. Sie müssen aber bei Ihrem Editor darauf achten, dass dieser die *.php-Dateien im richtigen Zeichensatz speichert.

PHP-Scripting der Anfang

Bei der Behandlung von Fehlern kommt das Konzept des **Exceptionhandlings** (Ausnahmebehandlung) zum Einsatz. Es bietet die Möglichkeit, bestimmte Fehler abzufangen. Damit wird es einem Programm ermöglicht, weiterzulaufen, obwohl ein Fehler aufgetreten ist. Sie werden lediglich informiert, so dass Sie die Fehlerursache abstellen können.

Beim Exceptionhandling wird der Codebereich, in dem ein Fehler auftreten kann, in einen sogenannten `try`-Block eingeschlossen. Es wird „versucht“, den Code auszuführen. Falls ein definierter Fehler auftritt, wird ein Objekt der Klasse Exception durch die Anweisung `throw` erzeugt.

Anschließend wird statt des restlichen Codes im `try`-Block der Code im zugehörigen `catch`-Block ausgeführt; der Fehler wird somit „abgefangen“. Dies erläutere ich am Beispiel eines Programms, zu dem es zwei Versionen gibt: einmal ohne und einmal mit Ausnahmebehandlung.

Ohne Ausnahmebehandlung

Zunächst das Programm ohne Ausnahmebehandlung:

```
<html>
<body>
<?php
/* Für dieses Programm alle Fehler anzeigen */
ini_set("error_reporting", 32767);
/* Variable unbekannt */
```

```
$c = 2 * $a;
echo "<p>$c</p>";
/* Division durch 0 */
$x = 24;
for($y=4; $y>-3; $y--)
{
$z = $x / $y;
echo "$x / $y = $z<br />";
}
/* Zugriff auf Funktion */
fkt();
echo "Ende"
?>
</body>
</html>
```

Datei exception_ohne.php
Info
Zunächst wird mit Hilfe der Funktion `ini_set()` der **PHP-Konfigurationsparameter** `error_reporting` eingestellt. Dieser bestimmt, welche Fehler angezeigt werden und welche nicht. Der Wert 32767 führt dazu, dass der weitaus größte Teil der Fehler angezeigt wird.

Anschließend wird eine Variable innerhalb eines Ausdrucks verwendet, die bis zu diesem Zeitpunkt noch unbekannt ist. Darüber werden Sie mit Hilfe einer Notice informiert, siehe Abbildung 1.

- Nachfolgend wird die Zahl 24 nacheinander durch die Zahlen 4, 3, 2, 1, 0, -1, -2 geteilt. Bei der Division tritt ein Fehler auf, wenn durch 0 geteilt wird. Darüber werden Sie mit Hilfe eines Warnings informiert, siehe wiederum Abbildung 1.
- Als Letztes wird eine unbekannte Funktion aufgerufen. Dies führt zu einem Fatal Error und dem Abbruch des Programms. Daher ist die letzte Ausgabe mit dem Text „Ende“ nicht mehr zu sehen, siehe auch in Abbildung 1.

Mit Ausnahmebehandlung

Es folgt das gleiche Programm, diesmal aber mit Ausnahmebehandlung:

```
<html>
<body>
<?php
/* Für dieses Programm alle Fehler anzeigen */
ini_set("error_reporting", 32767);
/* Variable unbekannt */
try
{
```

```
if(!isset($a))
throw new Exception("Variable unbekannt");
$c = 2 * $a;
echo "<p>$c</p>";
}
catch(Exception $e)
{
echo $e->getMessage() . "<br />";
}
/* Division durch 0 */
$x = 24;
for($y=4; $y>-3; $y--)
{
try {
if($y == 0)
throw new Exception("Division durch 0");
$z = $x / $y;
echo "$x / $y = $z<br />";
}
catch(Exception $e)
{
echo $e->getMessage() . "<br />";
}
}
/* Zugriff auf Funktion */
try
{
if(!function_exists("fkt"))
throw new Exception("Funktion unbekannt");
fkt();
}
catch(Exception $e)
{
echo $e->getMessage() . "<br />";
}
echo "Ende"
?>
</body>
</html>
```

Datei exception_mit.php
Info
Vor der Nutzung der unbekannten Variablen wird mit Hilfe der Funktion `isset()` geprüft, ob die **Variable** existiert. Das Ganze findet in einem `try`-Block statt. Es ist also ein „Versuch“, die Variable zu nutzen.

Falls festgestellt wird, dass die Variable nicht existiert, wird mit Hilfe von throw eine Exception (Ausnahme) erzeugt und „geworfen“, mit einem zugehörigen Text.

Diese Exception wird in dem `catch`-Block „gefangen“, der dem `try`-Block zugeordnet ist. Bei der Erzeugung der **Exception** wurde ihr eine Meldung mitgegeben. Diese Fehlermeldung wird mit Hilfe der Methode `getMessage()` ausgegeben, siehe Abbildung 2.

- Der Rest des `try`-Blocks wird nicht mehr bearbeitet.
- Das Gleiche findet in diesem Programm noch zweimal statt:
- Zunächst wird geprüft, ob durch 0 geteilt werden soll. Ist dies der Fall, so wird eine entsprechende Meldung ausgegeben (siehe wiederum Abbildung 2) und mit dem nächsten Schleifendurchlauf fortgefahren.
- Anschließend wird mit Hilfe der Funktion `function_exists()` `function_exists()` geprüft, ob die Funktion existiert. Ist dies nicht der Fall, so wird eine entsprechende Meldung ausgegeben, siehe auch Abbildung 2.
- Das Programm läuft bis zum „Ende“.

Bislang haben wir nur einzelne PHP-Scripts kennen gelernt, die auch zugleich via URI im Browser aufrufbar waren. Bei größeren Projekten wird der Code jedoch in aller Regel auf mehrere, manchmal sogar sehr viele Scripts verteilt. PHP-Scripts können andere PHP-Scripts einbinden. Dabei müssen nicht alle Scripts innerhalb der Document Root liegen. Nur diejenigen, die im Browser direkt aufrubar sein sollen, müssen sich in einem Webverzeichnis befinden.

Die einzelnen Scripts übernehmen dabei typischerweise abgegrenzte Aufgaben. Ein Script etwa übernimmt die Kommunikation mit der Datenbank, ein zweites erledigt das Verarbeiten von Formulardaten, ein drittes das Behandeln aufgetretener Fehler und ein viertes das Zusammenfügen von HTML-Templates. Ein Script fungiert meistens als „Ober-Script“, das alles koordiniert.

Ein kleines Beispiel soll demonstrieren,

wie sich Code aufgabenspezifisch auf mehrere Scripts verteilen lässt und welche Vorteile das bringt. Wir setzen dabei auf dem Script aus php-mit-einsatz-von-html-templates auf und verfolgen das Template-Konzept in etwas veränderter Form weiter. Insgesamt wird unser Projekt jetzt aus folgenden PHP-Scripts bestehen:

- **linkweb_vars.php** definiert globale Variablen für das gesamte Script-Ensemble.
- **linkweb_templates.php** ist für das Einbinden von Templates zuständig.
- **linkweb_errors.php** übernimmt die Ausgabe von Fehlermeldungen.
- **linkweb.php** ist das koordinierende Script, das auch im Browser aufgerufen wird.

Dazu kommen noch zwei Template-Dateien:

- **linkweb.tpl** ist das Template für normale Webseiten des Projekts.

- **linkweb_errors.tpl** ist das Template für Fehlermeldungsseiten.

Außerdem benötigen wir natürlich Datendateien. Dazu nehmen wir wieder die Dateien <0001.txt bis <0015.txt wie im Beispiel aus php-mit-einsatz-von-html-templates.

PHP-Script linkweb_vars.php: die Variablenzentrale

```
<?php
#-----------------------------------------
# Templates:
$templates = array();
$templates['standard'] = "linkweb.tpl";
$templates['error'] = "linkweb_errors.tpl";
#-----------------------------------------
# Inhaltsdateien:
$content_files = array();
$content_files['home'] = "0001.txt";
$content_files['impressum'] = "0007.txt";
$content_files['themen'] = "0003.txt";
$content_files['branchen'] = "0009.txt";
$content_files['auskunft'] = "0005.txt";
$content_files['geo'] = "0002.txt";
$content_files['wissen'] = "0004.txt";
$content_files['literatur'] = "0008.txt";
$content_files['lexika'] = "0006.txt";
$content_files['glossare'] = "0010.txt";
$content_files['wikis'] = "0014.txt";
$content_files['ftp'] = "0011.txt";
$content_files['wais'] = "0015.txt";
$content_files['newsgroups'] = "0012.txt";
$content_files['foren'] = "0013.txt";
#-----------------------------------------
# Fehler:
$errors = array();
$errors['no_page'] = "Fehler: Seitenname existiert nicht";
$errors['no_template_key'] = "Interner Fehler: Template-Schlüssel existiert
nicht";
$errors['no_error_key'] = "Interner Fehler: Fehler-Schlüssel existiert nicht";
#-----------------------------------------
# Diverses:
$standard_title = "untitled";
?>
```

Info
Wie ersichtlich, werden in diesem Script ausschließlich Variablen definiert. Der Vorteil dieser Lösung ist, dass man bei mehreren Scripts nicht mehr überlegen muss, welche Variable in welchem Script eingeführt

wird. Wenn beispielsweise neue Seiten hinzukommen, neue Templates, neue Fehlermeldungen usw., dann wird alles in dieser Script-Datei notiert.

PHP-Script linkweb_templates.php: Zusammenkleben von Templates und Daten

```
<?php
#----------------------------------------
# Includes:
include_once("linkweb_vars.php");
include_once("linkweb_errors.php");
#----------------------------------------
# Funktionen:
function template_use($template_key){
global $templates;
if(! array_key_exists($template_key, $templates))
error_show("no_template_key");
else
return(file_get_contents($templates[$template_key]));
}
function template_set_var($template_content, $name, $value){
$pattern = "/\[\%".$name."\%\]/";
return(preg_replace($pattern, $value, $template_content));
}
function template_get_title($template_content){
global $standard_title;
$pattern = "/<h1>(.*)<\/h1>/";
preg_match($pattern, $template_content, $matches);
if(isset($matches[1]))
return($matches[1]);
else
return($standard_title);
}
function template_show($template_content){
echo $template_content;
exit();
}
?>
```

Info
In diesem Modul-Script begegnet uns erstmals die PHP-Funktion `include_once()`. Sie erlaubt das Einbinden anderer Dateien an der aktuellen Stelle im Code. Eingebunden werden die anderen Scripts linkweb_vars.php und linkweb_errors.php. Der Grund ist, dass das Script zum Template-Management auf Variablen zugreifen muss, die in linkweb_vars.php definiert wurden, und dass bei auftretenden Fehlern eine Funktion aus linkweb_errrors.php aufgerufen werden soll.

Neben `include_once()` gibt es auch die ältere Funktion `include()`. Wir ziehen `include_once()` jedoch vor, um Probleme mit Mehrfacheinbindungen zu vermeiden. Darauf kommen wir später noch zurück.

> Es ist nicht zwingend nötig, die anderen Dateien

wie in unserem Beispiel am Beginn eines Scripts einzubinden. Wichtig ist nur, dass ein anderes PHP-Script eingebunden wird, bevor auf Variablen oder Funktionen dieses Scripts zugegriffen wird. Im Sinne eines übersichtlichen Codes ist es jedoch zweckmäßig, die Inkludierungen am Beginn zu notieren, damit die Abhängigkeiten des Scripts transparent werden.

Das Script linkweb_templates.php besteht außer den `include_once`-Anweisungen am Beginn nur aus einer Reihe von Funktionen. Auch das ist typisch für modular aufgebaute Scripts: Der Code in den Modulen wird nicht direkt beim Einbinden ausgeführt, sondern steht in Routinen, die bei Bedarf aufgerufen werden. Folgende Funktionen sind in linkweb_templates.php vorhanden:

- **template_use()** lädt die Template-Datei eines gewünschten Templates und gibt deren kompletten Inhalt zurück. Ein Script, das ein Template laden möchte, muss sich darum also nicht selber kümmern, sondern ruft einfach diese Funktion mit dem Namen des gewünschten Templates als Parameter auf. Die Funktion ruft, falls kein Template mit dem übergebenen Namen existiert, die Funktion `error_show()` auf, die für eine ordentliche Fehlerausgabe sorgt. Diese Funktion befindet sich in dem eingebundenen Script `linkweb_errors.php`.
- **template_set_var()** ersetzt in HTML-Code selbst definierte Variablen der Notationsform `[%Variablenname%]` durch einen gewünschten Wert. Hier wurde diese „Leistung“ jedoch in eine dafür zuständige Funktion verbannt. Ein Script, das diese Leistung wünscht, ruft einfach die Funktion auf.
- **template_get_title()** sucht in HTML-Code nach der ersten `h1`-Überschrift und gibt deren Elementinhalt zurück. Der Rückgabewert lässt sich als Seitentitel interpretieren.
- **template_show()** gibt Inhalte aus. Gedacht ist die Funktion zur Ausgabe fertig zusammengeklebter Seiteninhalte. Deshalb enthält sie am Ende auch einen Aufruf von `exit()`, denn nach der Ausgabe an den Browser hat das Script seine Arbeit beendet.

Es fällt auf, dass alle Funktionsnamen mit template_ beginnen. Konventionen dieser Art sind nirgendwo vorgeschrieben. Sie sind einfach Teil einer ordentlichen Programmierung.

PHP-Script linkweb_errors.php: Fehlerausgaben

```
<?php
#-----------------------------------------
# Includes:
include_once("linkweb_vars.php");
include_once("linkweb_templates.php");
#-----------------------------------------
```

```
# Funktionen:
function error_show($error_key){
global $templates, $errors;
$page = template_use("error");
if(array_key_exists($error_key, $errors))
$page = template_set_var($page, "message",
$errors[$error_key]);
else
$page = template_set_var($page, "message",
$errors['no_error_key']);
template_show($page);
}
?>
```

Info
In diesem Script werden die zentralen Variablen und die Template-Verwaltung eingebunden. Auch dieses Script enthält ansonsten nur funktionsgebundenen Code, und zwar in einer einzigen Funktion namens `error_show()`. Diese Funktion ruft die in linkweb_templates.php stehende Funktion `template_use()` mit dem Template-Namen error auf. Dadurch wird `template_use()` veranlasst, die Template-Datei zu laden, deren Pfadname sie über den in linkweb_vars-php notierten Array `$templates` mit dem Schlüsselnamen error ermittelt. Das alles mag etwas verwirrend sein am Anfang, doch es ist wichtig, diese Zusammenhänge zu verstehen und zurückzuverfolgen. Denn das Verständnis dieser Querbezüge zwischen den Script-Modulen ist der Schlüssel für eine eigene erfolgreiche Programmierung im Stil der Profis.

> Der Rückgabewert von `template_use()` ist der komplette Inhalt einer Template-Datei.

Diese wird innerhalb von `error_show()` in der funktionslokalen Variablen `$page` gespeichert. Als nächstes muss die Variable `[%message%]` in der Template-Datei für Fehlermeldungen durch den gewünschten Fehlermeldungstext ersetzt werden. Dazu wird wieder eine Funktion aus linkweb_templates.php aufgerufen, nämlich `template_set_var()`. Die Funktion wird mit geeigneten Parametern versorgt. Da `show_error()` nicht sicher sein kann, ob der ihr übergebene Parameter, den sie als Schlüsselname einer Fehlermeldung im Array `$errors` interpretiert, dort tatsächlich vorkommt, wird eine `if`-Abfrage eingebaut, die das überprüft. Sollte keine Fehlermeldung zum übergebenen Wert existieren, wird eine Standardfehlermeldung ausgegeben.

Am Ende wird `template_show()` aufgerufen, wodurch die Fehlerseite an den Browser ausgegeben wird.

PHP-Script linkweb.php: das Hauptprogramm

```
<?php
#-----------------------------------------
# Includes:
include_once("linkweb_templates.php");
```

```
include_once("linkweb_errors.php");
#----------------------------------------
# Inhalt seitenabhängig einlesen:
if(isset($_GET['page']))
$get_page = $_GET['page'];
else
error_show('no_page');
$page = template_use("standard");
$page = template_set_var($page, "content",
file_get_contents($content_files[$get_page]));
$title = template_get_title($page);
$page = template_set_var($page, "title", $title);
template_show($page);
?>
```

Info
Dies ist das einzige Script, das im Browser aufgerufen wird. Auch hier werden zu Beginn die im weiteren Code benötigten Modul-Scripts eingebunden. Der Inhalt des „Hauptprogramms“ steht im Gegensatz zu denen der Modul-Scripts nicht in Funktionen, sondern wird direkt ausgeführt. Die Aufgabe des Hauptprogramms besteht darin, eine normale Seite aus Templates, Datendateien usw. zusammenzukleben und auszugeben. Dazu muss der übergebene GET-Parameter `page` ausgewertet werden. Wurde kein solcher GET-Parameter beim URI-Aufruf des Scripts übergeben, wird `error_show()` aufgerufen, um einen entsprechenden Fehler auszugeben. Ansonsten wird ähnlich wie innerhalb von `error_show()` ein Template-Inhalt in einer Variablen `$page` gespeichert. Der Seiteninhalt aus der passenden Datendatei wird durch Aufruf von `template_set_var()` gesetzt. Ersetzt wird die Zeichenfolge `[%content%]` im Template-Code. Der Wert, durch den sie ersetzt wird, wird durch Aufruf der PHP-Funktion `file_get_contents()` ermittelt. Der Rückgabewert dieser Funktion ist nämlich der komplette Inhalt der eingelesenen Datei. Welche Datei eingelesen werden soll, entscheidet das Script, indem es in dem in linkweb_vars.php gespeicherten Array `$content_files` den Schlüssel mit dem im GET-String übergebenen Namen verwendet.

> Der Titel wird noch ermittelt und in das `title`-Element als Elementinhalt eingesetzt.

Am Ende wird die fertige Seite durch Aufruf von `template_show()` ausgegeben.

Einbindungen kreuz und quer

In unserem Beispiel ergeben sich folgende Abhängigkeiten:

- **linkweb.php** bindet linkweb_template.php und linkweb_errors.php ein.
- **linkweb_template.php** bindet linkweb_vars.php und linkweb_errors.php ein.
- **linkweb_errors.php** bindet linkweb_vars.php und linkweb_templates.php ein.

Durch das muntere Einbinden entsteht die Situation, dass manche Dateien gleich mehrfach eingebunden werden, und sogar gegenseitige Einbindungen kommen vor. In vielen Programmiersprachen führt dies zu Problemen, da deren Compiler das Einbinden gnadenlos so durchziehen wie angewiesen. Dadurch kommen dann Funktionen oder Variablen aus mehrfach eingebundenen Modulen im Gesamtquelltext doppelt oder mehrfach vor, was den Compiler zu Fehlermeldungen veranlasst und dazu, das Kompilieren zu verweigern. Das Ergebnis sind die zahlreichen **ifdefs** etwa in C, um die sich der Programmierer selber kümmern muss.

In PHP sorgt die im Beispiel überall verwendete Funktion `include_once()` dafür, dieses Problem zu vermeiden. Wurde ein Modul bereits eingebunden, bindet es der Compiler nicht noch einmal ein.

Fazit: Modularisierung von Code lohnt sich

Vielleicht ist Ihnen aufgefallen, dass die PHP-Quelltexte in den modulartig verteilten Scripts dieses Abschnitts deutlich abstrakter wirken als diejenigen aus den ersten Beispielen. Das liegt daran, dass fast alle Aufgaben nicht mehr „selbst“ erledigt werden, sondern durch Aufruf entsprechender Funktionen. Das einheitliche Namenskonzept bei Variablennamen und Funktionsnamen tut ein übriges. Der Code wird dadurch andererseits eleganter und, was noch viel wichtiger ist, besser wartbar.

Mit diesem Modularisierungsbeispiel haben wir uns dem Programmierstil der professionellen PHP-Entwickler bereits angenähert. Was noch fehlt, ist die Verwendung von Objekten, Klassen und Instanzen. Darauf gehen wie später noch genauer ein.

Sie sind beim Surfen sicherlich schon oft darauf gestoßen: in irgendeiner Ecke der Webseite befindet sich ein kleiner Zähler, der anzeigt, der wievielte Besucher der Webseite Sie sind. Ist es die eigene Seite, ist es natürlich interessant zu wissen, wie groß die Heerschar der Besucher ist, die Ihre Seite aufruft.

- Wenn Sie keine großen Ansprüche haben, ist solch ein Zähler mit **PHP** relativ rasch programmiert. Im Folgenden entwickeln wir ein Script, mit dem Sie einen einfachen Counter ohne Cookie und mit Textausgabe erzeugen.
- Aufwändiger ist es, einen Counter zu programmieren, der einen Cookie setzt, um eine sogenannte Reloadsperre zu gewährleisten und den Zählerstand außerdem mit Hilfe von kleinen Grafiken ausgibt. Dieser Aufgabe widmen wir uns im zweiten Script.
- Im dritten Script wird der Counter dahingehend erweitert, dass er immer eine feste Anzahl an Ziffern anzeigt, egal wie groß die Besucherzahl ist. Es werden also führende Nullen hinzugefügt.

Ein einfacher Counter ohne Cookie

Bei einem einfachen **Counter** geht es lediglich darum, eine Textdatei zu erzeugen, die den Counterwert speichert, den Wert beim Aufrufen der Seite auszulesen und anschließend den neuen Wert (+1 für den neuen Besucher) zu speichern und anzuzeigen.

Da Sie die Funktionen zum Öffnen einer Textdatei bereits kennen gelernt haben, haben Sie den Programm-Code schnell geschrieben. Ins Spiel kommt eigentlich nur eine neue Funktion des Dateiensystems, die Funktion `rewind()`.

Diese Funktion setzt den Dateizeiger auf das erste Byte einer Datei zurück. Vor dem Einsatz von `rewind` muss zuvor eine Datei erfolgreich geöffnet worden sein, Sie müssen also bereits die Funktion `fopen` eingesetzt haben.

Info

```
int rewind(int file)
```

Die Funktion setzt die Position des Dateizeigers auf den Anfang der Datei zurück.

Das hier vorgestellte Beispiel kann nur einen Counterwert in der Textdatei speichern. Möchten Sie verschiedene Counter für mehrere Seiten erstellen, können Sie dies einfach bewerkstelligen, indem Sie den Namen der Textdatei in jeder Webseite variieren. Beachten Sie bitte, dass die Textdatei des Counters im gleichen Verzeichnis wie die Seite liegen oder aber die Pfandangabe angepasst werden muss. Sie können relative Pfadangaben, wie Sie sie von Hyperlinks kennen, verwenden.

Der folgende Code zeigt den Programm-Code für einen einfachen Counter. Er ist – wie Sie sehen – tatsächlich sehr kurz!

```
<?php
if(!file_exists("count.txt")){fopen("count.txt", "a" );}
$counter=fopen("count.txt","r+"); $aufruf=fgets($counter,100);
$aufruf=$aufruf+1;
rewind($counter);
fputs($counter,$aufruf);
echo $aufruf;
?>
```

Erläuterung des Scripts

Nach dem öffnenden PHP-Tag wird die Datei angelegt:

```
if(!file_exists("count.txt")){fopen("count.txt", "a" );}
```

Sofern Sie die bisherigen Beispiele mitgespielt haben, kennen Sie den Befehl `fopen` bereits, da wir ihn im Zusammenhang mit dem Gästebuch eingesetzt haben, um die Einträge in einer Textdatei zu speichern. Dieser Befehl wird in diesem Fall „missbraucht“ und mit dem Attribut `a` verwendet, um die Datei anzulegen. Deshalb wird der Befehl in den Block der `if`-Bedingung geschrieben, um die Datei nur anzulegen, wenn sie noch nicht existiert. Dieser Schritt ist notwendig, da mit dem später verwendeten `fopen("count.txt", r+)` die Datei nicht angelegt, sondern nur zum Lesen und Schreiben geöffnet wird.

```
$counter=fopen("count.txt","r+");
```

Nach dem Anlegen gibt es die Datei auf jeden Fall. Mit dem **Attribut** `r+` zum Lesen und Schreiben wird sie nun geöffnet und der Dateihandle erzeugt. Der Dateizeiger (Cursor) steht am Anfang der Datei.

Nun setzen Sie die folgende Funktion ein: `fgets()`.
Info
Diese Funktion liest eine Zeile von der aktuellen Position des Dateizeigers, der durch `fopen("count.txt", "r+")` am Anfang steht, bis die angegebene Anzahl Zeichen (100) oder das Zeilenende erreicht wird. Zurückgegeben wird eine Zeichenkette, die wir der Variablen $aufruf zuweisen.

```
$aufruf=fgets($counter,100);
```

Info
`sring fgets(int file, int length)`

Die Funktion liest aus einer Datei eine Zeile mit der angegebenen Länge (`length`) aus. Der Dateihandle (`file`) muss ein gültiger Handle für eine geöffnete Datei sein.

Eine Länge von 100 Zeichen ist für einen Counter zweifellos ausreichend, es gibt aber Einsatzbereiche, bei denen die Länge deutlich größer sein muss, um die komplette Zeile der Datei einzulesen. In $aufruf ist nach dieser Zuweisung der aktuelle Zählerstand gespeichert. Die Tatsache, dass die Datei gerade erst angelegt wurde und folglich noch kein Zählerstand enthalten ist, führt nicht zu Problemen, da die nächste Zeile die gezählten Aufrufe (Zählerstand) um eins erhöht:

```
$aufruf=$aufruf+1;
```

Auch wenn die TXT-Datei noch leer war, ist in $aufruf nun eine Zahl enthalten, nämlich 1, da PHP bei „Nichts“ bzw. „Null“ + 1 keine Probleme beim Rechnen hat.

Mit der Funktion `rewind()` bewirken Sie nun, dass der Cursor in der Textdatei, in der der Zählerstand gespeichert wird, wieder auf den Anfang gesetzt wird, sodass der neue Zählerstand den alten überschreibt.

```
rewind($counter);
```

Nun schreiben Sie den neuen Zählerstand in die Datei.

```
fputs($counter,$aufruf);
```

Zu guter Letzt heißt es

```
echo $aufruf;
```

um den Zählerstand anzuzeigen.

Schließen Sie PHP, speichern Sie das Dokument als PHP-Datei und testen Sie das Script im Browser. Sie müssten eine Seite angezeigt bekommen, die in etwa dem Bild 1 entspricht.

Klicken Sie dann ein paar Mal auf AKTUALISIEREN. Wenn es geklappt hat, wird die angezeigte Zahl hochgezählt. Der Browser zeigt, wie in Bild 2 zu sehen, den Erfolg.

Ein Counter mit Cookie

Vor der „Erfindung“ von Cookies war das Surfen im Netz ein Reise ohne Geschichte und ist/wäre es auch heutzutage, wenn Cookies unbekannt wären oder wenn ihr Gebrauch im Browser deaktiviert würde. Durch Cookies kann der Server Informationen über den User speichern und sich an ihn „erinnern“. Diese Informationen werden in Form von kleinen Textdateien auf dem Rechner des Surfers gespeichert und vom Browser automatisch an den Server, der das Cookie gesetzt hat, gesendet.
Info
Um weitere Informationen über Cookies zu erhalten, lesen Sie bitte den Exkurs am Ende dieses Abschnitts. In diesem Exkurs gehen wir kurz auch auf die vermeintliche Gefährdung durch Cookies ein.

Ohne Cookies wären viele Sachen im Internet nicht möglich. Ein häufig zitiertes Beispiel dafür sind Online-Einkaufstouren, bei denen Sie ihre gekauften Waren in einen virtuellen Warenkorb legen. Es leuchtet ein, dass während einer solchen Tour Informationen gespeichert werden müssen.

PHP unterstützt Cookies; Sie können mit PHP **Cookies** setzen, Informationen zurückgeben lassen, und Cookies auch wieder löschen. Im Zusammenhang mit einem Zähler werden wir im nächsten Abschnitt also den Programm-Code schreiben, mit dem Sie einen Cookie setzen. Dieser Cookie wird dazu verwendet, eine Reloadsperre zu implementieren, sodass nicht jedes Aktualisieren der Webseite als neuer Aufruf gezählt wird.

Einen Cookie implementieren

Einen Cookie setzen Sie mit dem Befehl `setcookie();`

Als Argumente tragen Sie in die Klammer einen beliebigen Namen und einen Wert ein.
Info

```
int setcookie(string name, string value [, int expire [, string path [, string domain [, int secure]]]])
```

Alle Parameter sind optional außer dem Namen des Cookies (`name`) und dem Wert (`value`). Die Funktion sendet einen Cookie – übertragen im Header – an den Browser.

Um einen Wert von einem Cookie zurückgeben zu lassen, brauchen Sie nur Bezug nehmen auf den Namen

des Cookies, also $cookiename. Werfen Sie einen Blick auf das abgebildete **Script** für den Counter mit Cookie. Sie finden hier die Zeile:

```
setcookie('willi',time());
```

Damit wird das Cookie gesetzt.

```
<?php
if(!file_exists("count.txt")){fopen("count.txt", "a" );}
$counter=fopen("count.txt","r+");
if(!$willi OR $willi<time()-60)
{
setcookie("willi",time());
$aufruf=$aufruf+1;
rewind($counter);
fputs($counter,$aufruf);
}
$aufruf=(string) $aufruf;
for($i=0;$i<strlen($aufruf);$i++)
{
echo "<img src='c1_".$aufruf[$i].".gif'>";
}
?>
```

Erläuterung des Scripts

Das Script beginnt wie das vorherige einfache Script. Neu und ergänzt ist es ab der Zeile

```
if(!$willi OR $willi<time()-60)
{
```

„Willi“ ist unser Name für das Cookie (das noch zu setzen ist). In der damit korrespondierenden Variablen $willi steht – sofern das Cookie „willi“ vorhanden ist – die Zeit des letzten Aufrufs der Seite. Damit das Cookie nur gesetzt wird, und der Zähler um eins hoch gezählt wird, wenn die Seite noch nicht besucht wurde, oder wenn seit dem letzten Aufruf länger als 60 Sekunden vergangen sind, wird mit einer `if`-Bedingung Folgendes getestet:

Ist die Seite zuvor schon besucht worden, wenn nicht, dann ist $willi naturgemäß nicht gesetzt, oder – wenn die Seite schon besucht wurde -, ist eine gewisse Zeit seit diesem Besuch (im Beispiel 60 Sekunden) verstrichen.

Deshalb wird die aktuelle Zeit des Aufrufs (`time()`) mit der Zeit des vorherigen Aufrufs, die im Cookie

„willi“ steht, verglichen, wobei eine Zeitdifferenz von 60 Sekunden berücksichtigt wird. Hierzu ist anzumerken, dass die Funktion `time()` immer die aktuelle Zeit in Sekunden zurückgibt, die seit dem 01.01.1970 um 00:00:00 Uhr verstrichen sind.

Info

```
int time()
```

Die Funktion gibt die aktuelle Zeit zurück, die über den sogenannten UNIX-Zeitstempel ermittelt wird. Der zählt die Sekunden ab dem 01.01.1970 um 00:00:00 Uhr.

Ist eine der Bedingungen in der `if`-Anweisung erfüllt, soll das Cookie erstmalig gesetzt bzw. mit einem neuen Wert gesetzt (überschrieben) werden. Mit `setcookie("willi", time());` wird das Cookie mit dem Namen „willi“ und der aktuellen Zeit des Servers gesetzt.

Danach legen wir fest, dass die gezählten Aufrufe (also der jeweilige Zählerstand) um eins erhöht werden. Dies ist ganz einfach gemacht. In der Variablen $aufruf ist der Wert gespeichert, den die Funktion `fgets` zurückgegeben hat. Also legen wir jetzt fest: `$aufruf=$aufruf+1;`.

Sodann muss in der Textdatei, in der der Zählerstand gespeichert wird, der Cursor wieder auf den Anfang gesetzt werden, damit der neue Zählerstand den alten überschreiben kann. Dazu benutzen wir wie gehabt die Funktion `rewind($counter);`.

Danach folgt die Zeile, um den neuen Zählerstand in die Datei zu schreiben. Dies ist die letzte Anweisung im `if`-Block, der dann mit der geschweiften Klammer geschlossen wird.

```
fputs($counter,$aufruf);
}
```

In der `if`-Anweisung wird der Zähler also nur dann hochgezählt, wenn die Seite das erste Mal aufgerufen wird, bzw. die festgesetzte Zeitspanne (60) verstrichen ist, ansonsten wird der unveränderte Zählerstand, der vor der `if`-Anweisung ausgelesen wird, verwendet.

Darstellung der Ziffern

Danach, d.h. in den folgenden Zeilen des Scripts, geht es darum, wie der Zählerstand angezeigt wird. Für die weitere Darstellung des Counters muss sichergestellt werden, dass die Variable $aufruf als String vorliegt, damit sie entsprechend editiert werden kann (Wir erklären weiter unten noch genauer, warum $aufruf eine `string`-Variable sein muss). Dafür gibt es den einfachen Befehl string

```
$aufruf=(string) $aufruf;
```

Info

`(string)`

Die Funktion verwandelt eine Variable in den Datentyp `string`.

Der Zählerstand soll mithilfe von kleinen Bildern angezeigt werden. Für jede Ziffer des Zählerstandes wird eine GIF-Datei angezeigt, die die entsprechende Ziffer abbildet. (Für den Zählerstand 250 beispielsweise werden drei GIF-Dateien nebeneinander gesetzt.) Die Namen der GIF-Dateien sind alle identisch bis auf die letzte Ziffer, die entsprechend des angezeigten Wertes verändert wird. Sie brauchen also 10 kleine GIF-Dateien, die die Ziffern 0 bis 9 darstellen (diese Bildchen müssten Sie nun, wenn Sie unser Beispiel mitspielen, in einem entsprechenden Programm erstellen).

Dann setzen Sie eine `for`-Schleife ein, damit jeweils eine Stelle des Zählers angezeigt wird. Die `for`-Schleife muss also entsprechend der Anzahl der Ziffern durchlaufen werden. Dazu ist zunächst die Anzahl der Ziffern zu ermitteln. Auch dafür gibt es eine Funktion. Sie lautet `strlen()`.

Diese Funktion gibt die Länge eines Strings zurück. Die Zeile heißt dann (achten Sie darauf, dass in `for`-Schleifen die Argumente mit Semikolon getrennt werden):

```
for($i=0;$i<strlen($aufruf);$i++)
{
```

Info
`int strlen(string zeichenketteX)`

Die Funktion gibt die Länge einer Zeichenkette (zeichenketteX) zurück.

Verwendung von GIF-Dateien

Um die Ziffern darzustellen, müssen wir auf die einzelnen Elemente (des Zählerstands) zugreifen können. Aus diesem Grund haben wir weiter oben $aufruf als **String** definiert, denn in PHP kann man auf die einzelnen Stellen einer String-Variablen wie auf ein Array zugreifen. Im nullten Element wird die erste Ziffer von links gespeichert.

Ein Beispiel: Bei einem Zählerstand von 250 ist $aufruf[0]=2, $aufruf[1]=5 und $aufruf[3]=0. Sodann brauchen wir den `echo`-Befehl, damit das Bild mit der richtigen Ziffer angezeigt wird. (Die Dateien heißen bei uns `c1_0.gif` etc.) Beim ersten Schleifendurchlauf ist die gesetzte Variable $i=0, somit wird $aufruf[0] verwendet und `echo` gibt aus: `<img src='c1_2.gif'>`. Das Bild `c1_2` soll vereinbarungsgemäß eine 2 anzeigen. Beim nächsten Durchlauf wird der Wert für $aufruf[1] verwendet, somit wird die Datei `c1_5.gif` angezeigt. Als Programmzeile sieht das dann so aus:

```
echo "<img src='c1_".$aufruf[$i].".gif'>";
}
```

Cookies, Cookies

Obwohl gern gegessen, haben Cookies im Internet einen eher schlechten Ruf. Bei Usern, die sich noch nicht näher mit dem Thema befasst haben, ist die Meinung weit verbreitet, dass über Cookies persönliche Daten aller Art weitergereicht werden. Dabei sind Cookies, auch die im Internet, zunächst einmal recht harmlos. Es handelt sich um Textinformationen, die auf Veranlassung eines Webservers z.B. mithilfe von PHP durch den Browser gespeichert werden. Wird die Webseite erneut aufgerufen, oder eine Seite der gleichen Domäne, sendet der Browser die Textinformation des Cookies an den Webserver. Von daher sind Cookies zunächst einmal völlig unproblematisch. Der Webserver kann nur Informationen in Cookies speichern, die ihm ohnehin schon bekannt sind und somit auch nur diese Informationen vom Browser wieder zurückgesandt bekommen.

Bei dieser Beschreibung sind Sicherheitslücken der Browser nicht berücksichtigt. So lassen der Internet-Explorer 5 und Internet-Explorer 6 auch einen Zugriff von anderen Domänen zu, sofern das entsprechende Sicherheitspatch nicht nachinstalliert wurde.

Cookies werden inzwischen bei fast allen dynamischen Webseiten eingesetzt und sie sind fast unvermeidbar auf Webseiten, bei denen Sie sich mit Benutzername und Passwort anmelden, da beim Wechsel von einer Webseite zur nächsten ihre Anmeldung nicht vergessen werden darf. Der Webserver ohne Cookies hätte keine vernünftige Möglichkeit, festzustellen, dass Sie vor fünf Minuten die Login-Seite besucht haben, und nun auf die Seite mit den neuesten Angeboten gewechselt sind. Auch die meisten Counter (Zähler) arbeiten mit Cookies, um eine sogenannte Reload-Sperre zu realisieren, also dafür zu sorgen, dass der Counter nicht jeden Klick auf den Aktualisieren-Button mitzählt. Fazit: Wenn sie sich dazu entschließen, Cookies zu deaktivieren, verzichten Sie auch auf eine Menge nützlicher Auswirkungen dieser kleinen Kekse.

Missbrauch von Cookies?

Aber es gibt mal wieder zwei Seiten der gleichen Medaille. Missbraucht werden Cookies mitunter dazu, das Surfverhalten von Internet-Usern auszuspionieren. Laut Festlegung können Cookies nur von der Domäne gelesen werden, von der sie auch gesetzt wurden, sodass zunächst nur Ihr Surfverhalten innerhalb einer Domäne verfolgt werden kann. Aber pfiffige Menschen haben eine Methode entwickelt, Cookies auch von außerhalb der besuchten Domäne auf Ihren Browser zu setzen. Hierbei handelt es sich um die Cookies von Drittanbietern.

Um ein Beispiel zu geben: Über den Aufruf eines Bildes innerhalb der besuchten Webseite, das auf der Domäne des Drittanbieters liegt, kann diese Domäne einen Cookie bei Ihnen setzen bzw. die gesetzten auslesen.

Was heißt das nun für Sie als Surfer? Kommerzielle Firmen machen Webseitenanbietern das Angebot, Daten über das Surfverhalten ihrer Besucher zu generieren. Alle Webseitenanbieter, die sich einer dieser Firmen, z. B. der Firma XY, anschließen, fügen eine kleine Grafik von der Domäne XY auf ihren Webseiten ein. Besuchen Sie eine solche Webseite, z.B. die von „Müller und Sohn“, das erste Mal, so wird ein Cookie der Domäne XY mit einer eindeutigen Nummer, z.B. 4711 bei Ihnen gesetzt. Die Firma XY speichert, dass Sie als 4711 die Webseite „Müller und Sohn“ besucht haben. Landen Sie beim Surfen bei einem weiteren

Kunden der Firma XY, z.B. bei „Meier und Tochter“, so wird Ihre Nummer 4711 ausgelesen und die Firma XY speichert, dass Sie als 4711 auch bei „Meier und Tochter“ zu Besuch waren. So zieht sich das von Kunde zu Kunde der Firma XY. Schließlich lässt sich, wenn die Kunden der Firma XY ihre Daten abgleichen, ohne größeren Aufwand folgende Zuordnung vornehmen: Sie als 4711 haben bei „Müller und Sohn“ Zahnpasta und Rasierschaum bestellt und bei der Firma „Meier und Tochter“ die Seiten über Rasenmäher und Gartenmöbel angeschaut. Schon kann man mit einiger Plausibilität den Schluss ziehen, dass Sie männlich und Hausbesitzer sind, sodass bei weiteren Besuchen der Kundenseiten der Firma XY die Werbeeinblendungen oder Artikelpräsentation auf Sie zugeschnitten werden können!

Nachdem die bedenklichen Seiten der Cookies von Drittanbietern beschrieben wurde, an dieser Stelle auch Beispiele, wo diese Cookies sinnvoll sind und allseits nützlich eingesetzt werden.

Webmaster binden häufig Inhalte/Angebote von so genannten Content Providern in ihre Seiten ein. Mitunter ist für die Funktion dieses eingebundenen Contents das Akzeptieren von Cookies von Drittanbietern Voraussetzung. Beispiele für diese Art Content sind: Newsticker, Gästebücher, Foren, Chats, Counter, Umfragen etc.

Ein Counter mit fester Stellenanzahl

Das Script kann noch erweitert werden, wenn Sie wünschen, dass der Zählerstand mit einer festen Anzahl an Stellen angezeigt wird. Die ersten Stellen von links werden mit Nullen aufgefüllt. Um dies zu erreichen, wird nur der Teil des Scripts, der den Zählerstand ausgibt, verändert, der erste Teil, der den Zählerstand ermittelt, bleibt wie gehabt. Die Mühe hält sich also in Grenzen!

Das erweiterte Script

Das Script, das ein Cookie setzt, die Ziffern als GIFs anzeigt und für die feste Stellenanzahl sorgt, sehen Sie unten im Code.

```
<?php
if(!file_exists("count.txt")){fopen("count.txt", "a" );}
$counter=fopen("count.txt","r+");
if(!$willi OR $willi<time()-60)
{
setcookie("willi",time());
$aufruf=$aufruf+1;
rewind($counter);
fputs($counter,$aufruf);
}
$aufruf=(string) $aufruf;
$temp=0;
for($i=0;$i<6;$i++)
{
if(6-$i>strlen($aufruf))
```

```
{echo "<img src='c1_0.gif'>";}
else
{
echo "<img src='c1_".$aufruf[$temp].".gif'>"; $temp=$temp+1;
}
}//Ende for
?>
```

Erläuterung des Scripts

Der erste Teil ist – wie gesagt – unverändert. Sie können Ihre Aufmerksamkeit also gleich den Änderungen widmen, die mit der `for`-Schleife beginnen. Mit $temp wird ein Hilfszähler zunächst auf 0 gesetzt. Mit Hilfe dieser **Variablen** wird die tatsächliche Anzahl vorhandener Stellen des Zählerstandes durchlaufen, `$temp=0;`

Mittels der `for`-Schleife wird die gewünschte Anzahl der angezeigten Stellen des Zählers, in diesem Fall 6, durchlaufen.

```
for($i=0;$i<6;$i++)
{
```

Sodann folgt wieder eine `if`-Bedingung. Mit dieser `if`-Anweisung wird die GIF-Datei mit der 0 für die führenden Stellen des Zählers ausgegeben. Wir müssen also anweisen: solange $i noch nicht in dem Bereich der Stellen, deren Werte aus dem Zählerstand ermittelt werden, angelangt ist, ist die Bedingung erfüllt (und die entsprechende GIF-Datei `c1_0.gif` wird angezeigt):

```
if(6-$i>strlen($aufruf))
{echo "<img src='c1_0.gif'>";}
else
{
echo "<img src='c1_".$aufruf[$temp].".gif'>";
$temp=$temp+1;
}
```

Zählerstand 250: ein Beispiel

Das Ganze ist ein bisschen trickreich und deshalb nehmen wir zur Verdeutlichung mal ein konkretes Beispiel an: Angenommen, der Zählerstand ist 250, dann ist `strlen($aufruf)`= 3 (drei Ziffern). Eine sechsstellige Zahl soll angezeigt werden, die ersten drei Stellen von links müssen also jeweils eine Null aufweisen. Folglich muss die Prüfung der `if`-Bedingung `true` ergeben. Dies ist für $i=0 bis $i=2 der Fall, denn nach Adam Riese gilt: 6-0>3, 6-1>3, 6-2>3. Mit `echo` wird die entsprechende GIF-Datei ausgegeben. Anschließend ist die Bedingung `false`: 6-3 ist nicht >3.

Ab jetzt wird der `else`-Zweig ausgeführt. Hier wird der Hilfszähler $temp wichtig, da der Wert nicht wie

oben in $aufruf[$i] zu finden ist, (da im Beispiel $i=3 ist) und $aufruf[3] somit eine 0 ausgeben würde (dritte Stelle von 250). Es wird also für die folgenden Schleifendurchläufe, wenn der `else`-Zweig ausgeführt wird, nacheinander $aufruf[0], [1], [2] benötigt. Dies erreichen wir durch den Hilfszähler, der jeweils um eins erhöht wird.

Die letzte geschweifte Klammer schließt die `for`-Schleife.

Erweiterung für „viel besuchte“ Webseiten

Im obigen Beispiel wurde die maximale Anzahl der Stellen des Zählers auf 6 gesetzt, diese Zahl können Sie natürlich im Script ändern. Aber was passiert, wenn ein ungeahnter Besucheransturm – Sie haben z.B. die neueste Version Ihres Moorhuhn-Ablegers gerade ins Netz gestellt – Ihren Zähler innerhalb von Minuten diese Grenze sprengen lassen. Im obigen Beispiel würde der Zähler die hinteren Stellen der Besucherzahl einfach abschneiden. Bild 6 zeigt den Zählerstand in der Textdatei und im Browser.

Für diesen Fall können Sie das Script so anpassen, das der Counter aus mindestens 6 Stellen besteht, sollte der Zählerstand höher sein, werden aber auch mehr Ziffern angezeigt. Code unten zeigt das modifizierte Scriptfragment:

```
$aufruf=(string) $aufruf;
$temp=0;
$anzstellen=max(6, strlen($aufruf));
for($i=0;$i<$anzstellen ;$i++)
{
if($anzstellen-$i>strlen($aufruf))
{echo "<img src='c1_0.gif'>";}
else
{
echo "<img src='c1_".$aufruf[$temp].".gif'>";
$temp=$temp+1;
}
}//Ende for
?>
```

Erläuterung des erweiterten Scriptteils

Das Script hat nur kleine Veränderungen zum Vorgänger. Anstatt die gewünschte Anzahl der Stellen festzusetzen, wird diese in einer Variablen namens $anzstellen gespeichert. Diese Variable findet in der `for`-Schleife und in der `if`-Bedingung Verwendung:

```
for($i=0;$i<$anzstellen ;$i++)
```

sowie

```
if($anzstellen-$i>strlen($aufruf))
```

Der Trick: wir weisen der Variablen zuvor die gewünschte Anzahl mindestens anzuzeigender Stellen oder, wenn die Länge des Zählerstands größer ist, die Länge des Zählerstandes zu.

```
$anzstellen=max(6, strlen($aufruf));
```

Die Funktion `max()` gibt immer den höchsten Wert der angegebenen Argumente zurück, im Beispiel also 6 oder die Länge des Zählerstandes.

Wenn die Ergänzung geklappt hat, sind der Zählerstand der Textdatei und der angezeigte Zählerstand nun identisch, zu sehen in Bild 7.

Info

```
mixed max(mixed argument1, argument2,)
```

Die Funktion ermittelt den höchsten Wert der übergebenen Werte. Zwei Werte sind logischerweise zwingend erforderlich.

Ausblick – Identifikation über die IP-Adresse

Cookies genießen einen schlechten Ruf und aus diesem Grund haben viele Surfer Cookies im Browser deaktiviert. In diesem Fall wird die Reloadsperre nicht funktionieren, da kein Cookie gesetzt werden kann. Es wird beim User aber auch keine Fehlermeldung angezeigt, der Counter wird einfach bei jedem Aufruf hochgezählt, egal, ob 60 Sekunden seit dem letzten Aufruf vergangen waren oder nicht.

Anstelle von Cookies wird deswegen mitunter auch die IP-Adresse des Users verwendet, um festzustellen, ob der Surfer die Seite zuvor besucht hat oder nicht. Der Browser sendet bei jedem Aufruf die IP-Adresse, die der Surfer – meistens dynamisch – von seinem Provider zugewiesen bekommen hat, an den Server. Anhand der IP-Adresse kann man den Surfer an sich identifizieren, da die zugewiesene IP-Adresse einmalig ist.

Die Sache hat aber einen Haken, da alle Theorie grau ist: die Zuordnung ist leider nur angeblich eindeutig, in der Praxis werden die dynamischen IP-Adressen mitunter während des Surfens vom Provider geändert oder der User verwendet einen Proxyserver oder NAT-Server, sodass unter Umständen mehrere Besucher der Seite die gleiche IP-Adresse aufweisen.

Welche Lösung für einen Counter zu bevorzugen ist, kann man nicht pauschal sagen. Es ist abzuwarten, ob der Trend, Cookies auszuschalten, abflaut oder nicht. Im Allgemeinen ist es so, dass Sie über den Counter mit Cookies eher mehr Besucher zählen, da mitunter die Cookies deaktiviert sind, während Sie über die IP-Adresse eher weniger Besucher zählen, da Proxy- und NAT-Server unabhängig davon, wie viele Menschen über die Server auf Ihre Seite gleichzeitig zugreifen, nur als ein Surfer/Aufruf gewertet werden.

Die IP-Methode

Dieses Verfahren über die IP-Adresse wird hier aus Platzgründen nur prinzipiell erklärt, es folgt kein ausführliches Script.

Der Weg ist folgender:

Bei jedem Aufruf der Seite wird die IP-Adresse des Surfers in einer Datenbanktabelle gesucht. Ist noch kein Eintrag vorhanden, wird der Seitenaufruf gezählt und ein neuer Datensatz in die Tabelle mit der IP-Adresse des Users und der aktuellen Uhrzeit geschrieben. Ist bereits ein Datensatz mit der IP-Adresse vorhanden, wird getestet, ob der letzte Aufruf lange genug zurückliegt, um den Aufruf erneut zu zählen. Ist dies der Fall, wird der Zähler um eins erhöht und der Datensatz mit der IP-Adresse mit der neuen aktuellen Uhrzeit aktualisiert. Ist noch nicht genug Zeit verstrichen, passiert nichts, weder wird der Zähler verändert noch die Uhrzeit des Datensatzes.

Bei diesem Verfahren sammeln sich schnell viele Einträge in der Datenbank, denn jeder Besucher generiert einen Datensatz. Daher ist es sinnvoll, von Zeit zu Zeit aufzuräumen und alte Datensätze zu löschen. Am einfachsten geht dies, wenn Sie bei jedem Aufruf alle Datensätze löschen, deren Uhrzeit anzeigt, dass der Aufruf länger als 1 Stunde (oder Ähnliches) zurückliegt.

Für die Realisierung dieser Variante benötigen Sie die IP-Adresse des Users. Diese stellt Ihnen PHP in der Variablen $REMOTE_ADDR zur Verfügung.

Oft ist es erwünscht, eine Benutzereingabe nicht nur auf einer Internetseite anzuzeigen, sondern den Benutzer auch via **E-Mail** davon zu informieren. E-Mails sind leichter zu archivieren als Verweise auf dynamische Webseiten und werden deshalb oft als Bestätigung von Bestellungen oder Reservierungen zusätzlich eingesetzt. In der einfachsten Form reicht eine Zeile PHP-Code aus, um eine E-Mail zu verschicken:

```
mail("user@server.com", "Greetings", "Hello User");
```

Mit dieser Anweisung wird die Nachricht mit dem Betreff „Greetings“ an den Benutzer user bei server.com zugestellt. Der Inhalt der Nachricht beschränkt sich auf die Zeile „Hello User“. Diese Form hat aber noch einige Nachteile: So wird als Absender der E-Mail der Benutzeraccount angegeben, unter dem der Webserver läuft (oft apache oder www-data). Je nachdem, ob und wie die Namesrückauflösung für den Server eingestellt ist, wird der Domain-Name ergänzt, was oft dazu führt, dass der Empfänger nicht auf diese Nachricht antworten kann.

Damit Sie weitere Details des Mail-Transports mit **PHP** verstehen, werden im Folgenden einige grundsätzliche Eigenschaften der E-Mail-Zustellung erläutert.
Info
Der E-Mail-Standard RFC822 verbietet es, dass Zeilen im Text von E-Mails einfach durch `line feed` (entspricht `\n`) getrennt werden. Stattdessen ist eine Trennung durch `carriage return` und `line feed` vorgeschrieben. Wenn Sie zum Versenden von E-Mails die PHP-Funktion `mail` verwenden, müssen Sie

unbedingt darauf achten, statt `\n` die Kombination `\r\n` zu verwenden!

Probleme beim E-Mail-Versand

Das Versenden von E-Mails von lokalen Testrechnern bereitet oft Probleme: Unter Windows steht zumeist kein lokaler E-Mail-Server zum Versenden zur Verfügung. php.ini kann zwar so eingestellt werden, dass mail auf den **SMTP-Server** Ihres Internet-Providers zugreift, aufgrund verschiedener SPAM-Schutzmaßnahmen kann es aber dennoch passieren, dass der SMTP-Server die von PHP kommenden E-Mails einfach ignoriert.

Unter Linux läuft zwar oft `sendmail` oder `postfix`, aber diese Programme müssen häufig erst konfiguriert werden. Mit diesem Thema lassen sich eigene Bücher füllen, weswegen hier ein einziger Tipp ausreichen muss: Stellen Sie sicher, dass der Hostname des E-Mail-Servers richtig eingestellt ist und dass es sich dabei um einen im Internet gültigen Namen handelt (nicht nur im lokalen Netz)! Falls Sie postfix verwenden, ändern Sie in /etc/postfix/main.cf den Parameter `myhostname` und starten postfix anschließend neu.

> Selbst wenn das Versenden an sich klappt, wird die E-Mail vom Empfänger möglicherweise abgelehnt,

wenn die IP-Adresse des Absenders (also die IP-Adresse, unter der Ihr Rechner aus dem Internet sichtbar ist) und der vom E-Mail-Server verwendete Hostname nicht übereinstimmen. Diese an sich vernünftige Maßnahme versucht das Versenden von SPAM zu unterbinden.

Bevor Sie eine Menge Zeit vergeuden, um das Versenden von E-Mails von Ihrem lokalen Rechner in den Griff zu bekommen, testen Sie die `mail`-Funktion Ihrer PHP-Projekte auf der PHP-Installation Ihres ISPs bzw. auf einem Root-Server: In der Regel treten dabei keine Probleme auf, sofern der Domainname in Ihrer Absenderadresse mit dem tatsächlichen Hostnamen des Rechners übereinstimmt.

PHP-Mail mit `sendmail` unter Unix/Linux

Die Arbeitsweise der `mail()`-Funktion unterscheidet sich grundlegend unter Windows- und UNIX-Betriebssystemen. So steht unter UNIX-Betriebssystemen ein Programm zur Verfügung, das sich um den **Mail-Transport** kümmert, ein Mail Transport Agent (MTA). Das macht sich PHP zunutze und delegiert die Arbeit des E-Mail-Verschickens an dieses Programm. In der PHP-Konfiguration können spezielle Optionen für den Programmaufruf angegeben werden, zum Beispiel der Pfad, sollte das Programm nicht standardmäßig installiert worden sein.

Der am weitesten verbreitete MTA ist `sendmail`. Bekannt durch seine nicht triviale Konfiguration und die Flexibilität, ist `sendmail` heute auf großen Mailservern von Universitäten und Firmen gleichermaßen wie auf dem Linux-Desktop im Einsatz. Inzwischen gibt es einige Programme, die die Funktionalität von `sendmail` nachahmen und dabei eine übersichtlichere Konfiguration versprechen. Qmail, postfix oder

smail sind einige der prominenten Vertreter. Da alle diese Programme ein `sendmail`-Programm mitbringen, das noch dazu die wichtigsten Parameter des Originals unterstützt, ändert sich für die PHP-Konfiguration nichts.

> Die weitere Zustellung hängt von der Konfiguration des MTA ab.

Dort kann eingestellt werden, ob die Nachricht direkt an den Mailserver des Empfängers übermittelt oder an einen anderen SMTP-Server weitergeleitet wird. Für PHP ist der Prozess abgeschlossen.

Um das in der Einleitung angesprochene Problem mit dem falschen Absender zu lösen, kann der **MTA** zusätzlich zum E-Mail-Text auch die Kopfzeilen der E-Mail verändern, um den richtigen Absender einzutragen. So reicht die Kopfzeile

```
From: anotherUser@server.com
```

aus, um dem Empfänger die richtige Absenderadresse mitzuteilen. Der Aufruf im **PHP-Code** ändert sich damit auf:

```
mail("user@server.com", "Greetings", "Hello User", "From:
anotherUser@server.com");
```

Die Umsetzung weiterer Kopfzeilen (zum Beispiel From, Reply-To, User-Agent, …) erledigt der MTA.

PHP-Mail via SMTP unter Windows

Unter Windows-Betriebssystemen gehört das `sendmail`-Paket nicht zum Standardumfang. Hier braucht man eine andere Strategie, um eine E-Mail zu verschicken. PHP kontaktiert dazu einen externen Mailserver und gibt die Nachricht an diesen weiter. In der Konfigurationsdatei wird dieser Eintrag in der Sektion `[mail functions]` mit SMTP bezeichnet. Dieser SMTP-Server wird auch beim traditionellen E-Mail als Postausgangsserver oder eben SMTP-Server eingestellt.

Wegen der Problematik von unerwünschten E-Mail-Nachrichten (SPAM) nehmen richtig konfigurierte SMTP-Server nur Nachrichten an, für die sie auch zuständig sind. Ist man beispielsweise über ein Einwahlnetzwerk mit dem Internet verbunden, so stellt in der Regel der Provider einen SMTP-Server zur Verfügung.

> Ist die Konfiguration von PHP richtig eingestellt, funktioniert das E-Mail-Verschicken auch unter Windows.

Einschränkungen gibt es allerdings für PHP-Versionen vor 4.3: Extra Kopfzeilen (zum Beispiel From:, Reply-To:, ...) werden nicht umgesetzt. Einzig die Cc:-Kopfzeile funktioniert auch mit älteren PHP-Versionen unter Windows.

Sehr strikte Server-Administratoren verlangen auch für das E-Mail-Verschicken einen gültigen Benutzeraccount und ein Passwort am SMTP-Server. Das in RFC2554 festgeschriebene Verfahren ist noch eher jung (1999), und daher unterstützen es noch nicht alle Mail-Clients. Um mit PHP einen authentifizierten E-Mail-Versand zu erreichen, bedient man sich am besten der **PEAR-Mail-Klassen**, die seit PHP 4.1 zum Standardumfang von PHP gehören.

E-Mail mit PEAR

Die in PEAR enthaltene Klasse zum E-Mail-Versand (`PEAR::Mail`) geht über die Funktionalität des PHP-`mail()`-Aufrufs hinaus. So können mit `PEAR::Mail` auch Attachments verschickt oder HTML-E-Mails erzeugt werden. Die Art des Versands (`smtp` oder `sendmail`, bei PEAR als Treiber oder Backend bezeichnet) kann von Fall zu Fall ausgewählt werden.

Um `PEAR::Mail` zu verwenden, muss als Erstes eine Instanz der Hauptklasse Mail erzeugt werden:

```
$mail_inst =& Mail::factory('smtp', $params);
```

Die Instanz wird von der Funktion `factory` angelegt, der als erster Parameter der Treiber (in diesem Fall `smtp`) und weiters ein Array mit Einstellungen übergeben wird. Erfordert der Postausgangsserver zum Beispiel eine Validierung, könnten die Parameter so aussehen:

```
$params['host'] = 'smtp.server.com';
$params['port'] = '25';
$params['auth'] = true;
$params['username'] = 'user';
$params['password'] = 'password';
```

Plain-Text-E-Mail mit PEAR

`send_text_mail.php` verschickt eine kurze Textnachricht über den Server `smtp.server.com`. Die Kopfzeilen (Betreff, Absender und Empfänger) können dabei übersichtlich als **Array** definiert werden.

```
<?php // Datei mail/send_text_mail.php
require('Mail.php');
$headers['From'] = 'Another User <anotheruser@host.com>';
$headers['To'] = 'User <user@host.com>';
$headers['Subject'] = 'Testnachricht von PEAR::Mail';
$recipients = 'user@host.com';
```

```
$body = 'Hallo User!
Das ist eine kleine Testnachricht.';
$params['host'] = 'smtp.server.com';
$params['port'] = '25';
$mail_object =& Mail::factory('smtp', $params);
if ($mail_object->send($recipients,
$headers, $body)== TRUE) {
echo 'Ihre Nachricht wurde verschickt';
} else {
echo 'Fehler beim Senden Ihrer Nachricht.';
}
?>
```

Ohne die Kopfzeile für den Empfänger (`$headers['To']`) wird die E-Mail zwar trotzdem zugestellt (beim `send`-Aufruf wird `$recipients` verwendet), das E-Mail-Programm zeigt aber keine Adresse an.

MIME-E-Mail mit PEAR

Im nächsten Beispiel wird folgendes Szenario behandelt: Auf einer Internetseite befinden sich drei Bilder, aus denen der Benutzer eines auswählen kann und es via E-Mail an einen Freund verschickt. Zu dem Bild kann noch ein kurzer Text eingegeben werden.
Info

Die **Mail_Mime-Klasse** aus dem PEAR-Paket gehört, im Unterschied zu den bisher besprochenen Mail-Klassen, nicht zum Standardumfang von PHP. Um die praktischen Funktionen der Mime-Klasse zu verwenden, muss das Paket Mail_Mime-Klasse PEAR heruntergeladen werden. Dabei ist es ausreichend, das Paket im Arbeitsverzeichnis zu entpacken.

Alternativ kann das Paket auch mit dem PEAR-Installer eingespielt werden:

```
root# pear install Mail_MIME
```

Um eine gültige E-Mail mit einem binären Anhang zu erzeugen, muss die Nachricht in das MIME-Format umgewandelt werden. Diese Prozedur selbst zu programmieren, ist etwas aufwändig, aber zum Glück gibt es eine Klasse im PEAR-Paket, die diese Arbeit erledigt: `Mail_mime`. Diese Klasse erzeugt die Kopfzeilen und den Inhalt der E-Mail und wird dann mit der schon bekannten Funktion `send` aus der PEAR::Mail-Klasse verschickt.

```
$mime_message = new Mail_mime();
$mime_message->setTXTBody('Im Anhang das Bild');
$mime_message->addAttachment('/tmp/flower.jpg', 'image/jpeg');
$mime_body = $mime_message->get();
```

```
$mime_hdr = $mime_message->headers($headers);
```

Nach diesen Aufrufen enthält `$mime_body` den gesamten Inhalt der E-Mail (inklusive des Bilds flower.jpg) und das Array `$mime_hdr` die Kopfzeilen. Um möglichst plattform-unabhängig zu sein, wird das Bild dabei in **base64** kodiert, einem Format, das nur mit ASCII-Zeichen auskommt.

Die gesamte Anwendung kann in einer Datei implementiert werden:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="de">
<head>
<title>E-Mail-Grußkarten</title>
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-1">
</head>
<body>
<?php // Datei: mail/send_image.php
require 'Mail.php';
require 'Mail/mime.php';
if ($_POST['submit'] === 'Abschicken') {
if (strlen($_POST['message']) > 400) {
echo 'Fehler: Der Text Ihrer Nachricht ist zu lang';
echo '<br><a href="send_image.php">Neuer Versuch</a>';
echo '</body></html>';
exit();
}
```

Nach der Ausgabe der HTML-Kopfzeilen werden die benötigten PEAR-Klassen geladen: Mail.php (zum Überprüfen der Empfängeradressen) und mime.php. Wurde das Formular abgeschickt (`$_POST['submit'] === 'Abschicken'`), so beginnt die Ausführung des PHP-Codes:

Zuerst wird die Länge der Nachricht überprüft. Sind es mehr als 400 Zeichen, wird das Programm beendet und der Anwender kann neu beginnen. Ist die Länge der Nachricht korrekt, werden der oder die Empfänger überprüft:

```
// überprüfen der eingegebenen Empfänger
$mail_instance = new Mail();
$recipients = $mail_instance->parseRecipients($_POST['email']);
if (count($recipients) <= 0) {
echo 'Es wurden keine gültigen Empfänger ermittelt';
echo '<br><a href="send_image.php">Neuer Versuch</a>';
echo '</body></html>';
exit();
```

```
}
```

Dazu wird eine Instanz der Mail-Klasse benötigt, über die dann der Aufruf an `parseRecipients` gültige Adressen aus einer Zeichenkette extrahiert (intern wird dabei auf die Klasse RFC822 zurückgegriffen). Das erspart eine Menge Arbeit beim Überprüfen der Eingabe und ermöglicht eine flexible Angabe der E-Mail-Adressen. So kann `parseRecipients` aus den Strings

```
Username <user@server.com>, anotherUser@server.com
```

und

```
user@server.com
```

gültige Adressen herausfiltern. Der Rückgabewert der Funktion (das Array `$recipients`) enthält nur mehr gültige E-Mail-Adressen. Ist diese **Variable** leer, wird die Ausführung mit dem Hinweis beendet, dass keine Empfänger gefunden werden konnten.

```
$headers = array (
'Subject' => 'Eine Nachricht mit Bild',
'From' => 'Internetformular <webform@server.com>'
);
$mime_message = new Mail_mime();
$mime_message->setTXTBody($_POST['message']);
$mime_message->addAttachment($_POST['img'], 'image/jpeg');
$mime_body = $mime_message->get();
$mime_hdr = $mime_message->headers($headers);
$final_message =& Mail::factory('mail');
if ($final_message->send($recipients, $mime_hdr,
$mime_body) == TRUE) {
echo '<h1>Ihre Nachricht wurde verschickt</h1>';
} else {
echo '<p>Es ist ein Fehler aufgetreten, Ihre Nachricht konnte
nicht verschickt werden</p>';
}
}
?>
```

Im Anschluss werden die Kopfzeilen vorbereitet (in diesem Beispiel wird der Absender fix eingestellt), und die Nachricht wird, wie bereits beschrieben, in das **MIME-Format** konvertiert. Der Versand der Nachricht erfolgt über die Mail-Klasse von PEAR. In diesem Beispiel wird als Backend die von PHP zur Verfügung gestellte `mail`-Funktion verwendet.

```
<FORM action="send_image.php" method="POST">
<table summary="Bilder">
<caption>Bilder</caption>
<tbody>
<tr>
<td><IMG src="img1.jpg" alt="Bild 1" border="0"></td>
<td><IMG src="img2.jpg" alt="Bild 2" border="0"></td>
<td><IMG src="img3.jpg" alt="Bild 3" border="0"></td>
</tr>
<tr align="center">
<td>Bild 1<INPUT type="radio" checked name="img" value="img1.jpg"></td>
<td>Bild 2<INPUT type="radio" name="img" value="img2.jpg"></td>
<td>Bild 3<INPUT type="radio" name="img" value="img3.jpg"></td>
</tr>
</tbody>
</table>
<p>Nachricht (max. 400 Zeichen):</p>
<textarea name="message" cols="50" rows="10"></textarea>
<p>Empfänger:</p>
<INPUT type="text" name="email" size="25">
<INPUT type="submit" name="submit" value="Abschicken">
<INPUT type="reset" name="reset" value="Reset">
</FORM>
</body>
</html>
```

Der restliche HTML-Code ist sehr schlicht und stellt die Auswahl der Bilder und die Textfelder für Nachricht und Empfänger zur Verfügung.

Das Problem tritt schon bei Betrieben mittlerer Größe auf: Ist das Besprechungszimmer heute um 15:00 Uhr frei, oder findet zu dieser Zeit bereits eine Sitzung statt? Wenn außerdem firmeneigene Geräte zum Ausleihen zur Verfügung stehen, ergibt sich bald die Notwendigkeit, ein Verwaltungssystem dafür zu installieren. Bei großen Unternehmen oder Universitäten können diese logistischen Probleme nicht ohne durchdachte Planung bewältigt werden.

Dem vorliegenden Beispiel fehlt eine wichtige Komponente, um es in der Praxis zu verwenden: die Multi-User-Fähigkeit. Es sieht nicht vor, dass mehrere Benutzer von unterschiedlichen Arbeitsplätzen aus die gleichen Daten bearbeiten können. Dieser Mangel kommt daher, dass die von Ihnen eingegebenen Daten lediglich in einer Session gespeichert werden. Um das System Multi-User-fähig zu machen, müssten die Daten in einer Datenbank abgelegt werden – aber das würde hier zu viel Wissen voraussetzen. Das Beispiel erfüllt daher ausschließlich den Zweck, Ihnen die objektorientierten Konzepte zu verdeutlichen.

Der Vorteil an der **Session-Speicherung** ist, dass die Methoden zum Speichern und Abfragen der Objekte sehr einfach sind. In der Session-Umgebung wird ein neues Array erzeugt, in dem alle Objekte abgelegt werden. Der Nachteil an diesem Ansatz ist, dass alle Daten im digitalen Nirwana verschwinden, sobald Sie

Ihren Browser schließen.
Info

Für Buchungssysteme in der Art dieses Beispiels existieren fertige **PHP-MySQL-Lösungen**. PHP-MySQL-Lösungen

Bedienung der Anwendung

Zu Beginn der Anwendung müssen Sie Ressourcen erzeugen, die zur Verwendung bereitstehen. Im vorliegenden Beispielprogramm stehen Ihnen fünf verschiedene Ressourcen zur Verfügung: ein Besprechungszimmer, ein Unterrichtsraum, ein Beamer, eine Digitalkamera und ein Flipchart. Alle diese Objekte haben unterschiedliche Eigenschaften, z.B. kann bei einem Besprechungszimmer die Anzahl der Sitzplätze eingegeben werden, während bei einem Beamer die Auflösung gespeichert wird.

Nachdem Sie eine oder mehrere Ressourcen hinzugefügt haben, können Sie beginnen,

Ihre Termine vorzumerken. Wählen Sie dazu die Verfügbarkeitsanzeige des gewünschten Geräts oder Raums, und geben Sie dort den Zeitraum ein. Kommt es beim Eintragen zu einer Überschneidung mit einem bereits gebuchten Termin, werden Sie darauf hingewiesen. Abbildung 1 zeigt die Verfügbarkeit eines Unterrichtsraums und den Versuch, einen bereits gebuchten Termin zu reservieren.

Sobald Sie Ihren Browser schließen, gehen die Session und mit ihr alle Ressourcendaten verloren. Sie können die Daten aber auch explizit löschen, in dem Sie den Link ALLE DATEN LÖSCHEN anklicken. Dabei werden alle relevanten Session-Variablen gelöscht und Sie werden auf die Startseite weitergeleitet.

Die Klassenstruktur

Die Basisklasse: Resource

Alle Klassen, die Sie als Ressourcen auswählen können, leiten sich von der Basisklasse Resource ab. Sie stellt diese Funktionen zur Verfügung, die alle Ressourcen gleich implementieren. Außerdem beinhaltet sie eine abstrakte Funktion: `showCreate`. Jede Subklasse muss diese Funktion überschreiben. Klassen, die eine oder mehrere abstrakte Funktionen beinhalten, werden als `abstract class` definiert. Abstrakte Klassen können Sie sich ein wenig wie Interfaces vorstellen. Von ihnen kann keine Instanz erzeugt werden, und sie geben bestimmte Funktionen vor, die überschrieben werden müssen. Der große Unterschied zu Interfaces ist aber, dass abstrakte Klassen sehr wohl **Member-Variablen** und nicht abstrakte Funktionen beinhalten können.

Zwei interessante Member-Variablen der Basisklasse sind als Arrays enthalten: In `$reservations` werden alle festgelegten Termine gespeichert, und `$valid` enthält eine Liste von gültigen Unterklassen.

```
abstract class Resource {
public $id;
public $name;
```

```
public $reservations = array();
// vorhandene Subklassen
public static $valid = array(
"MeetingRoom" => "Besprechungszimmer",
"LectureRoom" => "Unterrichtsraum",
"Beamer"      => "Beamer",
"DigiCam"     => "Digitalkamera",
"FlipChart"   => "Flipchart"
);
// zeigt ein HTML-Formular zum Erstellen einer neuen Klasse
// jede Unterklasse muss diese Funktion überschreiben
public abstract function showCreate();
```

Auf diese Definitionen folgt die Funktion zum Buchen einer Zeiteinheit: `book`. Ihr wird eine Start- und eine Endzeit übergeben. Die Zeiten werden mit der `strtotime`-Funktion in einen **UNIX-Timestamp** umgewandelt und müssen daher den sehr flexiblen, aber leider englischsprachigen Zeit- und Datumsformatanweisungen entsprechen. Für einfache Datumsberechnungen sind Timestamps sehr praktisch, da es sich um Integerzahlen handelt, auf die die bekannten mathematischen Funktionen (`>, <, ==, ...`) angewendet werden können.

> Ist eine der beiden Zeitangaben leer, wird ein Fehler, genauer gesagt eine Exception, ausgelöst.

Der Fehler ist vom Typ DateFormatException, das ist eine Subklasse der Exception-Klasse, die etwas später besprochen wird. Die Ausführung der Funktion wird nach dem Erzeugen der Exception abgebrochen.

Die Umwandlung der String-Variablen in Timestamps mit `strtotime` gibt -1 zurück, wenn das Format nicht als gültiges Datum erkannt wurde. In diesem Fall wird erneut eine DateFormatException erzeugt. Nur wenn zwei gültige Timestamps vorhanden sind, werden diese mit der IP-Adresse des Computers, von dem aus das Programm verwendet wird (`$_SERVER["REMOTE_ADDR"]`), einem neuen Objekt vom Typ Reserved übergeben. Diese Klasse überprüft im Konstruktor, ob es sich um eine gültige Zeitspanne handelt. Sollte die Zeitspanne nicht gültig sein, wird in der Reserved-Klasse eine Exception erzeugt, die an die aufrufende Funktion weitergeleitet wird und den Ablauf ebenso abbricht.

```
public function book($s, $e) {
if ($s == '' || $e == '')
throw new DateFormatException("Leeres Datum.");
$start = strtotime($s);
$end = strtotime($e);
$user = $_SERVER["REMOTE_ADDR"];
if ($start == -1 || $end == -1)
```

```
throw new DateFormatException("Zeitumwandlungsfehler");
$res = new Reserved($start, $end, $user);
if ($this->isBooked($start,$end))
throw new Exception("{$this->name} ist zu dieser Zeit".
"bereits belegt.");
$this->reservations[$start] = $res;
}
```

Bevor das neue Reserved Objekt zu dem bestehenden Objekt hinzugefügt wird, muss natürlich noch überprüft werden, ob der Zeitraum bereits belegt ist. Dabei kommt die `isBooked`-Funktion zum Einsatz.

```
public function isBooked($start, $end=NULL) {
// wenn der Aufruf nur mit einer Zeitangebe erfolgt, wird
// die Endzeit auf eine Sekunde nach der Startzeit gesetzt
if ($end == NULL) {
$end = $start + 1;
}
if (count($this->reservations) > 0) {
foreach ($this->reservations as $r) {
if ($r->isBusy($start, $end)) {
return TRUE;
}
}
}
return FALSE; // keine gebuchten Ressourcen gefunden
}
```

Diese Methode unterstützt das Überladen, d.h. in diesem Fall, dass sie mit einem oder zwei Parametern (einem Zeitpunkt oder einer Zeitspanne) aufgerufen werden kann. Die eigentliche Arbeit verrichtet eine **Schleife** über alle vorhandenen Reservierungen (`$this->reservations`). Für jedes Reservierungsobjekt wird geprüft, ob der Zeitraum bereits belegt ist (`$r->isBusy`). Sobald eine Überschneidung gefunden wird, gibt die Funktion TRUE zurück. Ist der Termin noch frei, retourniert die Funktion FALSE (die Reserved-Klasse wird gleich im Anschluss an diesen Abschnitt erklärt).

Neben einigen Hilfsfunktionen zur HTML-Ausgabe enthält die Resource-Klasse noch die Funktionen zum Speichern und zum Abrufen der Objekte. Wie bereits in der Einleitung erwähnt, werden die Objekte in der Session-Umgebung gespeichert. (Wenn Sie die Daten permanent speichern möchten, müssen Sie bei diesen Funktionen ansetzen. Anstatt in einer Session, könnten Sie die Daten auch in eine Datei oder eine Datenbank speichern. Der Dateizugriff ist meist problematisch, da Sie immer beachten müssen, dass es keine zeitgleichen Schreibzugriffe auf die Datei gibt. Datenbanken sind hier das Mittel der Wahl, da sie Ihnen diese Arbeit abnehmen.)

Die `save`-Funktion speichert das aktuelle Objekt (`$this`) in dem Session-Array

`$_SESSION['resource']`. Um jedes Objekte später wiederzufinden, wird als Array-Index die ID des Objekts verwendet.

```
public function save() {
$_SESSION['resource'][$this->id] = $this;
return TRUE;
}
public static function clearAll() {
unset($_SESSION['resource']);
return TRUE;
}
public static function getAll() {
if (array_key_exists('resource', $_SESSION)) {
return $_SESSION['resource'];
} else
return FALSE;
}
// Session-Funktion: gibt genau ein Objekt zurück
public static function getById($id) {
if (array_key_exists($id, $_SESSION['resource'])) {
return $_SESSION['resource'][$id];
} else {
throw new Exception("Id not found");
}
}
```

Die Funktionen `clearAll`, `getAll` und `getById` sind als statische Funktionen definiert. Das bedeutet, dass man keine Instanz der Klasse braucht, um sie aufzurufen, was bei diesen Funktionen sinnvoll ist. Die `getById`-Funktion überprüft, ob die gewünschte ID in der Session-Variable vorhanden ist. Sollte sie nicht vorhanden sein, wird eine Exception an die aufrufende Funktion weitergeleitet.

Die Klasse für die einzelnen Zeiteinträge: Reserved.php

Die Reserved-Klasse nimmt jeweils eine Startzeit, eine Endzeit und einen Benutzernamen im Konstruktor auf. Jedes Objekt repräsentiert einen gebuchten Eintrag.

```
class Reserved {
public $start;
public $end;
public $user;
// akzeptiert zwei Timestamps und einen String für den Benutzer
public function __construct($start, $end, $user) {
if ($start > $end) {
```

```
throw new Exception('Endzeit vor dem Beginn');
}
$this->start = $start;
$this->end = $end;
$this->user = $user;
if ($this->getDuration() > 3600*24*14) {
throw new Exception('Keine Buchungen über 14 Tage');
}
}
```

Der **Konstruktor** überprüft zwei Dinge: Erstens, ob die Startzeit nach der Endzeit liegt (das ist in jedem Fall eine ungültige Zeitspanne), und zweitens, ob die Zeitspanne mehr als zwei Wochen lang ist (3600*24*14 Sekunden). Diese zweite Kontrolle soll verhindern, dass Ressourcen über einen zu langen Zeitraum reserviert werden können. In beiden Fällen wird eine Exception erzeugt und das Objekt wird nicht erstellt.

```
public function getDuration() {
return $this->end - $this->start;
}
// gibt TRUE zurück, wenn das Objekt zwischen $from und $to
// bereits gebucht ist
public function isBusy($from, $to) {
if (($this->start <= $from && $this->end >= $from) ||
($this->start <= $to && $this->end >= $to) ||
($from <= $this->start && $to >= $this->end)) {
return TRUE;
} else {
return FALSE;
}
}
```

Die `isBusy`-Funktion wird ihrerseits mit zwei Timestamps aufgerufen und überprüft, ob die nachgefragten Zeiten mit der Zeit des Objekts kollidieren. Dazu wird überprüft, ob die Start- oder die Endzeit in den objekteigenen Zeitraum fallen. Die dritte Zeile in der `if`-Abfrage kontrolliert schließlich noch, ob der gefragte Zeitraum vor der Startzeit des Objekts beginnt und erst danach zu Ende ist.

Die Subklassen

Alle Subklassen von Resource müssen die `showCreate`-Methode überschreiben. In dieser Methode werden die Funktionalitäten einer Ressource eingegeben. In der Beamer-Klasse wird zum Beispiel die Auflösung des Geräts gespeichert und ob ein TV-Eingang vorhanden ist. Die Funktionen `showTextFeature` und `showBoolFeature` sind in der Basisklasse Resource implementiert und geben hauptsächlich **HTML-Formatierungen** aus. Der Konstruktor übergibt die ID an die Basisklasse, damit diese auch dort zur

Verfügung steht.

```
class Beamer extends Resource {
public $resolution = '';
public $tv_in = true;
public function __construct($id) {
parent::__construct($id);
}
public function showCreate() {
echo "<table border='1'>\n";
$this->showTextFeature('Name', 'name');
$this->showTextFeature('Auflösung', 'resolution');
$this->showBoolFeature('TV-Verbindung', 'tv_in');
echo "</table>\n";
echo "<input type='hidden' name='object' ".
"value='Beamer'>\n";
}
}
```

Die Klassen für Digitalkameras und Flipcharts funktionieren im Wesentlichen genauso wie die hier abgedruckte Beamer-Klasse. Einen Unterschied gibt es bei den Klassen zur Verwaltung von Räumen, also LectureRoom und MeetingRoom. Sie haben eine weitere gemeinsame Eigenschaft, nämlich die Anzahl von Sitzplätzen. Das Interface Room stellt sicher, dass alle Raum-Subklassen eine Sitzplatzfunktion bereitstellen. Alle Klassen, die das Room-Interface verwenden, müssen die Funktion `getSeats` überschreiben. Für die LectureRoom-Klasse sieht das zum Beispiel so aus:

```
class LectureRoom extends Resource implements Room {
public $microphone = FALSE;
public $seats = 0;
...
public function getSeats() {
return $this->seats;
}
```

Fehlerbehandlung: DateFormatException

Wenn ein Fehler bei der Zeiteingabe auftritt, wird eine Exception vom Typ DateFormatException erzeugt. Diese von der System-Klasse Exception abgeleitete Klasse formatiert die Fehlermeldung mit einer **Stylesheet-Anweisung** und kann außerdem den Link zur Dokumentation des Datumsformats ausgeben.

```
class DateFormatException extends Exception {
public function __construct($msg) {
parent::__construct($msg);
```

```
}
// zeigt den Link zur Datumsformat-Dokumentation
public function showHelp() {
echo "<p class='hint'><a ".
"href='http://www.gnu.org/software/tar/manual/".
"html_chapter/tar_7.html'>Zur Liste</a>
der ". "unterstützten Datumsformate.</p>\n";
}
public function __toString() {
return "<p class='error'>".$this->getMessage().
" (".__CLASS__.").</p>\n";
}
```

Durch die `__toString`-Funktion wird es möglich, die formatierte Fehlermeldung einfach mit `echo $e` auszugeben.

Die Verwendung der Klassen

Der Ablauf der Anwendung wird von drei Scripts gesteuert: index.php, book.php und add.php, wobei add.php nur für das Hinzufügen neuer Ressourcen gebraucht wird. Alle Scripts inkludieren in der ersten Zeile die Hilfsdatei inc/main.php. Außer den Funktionen zum Erstellen eines HTML-Kopfteils (html_start) und dem Ende der HTML-Datei (`html_end`) wird hier die Session gestartet. Die **PHP-5-Funktion** `__autoload` stellt schließlich sicher, dass alle angeforderten Klassen mit `require_-once` geladen werden. Wenn Sie eine neue Klasse zum System hinzufügen und sie entsprechend benennen (Klassenname in Kleinbuchstaben), wird sie automatisch von `__autoload` gefunden.

```
<?php // Beispieldatei: booking/inc/main.php
session_start();
function __autoload($class_name) {
require_once(strtolower(dirname(__FILE__)."/$class_name.php"));
}
```

Da die Dateien aller Klassen in Kleinbuchstaben gespeichert sind, wird bei dem `require_once`-Aufruf die Zeichenkette mit `strtolower` umgewandelt. Durch `dir-name(__FILE__)` wird sichergestellt, dass der korrekte Pfad zu den Klassen vorangestellt wird.
Info
Die `strtolower`-Funktion arbeitet abhängig von den verwendeten Locale-Einstellungen. Wenn Sie Umlaute in Ihren Dateinamen verwenden (generell eine schlechte Idee) und Locale nicht entsprechend eingestellt ist, wandelt `strtolower` die Umlaute nicht um.

Neue Ressourcen hinzufügen: add.php

Die Datei add.php arbeitet in drei Schritten: Schritt eins ist die Ausgabe einer Liste von verfügbaren Typen,

also aller Subklassen von Resource. Nachdem ein Typ ausgewählt wurde, wird im zweiten Schritt das Formular zur Eingabe der Daten für ein neues Objekt angezeigt (die `showCreate`-Funktion der jeweiligen Klasse). Der dritte Schritt ist das Erzeugen des neuen Objekts.

```
<select name="resource">
<?
// Schritt 1: Anzeigen der verfügbaren Klassen
foreach (Resource::$valid as $k=>$v) {
echo "<option value='$k'>$v</option>\n";
}
?>
```

Als `value` der Auswahlliste `resource` wird im ersten Formular der Name der Klasse übergeben (das ist der Schlüssel aus dem `$valid-Array` in der Resource-Klasse, z.B. LectureRoom). Im zweiten Schritt wird aus diesem Klassennamen ein neues Objekt (`$r`) erzeugt. Durch diesen kleinen Trick kann ein Dummy-Objekt mit der ID 0 für den `showCreate`-Aufruf verwendet werden.

```
// Schritt 2: Eingabeformular für die neue Klasse
if ($send != '') { // überprüft, von welchem Formular die
// Anfrage kommt
$r = new $res(0);
html_start("Add a new resource");
echo "<h2>Hinzufügen einer Ressource: $res</h2>\n";
printf("<form method='GET' action='%s'>\n",$_SERVER["PHP_SELF"]);
$r->showCreate();
echo "<input type='submit' name='add' value='Hinzufügen'>\n";
html_end();
exit();
}
```

Nach dem neuerlichen Abschicken der Seite wird das echte Objekt (`$o`) erzeugt. Als ID bekommt das Objekt den aktuellen UNIX-Timestamp zugewiesen (Sie können also in einer Sekunde nicht zwei Objekte erzeugen). Alle vom Formular übergebenen Werte (`$_GET`) werden in dem neuen Objekt als Member-Variablen gespeichert (`$o->$k = $v`).

```
// Schritt 3: neues Objekt erzeugen
if ($add != '') {
$o = new $object(time());
foreach($_GET as $k=>$v) {
$o->$k = $v;
}
```

Ist der Name der neuen Ressource nicht eingegeben worden (`$o->name == ''`), so wird die ID-Nummer als Name verwendet. Da bei dem Speicher-Aufruf (`$o->save`) eine Exception ausgelöst werden kann, ist diese Anweisung in einem `try-catch`-Block geklammert. Im `catch`-Abschnitt wird bei Bedarf die entsprechende Fehlermeldung ausgegeben.

```
try {
if ($o->name == '')
$o->name = $o->id;
$o->save();
html_start("Neue Ressource hinzugefügt");
echo "<h2>Neue Ressource hinzugefügt: ID: ".
"{$o->id}</h2>\n";
echo "<a href='book.php?id={$o->id}'>Termine ".
"reservieren</a>.\n";
html_end();
exit();
} catch (Exception $e) {
html_start("Fehler beim Hinzufügen");
echo "<h2>Es ist ein Fehler aufgetreten: ".
$e->getMessage()."</h2>\n";
}
}
```

Nach erfolgreicher Erstellung des neuen Objekts wird gleich der Link zum Buchen einer Zeiteinheit für diese Ressource angezeigt.

Die Startseite: index.php

index.php ist die Startseite des Beispiels (siehe Abbildung 3).

```
<?php // Beispieldatei: booking/index.php
require_once(dirname(__FILE__)."/inc/main.php");
$elements = Resource::getAll();
if ($elements == FALSE) {
header("Location: add.php");
exit();
}
html_start("Reservierungssystem");
```

Wenn der Funktionsaufruf `Resource::getAll()` FALSE zurückgibt, sind keine Ressourcen vorhanden und man wird via `header`-Funktion auf die Eingabeseite add.php umgeleitet. Andernfalls wird in einer `foreach`-Schleife die Liste der verfügbaren Ressourcen angezeigt (vergleiche Abbildung 3).

```
<?php
foreach($elements as $r) {
echo "<option value='{$r->id}'>{$r->name}";
if ($r instanceof Room) {
printf(" (%s Plätze)", $r->getSeats());
}
echo "</option>\n";
}
?>
```

Interessant in dieser Schleife ist der `instanceof`-Operator. Implementiert eine Klasse das Room-Interface, ist sie automatisch eine Instanz von Room. Dementsprechend wird für alle Räume angezeigt, wie viele Sitzplätze vorhanden sind.

Ressourcen reservieren: book.php

Der für den Anwender interessanteste Teil ist das Reservieren einer Ressource. Das book.php-Script muss mit einer ID aufgerufen werden (`$id`), andernfalls wird man gleich auf die Startseite weitergeleitet.

```
if ($id == '' || $id == FALSE) {
header("Location: index.php");
}
try {
$r = Resource::getById($id);
} catch (Exception $e) {
html_start();
echo "<p class='error'>Die gewünschte Ressource konnte ".
"nicht gefunden werden\n";
html_end();
exit();
}
```

Das gewünschte Objekt wird von der statischen Methode `getById` aus der Resource-Klasse geholt. Ist die ID nicht vorhanden, wird eine **Exception** erzeugt, auf die mit der entsprechenden Fehlermeldung reagiert wird. Anschließend wird eine Hilfsfunktion aus der Resource-Klasse aufgerufen, die die beiden Texteingabefelder für den Start- und den Endzeitpunkt anzeigt (`$r->showAddReservation`). Wurde das Formular zur Buchung abgeschickt (`$reserve` hat einen Wert), wird versucht, die gewünschte Zeitspanne für die Ressource zu buchen. Andernfalls wird der ganze `if`-Block übersprungen und die Funktion `$r->showBooked` gibt eine List der vorhandenen Reservierungen aus.

```
$r->showAddReservation();
if ($reserve != '') {
$from = array_item($_GET, 'from');
```

```
$to = array_item($_GET, 'to');
try {
$r->book($from, $to);
$r->save();
} catch (DateFormatException $e) {
echo $e;
$e->showHelp();
} catch (Exception $e) {
printf("<p class='error'>%s</p>\n", $e->getMessage());
}
}
$r->showBooked();
```

Die eigentliche Buchung führt die Funktion `$r->book` aus der Resource-Klasse durch. Diese kann, wie oben erwähnt, unterschiedliche Exceptions erzeugen, je nachdem, was schief gegangen ist (ein Datumsproblem oder die Ressource ist bereits belegt). In book.php werden die Fehler für Datumskonvertierungen (**DateFormatException**) gesondert behandelt, denn bei ihnen wird zusätzlich der Link zu den korrekten Datumsformaten angezeigt (`$e->showHelp`). Alle anderen Exceptions werden in der zweiten Schleife abgefangen.

Zwei weitere kleine **PHP-Scripts** sind noch vorhanden: del.php und reset.php. Das `reset`-Script ist sehr einfach: Es ruft die statische `clearAll`-Funktion der Resource-Klasse auf und leitet den Anwender auf die Startseite zurück. del.php löscht einen Eintrag aus der Liste der gebuchten Zeiten. Auch dazu wird die entsprechende Funktion in der Resource-Klasse verwendet (`deleteReservation`).

Info

Das Beispiel war wirklich nur dazu gedacht, Ihnen einige Grundzüge von **OOP** näherzubringen. In dieser Form bietet es keine praktischen Einsatzmöglichkeiten. Daher wurde auch darauf verzichtet, die Eingaben streng zu prüfen, um etwaige Angreifer abzublocken. Der Umgang mit Textfeldern und GET-Variablen in einem öffentlichen Webformular ist sehr heikel und sollte gut abgesichert werden.

Ein wesentliches Anliegen einer Website ist die Möglichkeit, auf Daten und Informationen zugreifen und sie dauerhaft sichern zu können. Dabei kann es sich um Nutzerbewegungen handeln, die in Dateien protokolliert werden sollen oder um Daten, die in ein Formular eingegeben wurden.

Prinzipiell können Sie für diesen Zweck zwei Methoden einsetzen: Sie sammeln die Daten in Textdateien oder Sie verwenden Datenbanken. Der Vorteil bei der Verwendung von Textdateien liegt unmittelbar auf der Hand: Sie brauchen keine Kenntnisse über Datenbanken, ein Thema, das bekanntlich nicht ganz einfach ist oder zumindest den Ruf hat, nicht ganz einfach zu sein! Bevor wir also daran gehen, beispielhaft auch mit einer Datenbank zu arbeiten, wollen wir in diesem Kapitel klären, wie Sie mit **PHP** eine Textdatei anlegen, Daten in die Datei schreiben und diese Daten auslesen.

Obwohl Datenbanken zweifellos leistungsstärker als Textdateien sind, werden Sie sehen,

dass sich auch mit Dateien eine Menge machen lässt. Elementar sind zunächst die zwei Prozesse: in eine Datei zu schreiben und Daten auszulesen. Wir wollen diese Prozesse nicht nur abstrakt erläutern, sondern nach der Erklärung einiger Grundlagen an einem Beispiel durchspielen. Dazu wird eine Seite kreiert, die ein Gästebuch anbietet, dem User also die Möglichkeit gibt, einen Kommentar einzugeben. Diese Informationen werden dann zum einen in einer TXT-Datei gesammelt und sind zum anderen für andere Besucher einsehbar.

Daten in Textdateien sichern

Die Aktion, Daten in eine Datei zu schreiben, besteht aus drei Schritten:

- Zunächst wird eine Datei geöffnet, bzw. angelegt.
- Dann werden die Daten in die Datei geschrieben.
- Schließlich wird die Datei geschlossen.

PHP bietet Funktionen, die diese Aufgabe leisten. Sie sind relativ einfach nachzuvollziehen.

Zum Anlegen einer Datei benutzen Sie die Funktion `fopen()`.

Die Klammer nimmt als Argumente den Namen der Datei bzw. den Dateipfad auf und – und dies ist sehr wichtig – den Modus. Mit dem Modus bestimmen Sie, wie die Datei geöffnet werden soll und an welcher Stelle in der Datei der Dateizeiger (quasi der Cursor) steht, oder – anders ausgedrückt – wo der Startpunkt für das Lesen bzw. Schreiben in der Datei ist. Dabei können Sie zwischen diversen Varianten wählen. Die folgende Tabelle gibt einen kurzen Überblick über die verschiedenen Möglichkeiten:

Modus	Bedeutung
r	erlaubt das Auslesen einer Datei
w	schreibt in eine Datei und legt sie an, sofern sie noch nicht existiert
a	hängt neue Daten an das Ende einer Datei und legt eine Datei an, sofern sie nicht existiert
r+	schreibt Daten in eine Datei und liest die Daten aus
w+	schreibt Daten in eine Datei und liest die Daten aus, kreiert eine Datei, wenn sie nicht existiert, aber wirft vor dem Schreiben Daten raus, sofern die Datei existiert. Daten werden also überschrieben.
a+	schreibt Daten in eine Datei und liest die Daten aus, kreiert eine Datei, wenn sie nicht existiert. Neue Daten werden an das Ende der Datei gehängt.

Der Modus, der eigentlich alles zulässt, ist `a+`. Bei `r` oder `r+` muss die Datei, in die geschrieben oder aus der gelesen werden soll, bereits existieren. Mit allen anderen Modi wird sie gegebenenfalls neu angelegt. Vorsicht ist geboten bei `w+`: der Inhalt der Datei wird ohne Nachfrage überschrieben.
Info

```
int fopen(string dateiname, string modus [, int use_include_path])
```

Diese Funktion öffnet eine Datei oder URL. Sofern sie noch nicht existiert, wird die Datei bei einigen Modi auch erzeugt. Der Modus gibt an, auf welche Weise die Datei geöffnet wird.

Wenn Sie mit `fopen()` eine Datei anlegen, erzeugen Sie einen sogenannten Dateihandle. Dieser Handle wird von PHP als Verweis auf die Textdatei verwendet. Für alle weiteren Operationen arbeiten Sie dann mit diesem Handle. Dies ist wesentlich praktikabler als immer wieder den Pfad einer Datei angeben zu müssen. Die Funktion, mit der Sie Daten an die Position des Dateizeigers schreiben, lautet: `fputs()`.

Die Zeilen zum Öffnen (bzw. Anlegen) einer Datei und zum Schreiben von Daten in diese Datei sehen dann generell so aus (statt Modus eine der Abkürzungen aus der obigen Tabelle, also beispielsweise a):

```
$handle=fopen(Dateiname.txt, Modus);
fputs($handle, die zu schreibenden Daten);
fclose($handle);
```

Info

```
int fputs(int filehandler, string daten [, int length])
```

Mit dieser Funktion werden Daten (daten) an die aktuelle Position des Dateizeigers in eine Datei geschrieben. Gibt es keine Angabe zur Länge (`length`) des Strings, schreibt PHP den kompletten **String** in die Datei.

Diesen Prozess wollen wir anhand der Aufgabe, ein Gästebuch zu erstellen, im nächsten Abschnitt konkretisieren. Die Einträge der Gäste sollen zunächst in eine separate Textdatei geschrieben werden. Danach sorgen Sie durch die Erweiterung des Scripts dafür, dass die Besucher der Seite die Einträge des Gästebuchs einsehen können.

Vorüberlegungen zum Script

Wir wollen das **Script** zunächst soweit entwickeln, dass die Textdatei erzeugt wird, die die aus dem Formular übergebenen Daten sichert.

Im nächsten Schritt, d.h. im erweiterten Script, werden wir dann zu der Realisierung der Aufgabe kommen,

die Daten auszulesen, sodass sie für den User einsehbar sind.

> Für die erste Aufgabe brauchen Sie die oben vorgestellten Funktionen und eine Reihe weiterer Funktionen,

die vor allem dazu dienen, die Einträge, die übernommen werden, zu bearbeiten, sodass die Textdatei keine störenden Elemente enthalten wird.

Zeichen und/oder Text ersetzen

Kommen wir zunächst zu der Funktion `str_replace()`.

Hiermit lassen Sie einen Text bzw. eine **Zeichenkette** nach dem Vorkommen einer anderen Zeichenkette durchsuchen und durch einen anderen String ersetzen. Als Argumente werden in die Klammer gesetzt:

```
str_replace('zu ersetzende Zeichenkette', 'Ersatzzeichenkette', 'zu durchsuchende Zeichenkette');
```

Diese Aktion klingt sicherlich irritierend, aber der Sinn der Sache wird verständlich werden, wenn wir das Script erläutern. In unserem Beispiel hängt der Einsatz dieser Funktion damit zusammen, dass bestimmte Zeichen nicht in den Daten, die durch das Formular übergeben werden, enthalten sein dürfen und sie deswegen ersetzt werden, sofern sie vorkommen. Wir können an dieser Stelle auch verallgemeinert festhalten, dass insbesondere bei Text, der aus irgendeiner Quelle übernommen wird, auf störende (Sonder)zeichen geachtet und der Text in der Regel manipuliert werden muss.
Info

```
string str_replace(string zeichen, string ersatzzeichenkette, string inhalt)
```

Die Funktion ersetzt alle Vorkommen eines Strings durch einen anderen (von zeichen durch ersatzzeichenkette in inhalt)

Suchen und Ersetzen

Eine Spielart der Funktion `replace()` ist übrigens `ereg_replace()`. Damit wird nach dem zu ersetzenden Text/Zeichen gesucht und dann durch den/das gewünschte ersetzt. Ein kleines Beispiel, in dem ein Text und ein Zeichen ersetzt werden. Es gibt beispielsweise die folgenden Textzeilen

In PHP muss man darauf achten,
wie texte übernommen werden.

In PHP sieht dieser Text so aus:

```
In PHP muss man darauf achten, \n wie texte übernommen werden.
```

Wir hätten nun gern zwei Dinge verbessert. Zum einen sollen die PHP-Zeichen für Zeilenumbrüche ersetzt werden durch „richtige“ Zeilenumbrüche, die in **HTML** zu sehen sind, sodass die Textformatierung erhalten bleibt, und zum anderen soll aus texte Texte werden.

Wir speichern den obigen Text in einer Variablen

```
$testtext = In PHP muss man darauf achten,\n wie texte übernommen werden.
```

und setzen dann zunächst die Funktion `ereg_replace()` ein:

```
$testtext=ereg_replace(texte, Texte,$testtext);
echo $testtext;
```

Schauen Sie sich die Datei in Ihrem Browser an:

Durch das `ereg_replace` wird texte durch Texte ersetzt. Im nächsten Schritt kümmern Sie sich um den Austausch der Zeilenumbruchzeichen. Dazu setzen Sie die Funktion `nl2br()` ein, und schreiben

```
$testtext=nl2br($testtext);
echo $testtext;
```

Info

```
string ereg_replace(string suchennach, string ersatz, string inhalt)
```

Die Funktion sucht in einer Zeichenkette (inhalt) nach einer anderen Zeichenkette (suchennach) und ersetzt diese (ersatz).

Betrachten Sie die Datei erneut im Browser. Der Text müsste nun das gewünschte Aussehen haben. Das Bild 1 zeigt das kleine Script und die Ausgabe im Browser.

Info

```
string nl2br (string string)
```

Die Funktion wandelt aus einem String sämtliche Zeilenumbrüche in die HTML-Entsprechung (<br>) um. Der Befehl funktioniert bei Installation von PHP auf Windows-Rechnern nicht immer einwandfrei!

Leerstellen und HTML-Zeichen entfernen

Außerdem kommt in dem Script für das Gästebuch wieder die Funktion `trim()` zum Einsatz, die wir auch schon im Zusammenhang mit dem Kontaktformular verwendet haben. Sie sorgt dafür, dass überflüssige Leerzeichen aus einer Zeichenkette entfernt werden. Des Weiteren werden wir die Funktion `strip_tags()` verwenden. Damit lassen sich HTML-Tags und **PHP-Zeichen** aus Zeichenketten entfernen.

> Für die Darstellung des Formulars benutzen Sie die üblichen HTML-Tags.

Da die Besucher der Website in der Lage sein sollen, einen Kommentar einzutippen (und nicht nur einen Namen oder Ähnliches), brauchen Sie in dem Fall auch den Typ Textarea für mehrzeilige Eingabefelder. Die Seite könnte – schlicht und ohne weiteres Layout – aussehen wie das Bild 2.

Sie können das Script mit dem PHP-Teil beginnen und dann das **HTML-Formular** anlegen. Zunächst setzen Sie die eben kurz erläuterten Funktionen zur Manipulation des in die Felder eingegebenen Textes ein, dann bauen Sie die Fehlermeldungen zusammen für den Fall, dass das Formular nicht korrekt abgeschickt wird, danach folgen die Befehle für das Anlegen der Datei und das Schreiben der Daten in diese Datei. Als Letztes schreiben Sie den HTML-Teil für das Formular.

Das Script für das Gästebuch und die Textdatei

Der Code zeigt das Script für das Gästebuch. Es enthält Fehlermeldungen, Befehle zum Bearbeiten der Texteingaben und für das Anlegen der Textdatei sowie das Schreiben der Daten in die Datei und die HTML-Elemente für das Formular. Gestalterische Elemente beim Formular haben wir wieder weitgehend ausgespart, da wir uns auf die PHP-Codes konzentrieren.

```
<?php
if($sent==1)
{
$t1=chr(10);
$t2=chr(13);
$name=str_replace('~','',$name);
$betreff=str_replace('~','',$betreff);
$message=str_replace('~','',$message);
$name=trim($name);
$betreff=trim($betreff);
$message=trim($message);
$name=strip_tags($name);
$betreff=strip_tags($betreff);
$message=strip_tags($message);
If(!$name){$fehler="Bitte geben Sie einen Namen ein
```

```
<br>";}
If(!$betreff){$fehler=$fehler."Bitte geben Sie den Betreff an
<br>";}
If(!$message){$fehler=$fehler."Bitte geben Sie eine Nachricht ein
<br>";}
if($fehler){$fehler="<font color=red>
<h4>".$fehler."</h4></font>";}
}
if($name AND $betreff AND $message)
//Formular wurde ausgefüllt
{
$message=str_replace($t1,'<br>',$message);
$message=str_replace($t2,'<br>',$message);
IF(file_exists('gast.txt')){$ausgabe="\n";}
$comment=fopen('gast.txt','a');
$ausgabe=$ausgabe.$name."~".$betreff."~".$message;
fputs($comment,$ausgabe);
fclose($comment);
$name="";
$betreff="";
$message="";
}
?>
<html>
<head>
<title>Gästebuch</title>
</head>
<body>
<h3>Unser Gästebuch</h3>
<?php echo $fehler; ?>
<form action='<?php echo $PHP_SELF; ?>' method='post'>
<input type='hidden' name='sent' value=1>
<p>Ihr Name:
<br>
<input type='text' name='name' size='30' value='<?php echo $name;
?>'>
<br>
Betreff:<br>
<input type='text' name='betreff' size='30' value='<?php echo $betreff;
?>'>
<br>
Ihre Nachricht:
<br>
<textarea name='message' rows='10' cols='30' wrap=virtual>
<?php echo $message; ?>
</textarea>
```

```
<br>
<input type=submit value=abschicken>
</form>
</body>
</html>
```

Erläuterung des Scripts

Wie auch schon bei bisherigen Scripts setzen Sie mit

```
if($sent==1)
{
```

am Anfang des Scripts eine `if`-Bedingung ein, die prüft, ob das Formular abgeschickt wurde. Die Anweisungen in dem Block werden nur ausgeführt, wenn die Prüfung `true` ergibt.

Zum Verständnis der Definition der nächsten beiden Zeilen, in denen die Funktion `chr()` verwendet wird, müssen Sie sich klar machen, wie der Text in den Textbereich (`<textarea>`) des Formulars eingegeben wird. Hier erfolgen Zeilenumbrüche, die in der Darstellung des Textes in der Textdatei nicht enthalten sein dürfen, da dadurch die Datensätze verfälscht werden würden, denn mit einem Absatz beginnt ja eine neue Zeile mit dem nächsten Datensatz.

Diesem Problem wollen wir mit der Funktion `chr()` zu Leibe rücken, mit der man sich das ASCII-Zeichen (8-Bit-Zeichensatz, bei dem jeder Zahl von 0-255 ein Zeichen zugewiesen ist) zu einer angegebenen Nummer ausgeben lassen kann. So gibt `chr(10)` das Zeichen für den ASCII-Wert 10 aus und der ASCII-Wert 10 (bzw. 13) ist der Zeilenumbruch. Diese Werte werden zunächst in den beiden Variablen `$t1` und `$t2` gespeichert:

```
$t1=chr(10);
$t2=chr(13);
```

Info
`string chr(int ascii)`

Die Funktion gibt das ASCII-Zeichen zu einer angegebenen Nummer aus.

Weiter unten im Script werden wir die Werte dieser Variablen ersetzen lassen, um damit die unerwünschten Zeilenumbrüche loszuwerden.
Info
Seit kurzer Zeit klimpert der Euro in den Kassen. Aus aktuellem Anlass deswegen der folgende Tipp: Schreiben Sie doch einfach einmal `echo chr(128);` in ein Script. Was zeigt der Browser? Voilà, das Eurozeichen!

Als Nächstes verwenden wir die Funktion `str_replace`, und zwar dafür, eventuelle Tilden (~) aus dem eingegebenen Text zu entfernen. Warum wir dies machen, erklärt sich im Prinzip erst weiter unten im Script. Dort sehen Sie, dass wir die Tilde als Trennzeichen beim Schreiben der Daten in die Textdatei benutzen. Mit anderen Worten: zwischen Name, Betreff und Message steht jeweils eine Tilde. Daher darf dieses Zeichen nicht in eingegebenen Formulardaten vorkommen und wir lassen es ersetzen, falls es doch vorkommt. Es wird durch „Nichts“ ersetzt, d. h. durch einen Leerstring. Diese Funktion müssen wir auf alle drei Texteingaben anwenden. Die Klammer enthält als Argumente das zu ersetzende Zeichen, den Ersatzstring und den zu durchsuchenden String. Dass die Tilde durch „Nichts“ ersetzt wird, erreichen wir mit zwei direkt aufeinanderfolgenden Anführungszeichen:

```
$name=str_replace('~','',$name);
$betreff=str_replace('~','',$betreff);
$message=str_replace('~','',$message);
```

Danach werden mit der Funktion `trim()` eventuelle Leerzeichen aus den Texteingaben entfernt und zwar am Anfang und am Ende. Die Variablen $name, $betreff und $message erhalten also neue Werte.

```
$name=trim($name);
$betreff=trim($betreff);
$message=trim($message);
```

Zu guter Letzt entfernen Sie eventuelle HTML-tags aus den Eingaben. Dies empfiehlt sich, da Sie so verhindern, dass ein User JavaScripte zum Umleiten der Seite eingibt.

```
$name=strip_tags($name);
$betreff=strip_tags($betreff);
$message=strip_tags($message);
```

Info
`string strip_tags(string zeichenkette [,string erlaubte_tags])`

Die Funktion entfernt aus zeichenkette alle **HTML**– und **PHP-Tags**, außer die in `erlaubte_tags` aufgelisteten.

Die Fehlermeldung

Damit auf alle denkbaren Fälle fehlender Eingaben reagiert wird, aber dennoch nicht zu viele `if`-Anweisungen geschrieben werden müssen, haben wir versucht, die Fehlermeldung durch Erweiterung der Variablen möglichst kompakt zusammenzubauen.

```
If(!$name){$fehler="Bitte geben Sie einen Namen ein <br>";}
If(!$betreff){$fehler=$fehler."Bitte geben Sie den Betreff an <br>";}
If(!$message){$fehler=$fehler."Bitte geben Sie eine Nachricht ein<br>";}
If($fehler){$fehler="<font color=red><h4>".$fehler."</h4></font>";}
```

Zunächst benutzen Sie in der `if`-Bedingung wieder das Ausrufezeichen, um die Eingabe zu negieren, sodass als Prüfergebnis der Bedingung true ausgegeben und die Anweisung ausgeführt wird, wenn keine Eingabe in dem Feld vorgenommen wurde. Im ersten Anweisungs-Block wird der auszugebende Text in der Variablen $fehler gespeichert. In der nächsten `if`-Anweisung weisen Sie mit $fehler=$fehler. den Wert der Ursprungsvariablen erneut zu und erweitern diesen Wert durch das Setzen des Punktes mit dem neuen Wert. Ergibt die Prüfung der ersten und zweiten Bedingung `true`, werden also die beiden Meldungen als Wert in der Variablen $fehler gespeichert. Analog geben Sie dann die `if`-Anweisung für die letzte Fehlermeldung ein, Sie erweitern also erneut.

> Zu guter Letzt kümmern Sie sich um die Formatierung der Fehlermeldung.

In der Anweisung schreiben Sie, dass der in der Variablen enthaltene Wert die Farbe rot annimmt und als Überschrift der Ebene 4 formatiert wird.

Danach folgt die geschweifte Klammer, die die erste `if`-Anweisung (die Prüfung, ob das Formular abgeschickt wurde) schließt.

Dann verzweigt das Script in die nächste `if`-Bedingung, die prüft, ob alle Felder ausgefüllt wurden. Dazu benutzen Sie AND-Verknüpfungen:

```
if($name AND $betreff AND $message)
{
```

Wird als Prüfergebnis `true` ausgegeben, sollen als Erstes die Zeilenumbrüche aus der Eingabe im Textbereich entfernt und ersetzt werden. Wir sprachen das Problem bereits weiter oben an: Der Browser schickt die in der `<textarea>` befindlichen Zeilenumbrüche mit. Da die Texte in einer Textdatei gespeichert werden sollen, und zwar jeder Datensatz in einer Zeile, dürfen keine Zeilenumbrüche (außer den bewussten am Ende eines Datensatzes) in die Textdatei geschrieben werden, da der Datensatz sonst nicht komplett mit einer Zeile ausgelesen werden kann. In der Variablen $t1 bzw. $t2 sind die Zeichen für den Zeilenumbruch gespeichert (weiter oben zugewiesen über die Funktion `chr(10)` bzw. `chr(13)`). Nun ersetzen wir die Zeilenumbrüche durch `<br>`, (dieses Zeichen stört nicht, sondern sorgt nur dafür, dass die Zeile im Browser umgebrochen wird), indem wir die Variable $message mit dem neuen Wert überschreiben. Die beiden Zeilen für die beschriebene Aufgabe lauten:

```
$message=str_replace($t1,'<br>',$message);
```

```
$message=str_replace($t2,'<br>',$message);
```

Erzeugen der Textdatei

In den nächsten fünf Zeilen geht es nun um die Textdatei zur Sicherung der Formulardaten. Hier müssen wir etwas ausholen, denn trotz der Erklärung weiter oben ist manches vermutlich nicht unmittelbar einleuchtend. Verständlich dürfte die zweite Zeile sein:

```
$comment=fopen('gast.txt','a');
```

Hier wird mit `fopen` eine Datei erzeugt, diese Datei erhält den Namen gast.txt und als Modus wurde a festlegt, d.h. es wird eine Datei kreiert, sofern sie noch nicht existiert und die Daten werden an das Ende angehängt. Dies wird in dem Dateihandle ($comment) gespeichert. Die Zeile darunter definiert eine Variable für die Daten, die in die Datei geschrieben werden sollen. Wir möchten aus dem Formular den Nachnamen, den Betreff und die Nachricht übergeben lassen (die Variablen korrespondieren mit den Namen, die im HTML-Teil in den Formularfeldern vergeben wurden bzw. noch vergeben werden):

```
$ausgabe=$ausgabe.$lastname."~".$betreff."~".$message;
```

Nun sind noch die verwendeten Tilden erklärungsbedürftig. Um sie zu verstehen stellen Sie sich am besten vor, in welcher Form die Daten in die Datei geschrieben werden sollen. Sicherlich nicht ohne Punkt und Komma, sondern jeweils mit einem Trennzeichen zwischen den einzelnen Elementen. Sie brauchen also ein Trennzeichen. Da fast alle Zeichen in den Texteingaben der User vorkommen können und für den Text wichtig sind, muss man ein Zeichen finden, das sozusagen noch „frei“ ist. Deswegen haben wir hier die Tilde – als String in Anführungszeichen – eingesetzt. Sie trennt die Elemente. Danach folgt die Funktion `fputs`. Die Klammer nimmt den Dateihandle auf und die Variable, deren Wert die Daten bilden, die in die Datei geschrieben werden sollen:

```
fputs($comment,$ausgabe);
```

Zu guter Letzt wird der Prozess mit `fclose` geschlossen.

Nun bleibt noch die erste Zeile dieses Blocks, also die `if`-Bedingung über `fopen` zu erklären.

```
IF(file_exists('gast.txt')){$ausgabe="\n";}
```

Stellen Sie sich dazu die Textdatei vor. Wenn der erste Datensatz in eine Zeile geschrieben ist, soll der nächste Datensatz in einer neuen Zeile stehen. Deshalb braucht man am Anfang ab dem zweiten Datensatz ein `\n` für einen Zeilenvorschub. Dadurch wird dieser Datensatz in die nächste Zeile geschrieben.

Wir stoßen hier in philosophische Gefilde, denn Sie müssen bedenken: der erste Datensatz erzeugt automatisch die Datei, folglich gibt es bereits einen Datensatz, wenn es die Datei gibt. Daher die `if`-Bedingung mit der Funktion `file_exists()`: Wenn es die Datei bereits gibt (die Prüfung TRUE ergibt), soll vor dem Datensatz ein `\n` gesetzt werden. Ansonsten (also der allererste Datensatz) wird `\n` nicht gebraucht.

Der nächste Dreizeiler ist einfach aber wirkungsvoll:

```
$name="";
$betreff="";
$message="";
```

Sie löschen mit der Zuweisung von „Leerstrings“ die in den Feldern eingegebenen Werte (sie sind bereits in der Textdatei gespeichert), damit das Formular wieder leer angezeigt wird.

Jetzt schließen Sie den Block der `if`-Bedingung für das ausgefüllte Formular und zu guter Letzt setzen Sie das schließenden PHP-Zeichen.

Das Formular

Dann beginnt der HTML-Teil mit dem üblichen Header. Bevor Sie den `<form>`-Tag für das Formular öffnen, sorgen Sie dafür, dass eine Meldung erscheint, sofern eines oder mehrere der Felder nicht ausgefüllt wurden. Dies ist ein einfacher Einzeiler, da die entsprechende Werte in der Variablen $fehler gespeichert wurden:

```
<?php echo $fehler; ?>
```

Mit dem Attribut `action` verweisen Sie entweder auf die aktuelle Datei oder Sie verwenden die **PHP-Variable** $PHP_SELF. Denken Sie aber daran, für die Variable PHP zu öffnen und zu schließen. Mit den Formularelementen erzeugen Sie die üblichen Textfelder und einen mehrzeiligen Textbereich. Dies ist die Zeile:

```
<textarea name="message" rows='10' cols='30'></textarea>
```

Den Formularelementen weisen Sie am besten wieder Werte zu, nämlich die Ausgabe der Variablen, in der die Namen gespeichert sind. Dies bewirkt dann, dass die Eingaben in den Feldern stehen bleiben, wenn ein User das Formular unvollständig abgeschickt hat.

```
value='<?php echo $name; ?>
```

Info
Haben Sie sich beim Surfen schon häufig darüber geärgert, dass Sie bei Formulareingaben immer wieder von vorn anfangen müssen, sofern Sie das Formular mit einem Fehler abgeschickt haben? Genau diese Ärgernis umgehen Sie mit der Ausgabe der Werte der Variablen in den Value-Attributen. Wenn Sie diesen Schritt beherzigen, machen Sie es auch als Einsteiger besser als viele alte Hasen unter den Programmierern!

Nach dem schließenden `</form>`-Tag schließen Sie den `<body>` und HTML.

Speichern Sie das Dokument als PHP-Datei und rufen Sie es im Browser auf. Geben Sie ein paar Daten ein, und schicken Sie das Formular ab. Da wir keine Programmzeilen in das Script eingebaut haben, die die Inhalte der Felder im Browser ausgeben, erhalten Sie nach dem Abschicken des vollständig ausgefüllten Formulars einfach wieder ein Leerformular.

Die Textdatei überprüfen

Testen Sie nun, ob es geklappt hat, dass Daten in eine Datei geschrieben werden. Schicken Sie das ausgefüllte Formular also etliche Male hintereinander ab, damit Sie ein paar Daten generieren. Öffnen Sie dann den Explorer mit dem Ordner, in dem auch das Script gespeichert wurde und rufen Sie die Datei auf, die Sie mit `fopen` angelegt haben. (Im Beispiel war es gast.txt) Liegen Ihre Dateien bereits auf Ihrem Speicherplatz bei Ihrem Provider, verwenden Sie Ihr FTP-Programm zum Anzeigen der TXT-Datei. Die eben abgeschickten Daten müssten nun in der Textdatei untereinander aufgelistet werden. Das sieht so aus wie in Bild 6.

Auslesen der Gästebucheinträge

Wie auch im „wirklichen Leben“ ist ein elektronisches Gästebuch erst richtig interessant, wenn man nachschauen kann, was andere Gäste als Kommentar geschrieben haben. Die eingegebenen Informationen müssen also im Browser einsehbar sein. Diese Aufgabe lässt sich lösen mit dem so genannten Auslesen von Dateien. Dazu lernen Sie eine bisher nicht verwendete Funktion kennen:

```
readfile(); oder nur file();
```

Die Funktion `readfile()` liest eine Datei und schickt den Inhalt an die Standardausgabe, also im Regelfall an den Browser. In der Klammer steht der Dateiname. Auch `file()` liest eine Datei, aber die Daten werden zeilenweise in einem Array abgelegt. Jede Zeile bildet ein Element des Arrays. Da dies für die Gästebucheinträge Sinn macht, schreiben Sie also in das Script

```
$eintrag=file('gast.txt');
```

Info

```
int readfile(string filename [int use_include_path])
```

Die Funktion liest eine Datei aus und gibt sie im Browser aus.

Dann setzen Sie eine Variable mit dem Wert <br>. Mit der Ausgabe der Variablen sorgen Sie dafür, dass zwischen den einzelnen Datensätzen jeweils ein Umbruch erfolgt:

```
$ausgabe="<br>";
```

Die Variable $temp ist eine Hilfsvariable, mit der Sie die `for`-Schleife einfach definieren können. Mit der Funktion `count($eintrag)-1;` zählen Sie die Elemente des Arrays, und ziehen eins ab, weil $eintrag[0] ja das erste Element ist. Entsprechend ist das letzte Element: Anzahl -1.

Danach beginnen Sie die `for`-Schleife zu schreiben. Der Variablen $i wird der Wert der Variablen $temp zugewiesen (gezählte Elemente), dann wird die Bedingung geprüft, ob $i größer/gleich 0 ist. Ergibt die Prüfung `true`, fährt die Schleife fort mit der Ausführung des abschließenden Ausdrucks.

> Der neueste Eintrag steht jeweils am Ende der Datei und deswegen im letzten Element des Arrays.

Wenn der neueste Eintrag oben angezeigt werden soll, muss man das Array von hinten nach vorn durchlaufen. Deswegen wird $i nicht inkrementiert, sondern dekrementiert.

```
for($i=$temp;$i>=0;$i--)
```

In der Anweisung verwenden wir die Funktion `explode()`.

Mit dieser Funktion lassen sich Zeichenketten anhand von Trennzeichen in Teile zerlegen. Sie sehen, dass in der Klammer das Trennzeichen angegeben wird und das Array Eintrag[$i], also das Array mit allen gezählten Elementen. Folglich zerlegt explode einen Datensatz mit Hilfe der Tilde in die einzelnen Feldinhalte. Die einzelnen Feldinhalte werden als Elemente des neuen Arrays abgelegt ($element).

Die einzelnen Elemente sollen in Form einer Tabelle untereinander angezeigt werden. Deswegen deklarieren Sie eine neue Variable, der über die Variablen-Erweiterung mit Hilfe des Punktes der Wert von $element zugewiesen wird sowie der HTML-Tag zum Erstellen einer Tabelle.

```
$ausgabe.="<table>";
```

In der nächsten Zeile wird der Variablen die Zeile und Spalte zugewiesen sowie der Wert des ersten

Elementes des Arrays $eintrag. Wir nehmen hier $eintrag[1], damit der Betreff als Erstes angezeigt wird. ($eintrag[0] ist der Name). Analog werden dann die nächsten Zeilen aufgebaut. Beachten Sie, dass auch das schließende </table>-Tag an die Variable $ausgabe angehängt wird.

Danach ist das Ende des Ausdrucks der for-Schleife erreicht, denken Sie also an die schließende geschweifte Klammer.

Info

```
array explode(string trennzeichen, string daten, [,int limit])
```

Die Funktion zerlegt eine Zeichenkette (daten) durch ein vorher festgelegtes trennzeichen. Alle Elemente werden in einem Array zurückgegeben.

Zu guter Letzt müssen Sie noch für die Ausgabe der in der Variablen $ausgabe festgelegten Werte sorgen. Vor dem Schließen des <body>-Tags ergänzen Sie das Script deswegen um die Zeile

```
<?php echo $ausgabe; ?>
```

Im Code sehen Sie noch einmal den kompletten Teil, um den Sie das Script ergänzt haben:

```
$eintrag=file('gast.txt');
$ausgabe="<br>";
$temp=count($eintrag)-1;
for($i=$temp;$i>=0;$i--)
{
$element=explode('~',$eintrag[$i]);
$ausgabe.="<table>";
$ausgabe.="<tr><td><b>".$element[1]."</b></td></tr>";
$ausgabe.="<tr><td>".$element[0]."</td></tr>";
$ausgabe.="<tr><td>".$element[2]."</td></tr>";
$ausgabe.="</table>";
}
```

Speichern Sie das Script ab und testen Sie es. Füllen Sie alle Felder des Formulars aus, und schicken Sie es ab. Diese Eingaben müssten nun oben in der Liste zu sehen sein. Die älteren Eingaben vom ersten Test werden darunter aufgelistet (es sei denn, Sie haben sie in der Textdatei gelöscht). In dem Bild 7 sehen Sie, wie die Seite in etwa aussehen müsste.

Info

Es ist gut möglich, dass im Browser bestimmte Einträge zweimal oder mehrfach angezeigt werden, weil Sie die Daten beim Probieren auch mehrfach abgeschickt oder die Seite aktualisiert haben. Dieses Phänomen zu verhindern, ist nicht ganz einfach. Auch bei einigen professionellen Webseiten wird mit dem Hinweis auf der Seite: „Bitte das Formular nur einmal abschicken“ gegen dieses Problem angekämpft. Die Lösung werden wir nicht vorstellen können, aber alle benötigten Techniken werden in anderen Zusammenhängen

erklärt. Der Weg ist folgender:

Beim ersten Aufruf des Formulars generieren Sie eine eindeutige Nummer. Diese Nummer wird in ein verstecktes Feld in das Formular geschrieben. Die gleiche Nummer speichern Sie in einer Datenbanktabelle, in der alle Nummern der noch nicht empfangenen Formulare gespeichert werden.

Nachdem das Formular abgeschickt wurde, testen Sie zunächst alle Angaben des Formulars. Sind alle diese Tests bestanden, suchen Sie die Nummer des Formulars in der Tabelle. Entdecken Sie sie, können Sie die Daten speichern. Anschließend löschen Sie die Nummer des Formulars aus der Tabelle, da die Daten dieses Formulars bereits gespeichert wurden.

Wird das gleiche Formular nun ein zweites Mal abgeschickt, steht in dem versteckten Formularfeld immer noch die gleiche Nummer. Die Suche nach dieser Nummer in der Tabelle wird aber ergebnislos verlaufen, da sie nach dem ersten Abschicken gelöscht wurde. In diesem Fall können Sie eine Fehlermeldung ausgeben und das wiederholte Speichern der Daten unterbinden.

Interessante Webseite die kostenloses Gästebuch mit vielen Features anbietet, sowie ein Gästebuch-Script.

Eine besondere Stärke und ein typischer Einsatzzweck von **PHP** ist jedoch die Auswertung von Benutzereingaben aus Formularen. Erst durch eine solche Auswertung wird die dynamische Informationsübermittlung zwischen Benutzer und Webserver ermöglicht. Dem Betrachter wird zunächst ein Formular vorgelegt, in dem er eigene Einträge vornehmen beziehungsweise bei dem er aus bereits vorhandenen Einträgen auswählen kann. Er füllt das Formular aus, sendet es ab und erhält nach der Auswertung eine Antwort vom Webserver.

Eingabeformular

In diesem Abschnitt soll eine Informationsübermittlung mit Hilfe von einzeiligen Texteingabefeldern ermöglicht werden. **Formulare** können noch aus einer Reihe weiterer Elemente bestehen.

```
<html>
<body>
<p>Bitte tragen Sie Ihren Vornamen und Ihren Nachnamen ein.
<br>
Senden Sie anschließend das Formular ab.</p>
<form action = "eingabe.php" method = "post">
<p><input name = "vor" /> Vorname</p>
<p><input name = "nach" /> Nachname</p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Die Ausgabe des Formulars im Browser, mit eingegebenen Beispieldaten, sehen Sie in Abbildung 1.

Innerhalb des HTML-Dokuments befindet sich ein `form`-Container. Die Markierung `<form>` beinhaltet

- das Attribut `action`, das auf die Datei mit dem **PHP-Auswertungsprogramm** (hier: eingabe.php) verweist, und
- das Attribut `method`, das auf die Übermittlungsmethode zum Webserver (hier `post`) verweist.

Der `form`-Container beinhaltet die verschiedenen Formularelemente. Dabei handelt es sich um:

- zwei einzeilige Texteingabefelder mit den Namen vor beziehungsweise nach für die Eintragung des Vornamens beziehungsweise des Nachnamens
- eine Schaltfläche zum Absenden (engl. `submit`); beim Betätigen werden die eingetragenen Daten an den Server gesendet, und es wird das genannte PHP-Auswertungsprogramm angefordert
- eine Schaltfläche zum Zurücksetzen (engl. `reset`) des Formulars; beim Betätigen wird das Formular wieder in den Anfangszustand versetzt, wie es zum Beispiel bei einer Fehleingabe notwendig sein kann.

Die Auswertung der Eingabedaten stelle ich im folgenden Abschnitt vor.

Auswertung mit $_POST

Das antwortende **PHP-Programm** für das Formular in Datei eingabe.htm sieht wie folgt aus:

```
<html>
<body>
<?php
echo "Guten Tag, " . $_POST["vor"]
. " " . $_POST["nach"];
?>
</body>
</html>
```

Falls der Benutzer das oben angegebene Beispiel eingegeben hat, antwortet der Server, wie in Abbildung 2 dargestellt.

Es gibt in PHP einige vordefinierte **Variablen**, unter anderem das assoziative Feld `$_POST`. Aus den Namen der Eingabefelder werden automatisch Elemente dieses Feldes, falls die Übermittlungsmethode `post` verwendet wird.

Die Elemente können angesprochen werden, indem Sie ihren Namen in Anführungszeichen und eckigen Klammern hinter dem Namen des Feldes `$_POST` angeben. Die Eintragung im Texteingabefeld vor wird also zum Wert der Variablen `$_POST["vor"]` im Programm.

Feldelemente lassen sich allerdings nicht in einer Zeichenkette innerhalb von Hochkommata ausgeben, wie dies bei einzelnen Variablen der Fall ist. Daher ist die Ausgabezeile mit `echo` etwas umfangreicher.

Sie können ein Formular statt mit der Methode `post` auch mit der Methode `get` versenden. Sie müssen dabei darauf achten, dass Sie das Feld `$_GET` statt des Feldes `$_POST` verwenden. Die Methode `post` ist im Allgemeinen zu bevorzugen, da sie sicherer und universell ist.

Umwandlung von Zeichenketten in Zahlen

Ein Texteingabefeld eines Formulars nimmt eine Zeichenkette auf; es wird dabei eine **Zeichenkette** an das PHP-Programm übermittelt. Häufig sollen jedoch Zahlen, zum Beispiel zur Ausführung von Berechnungen, übermittelt werden. Dabei sind die folgenden Regeln zu beachten.

Bei der Umwandlung einer Zeichenkette (Konvertierung) ist der Beginn der Zeichenkette wichtig. Falls sie mit gültigen numerischen Zeichen beginnt, werden diese Zeichen genutzt. Andernfalls ergibt sich der Wert 0. Eine gültige Folge von numerischen Zeichen beinhaltet:

- ein Vorzeichen (optional)
- eine oder mehrere Ziffern
- einen Dezimalpunkt (optional)
- einen Exponenten (optional); der Exponent ist ein kleines e oder ein großes E, gefolgt von einer oder mehreren Ziffern

Die Zeichenkette wird interpretiert

- als ganze Zahl, falls sie nur Ziffern beinhaltet, oder
- als Zahl mit Nachkommastellen, falls sie neben den Ziffern die Zeichen `.` (Punkt), e oder E beinhaltet.

Einige Beispiele sehen Sie in Tabelle.

Zeichenkette	Wert	Datentyp
`"352"`	352	ganze Zahl
`"352xz"`	352	ganze Zahl
`"xz352"`	0	Zeichenkette
`"35.2"`	35.2	Zahl mit Nachkommastellen

Zeichenkette	Wert	Datentyp
"35.2xz"	35.2	Zahl mit Nachkommastellen
"xz35.2"	0	Zeichenkette
"-352"	-352	ganze Zahl
"35e2"	3500	Zahl mit (möglichen) Nachkommastellen
"35e-2"	0.35	Zahl mit Nachkommastellen

Falls Sie **Zeichenkettenvariablen** der Sicherheit halber explizit (also vom Programmentwickler gesteuert) in Zahlen umwandeln möchten, können Sie die beiden Funktionen `doubleval()` beziehungsweise `intval()` anwenden. Ein kleines Beispiel für zwei Umwandlungen:

```
$a = "435";
$a = intval($a);
$b = "22.6";
$b = doubleval($b);
```

Nach der Bearbeitung dieses Programmteils stehen die Variablen $a und $b als Zahlenvariablen mit dem ganzzahligen Wert 435 beziehungsweise dem Wert 22.6 für weitere Berechnungen zur Verfügung.

Im nachfolgenden Beispiel wird der Benutzer aufgefordert, in einem Formular zwei Zahlen einzugeben und das Formular abzusenden. Ein PHP-Programm berechnet die Summe der beiden Zahlen und gibt das Ergebnis aus. Der **HTML-Code** des Formulars:

```
<html>
<body>
<p>Bitte tragen Sie zwei Zahlen ein und senden Sie das Formular ab.</p>
<form action = "eingabe_zahl.php" method = "post">
<p>Wert 1: <input name = "w1" /></p>
<p>Wert 2: <input name = "w2" /></p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Das PHP-Programm:

```
<html>
<body>
<?php
$erg = $_POST["w1"] + $_POST["w2"];
echo "Die Summe von " . $_POST["w1"]
. " und " . $_POST["w2"] . " ist $erg";
?>
</body>
</html>
```

Ein Aufruf mit den in Abbildung 3 dargestellten Eingabewerten ergibt die in Abbildung 4 dargestellte Antwort.

Im Antwortprogramm werden die eingegebenen Zeichenketten nach den oben angegebenen Regeln automatisch in Zahlen umgewandelt.

Es gibt Form-Mailer die in **Perl** geschrieben sind und auf dem Webserver als CGI-Script laufen. Das gleiche Script wollen wir nun in PHP realisieren. Dabei gehen wir wie folgt vor:

- Der HTML-Code des Feedback-Formulars, das als Beispiel dient, wird in einer Datei namens feedback.htm abgelegt.
- Der HTML-Code der Danke-Seite, die angezeigt wird, wenn die Formulardaten erfolgreich ausgewertet und versendet wurden, wird in einer Datei namens danke.htm abgelegt.
- Ein PHP-Script verwaltet das ganze Feedback-Handling, d.h., es zeigt das Formular an, prüft Eingaben auf fehlende Daten und versendet, wenn alles in Ordnung ist, dem Seitenanbieter eine Mail mit den Formulardaten.

Die HTML-Dateien: nicht für den direkten Aufruf bestimmt

Die Dateien feedback.htm und danke.htm sollen nicht vom Browser direkt aufgerufen werden. In unserem Beispiel werden wir sie zwar der Einfachheit halber im gleichen Verzeichnis wie das PHP-Script ablegen, doch in der Praxis ist es besser, solche Dateien gleich außerhalb der Document Root abzulegen, damit sie für Webclients gar nicht erst erreichbar sind. Die beiden HTML-Dateien enthalten nichts Ungewöhnliches bis auf zwei Platzhalter der Marke Eigenbau, wie wir sie im vorangegangenen Abschnitt bereits verwendet haben:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

```
<title>Feedback</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<h1>Feedback</h1>
<form action="[%self%]" method="post">
<input type="hidden" name="return" value="/danke.htm">
<input type="hidden" name="subject" value="Feedback-Formular">
<fieldset><legend>[%legend%]:</legend>
<div class="explain">Felder mit Stern * müssen ausgefüllt werden.</div>
</fieldset>
</td>
</tr>
<tr>
<td><fieldset><legend>Daten:</legend>
<div>Vorname:<br>
<input type="text" name="Vorname" class="text" style="width:400px">
</div>
<div>Zuname:<br>
<input type="text" name="Zuname" class="text" style="width:400px">
</div>
<div>E-Mail: <b>*</b><br>
<input type="text" name="Mail" class="text" style="width:400px">
</div>
<div>Feedback: <b>*</b><br>
<textarea name="Text" class="text" style="width:400px; height:300px"></textarea>
</div>
</fieldset>
<fieldset><legend>Daten:</legend>
<p>
<input type="submit" name="feedback" class="button" value="Absenden">
<input type="reset" class="button" value="Löschen">
</p>
</fieldset>
</form>
</body>
</html>
```

HTML-Datei danke.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
```

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Danke für Ihr Feedback</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<h1>Danke für Ihr Feedback</h1>
<p>Ihre eingegebenen Daten wurden uns zugesendet. Wir werden uns mit Ihnen in
Verbindung setzen.</p>
</body>
</html>
```

Info
In der Datei feedback.htm sind zwei Platzhalter notiert: `[%self%]` steht bei der Wertzuweisung an das `action`-Attribut im einleitenden `<form>`-Tag. Hier wird das **PHP-Script** seine eigene Adresse einsetzen, damit das Dateiensemble ohne Änderungen von einer Umgebung zu einer anderen portiert werden kann. Der zweite Platzhalter `[%legend%]`, Inhalt der Feldgruppenüberschrift, wird vom Script unterschiedlich versorgt, je nachdem, ob das Formular erstmalig angezeigt wird oder ob es wiederholt angezeigt wird, weil der Anwender zuvor erforderliche Daten nicht eingegeben hat.

Das PHP-Script: Verzweigungen, Funktionen und Selbstaufruf

Der Quelltext des PHP-Scripts sieht wie folgt aus:

```
<?php
#-----------------------------------------
# Globale Daten:
$mailto = "jemand@irgend.wo";
$form_file = "feedback.htm";
$thanks_file = "danke.htm";
$standard_legend = "Hinweis";
$error_legend = "Fehler";
#-----------------------------------------
# Entscheiden was zu tun ist:
if(isset($_POST['feedback']))
check_form_data();
else
show_form_data(false);
#-----------------------------------------
# Funktion show_form_data():
# Formular anzeigen
function show_form_data($with_error) {
global $form_file, $standard_legend, $error_legend;
$page = file_get_contents($form_file);
$page = preg_replace("/\[\%self\%\]/",
```

```
$_SERVER['SCRIPT_NAME'], $page);
if($with_error == true)
$page = preg_replace("/\[\%legend\%\]/",
$error_legend, $page);
else
$page = preg_replace("/\[\%legend\%\]/",
$standard_legend, $page);
echo $page;
exit(); }
#----------------------------------------
# Funktion check_form_data():
# Formulareingaben prüfen
function check_form_data() {
//echo "<pre>", var_dump($_POST), "</pre>";
if(empty($_POST['Mail']) or empty($_POST['Text']))
show_form_data(true);
else
mail_and_thanks();
}
#----------------------------------------
# Funktion mail_and_thanks():
# Formulareingaben mailen und Dankeseite ausgeben
function mail_and_thanks() {
global $mailto, $thanks_file;
$headers = "From: {$_POST['Vorname']} {$_POST['Zuname']} ";
$headers .= "<{$_POST['Mail']}>\r\n";
$headers .= "Content-Type: text/plain; charset=ISO-8859-1\r\n";
$headers .= "Content-Transfer-Encoding: 8bit";
mail($mailto, $_POST['subject'], $_POST['Text'], $headers);
$page = file_get_contents($thanks_file);
echo $page;
exit();
} ?>
```

Info
In diesem Script führen wir eine Technik ein, mit der wir in den vorangegangenen Beispielen noch nicht gearbeitet haben: eigene Funktionen. Insgesamt sind im Script drei Funktionen notiert:

- **show_form_data()** zeigt das Formular an. Dazu liest die Funktion die Datei feedback.htm ein, ersetzt darin die beiden Platzhalter und gibt den HTML-Code am Ende aus. Die Funktion erwartet einen Parameter namens `$with_error`, der den Boolean-Wert `false` oder `true` haben sollte. Abhängig von diesem übergebenen Wert ersetzt die Funktion den Platzhalter im Legendentext der Feldgruppenüberschrift entweder mit „Hinweis“ oder mit „Fehler“.
- **check_form_data()** überprüft eingegebene Formulardaten daraufhin, ob die beiden Formular-Pflichtfelder Mail und Text einen Inhalt haben. Wenn ja, wird die Funktion `mail_and_thanks()`

aufgerufen. Sind nicht beide Felder gefüllt, dann wird die Funktion `show_form_data()` aufgerufen, und zwar so, dass sie „Fehler“ mit ausgibt.
- **mail_and_thanks()** erzeugt aus den übergebenen Formulardaten eine E-Mail, sendet sie an den Seitenbetreiber und gibt abschließend die Danke-Seite aus, indem sie die danke.htm einliest und ihren Inhalt ausgibt.

Die gesamte Logik des Scripts erschließt sich allerdings nur beim Betrachten des Abschnitts „Entscheiden, was zu tun ist“ im oberen Teil.

Dort wird abgefragt, ob eine Variable namens

`$_POST['feedback']` existiert. Da im **HTML-Formular** beim `<form>`-Tag `method="post"` angegeben ist, wird das PHP-Script auf dem Webserver mit der HTTP-Request-Methode POST aufgerufen. Der Webserver stellt dem Script die Formulardaten in Form einer entsprechenden Umgebungsvariablen zur Verfügung. In PHP können Sie über die superglobale Variable `$_POST` darauf zugreifen. Die Parallelvariable `$_GET` haben wir ja bereits kennen gelernt. Bei `$_POST` handelt es sich wie bei `$_GET` um einen assoziativen Array. Feldbezeichner erscheinen darin als Schlüssel und Feldwerte als den Schlüsseln zugewiesene Werte.

Eine Variable `$_POST['feedback']` gibt es dann, wenn unser Formular ausgefüllt und versendet wurde, und zwar deshalb, weil es innerhalb des Formulars ein Feld mit dem Namen feedback gibt. Wir haben den `Submit`-Button so benannt:

```
<input type="submit" name="feedback" class="button" value="Absenden">
```

Info
Beim Erstaufruf des Scripts liegen natürlich noch keine Formulardaten vor. In diesem Fall gelangt das Script in den `else`-Zweig der `if`-Abfrage nach `$_POST['feedback']`. Dort wird die Funktion `show_form_data()` aufgerufen. Als Parameter wird ihr der Boolean-Wert `false` übergeben. Die Funktion verrichtet ihren Dienst, d.h., sie liest die Datei feedback.htm ein, ersetzt darin den Platzhalter `[%self%]` durch den Inhalt der superglobalen Variablen `$_SERVER['SCRIPT_NAME']` (absoluter Webpfadname des Scripts auf dem Server), den Platzhalter `[%legend%]` wegen des übergebenen Parameters `false` auf „Hinweis“, gibt die Seite mit `echo` aus und beendet dann mit `exit()` das Script.

Die Seite mit dem Formular wird nun beim Anwender angezeigt und er hat Zeit,

das Formular auszufüllen. Im einleitenden `<form>`-Tag steht nun wegen der Platzhalterersetzung: `action="/buch/feedback.php"` (vorausgesetzt, das PHP-Script wurde so benannt). Das bedeutet, nach Absenden des Formulars wird das Script erneut aufgerufen.
Bei diesem erneuten Aufruf liegen die eingegebenen Formulardaten im superglobalen Array `$_POST` vor.

Das Script gelangt nun in den Ausführungszweig unterhalb der `if`-Abfrage `if(isset($_POST['feedback']))`. Dort wird `check_form_data()` aufgerufen. In dieser Funktion wird lediglich geprüft, ob die beiden Pflichtfelder mit den Namen Mail und Text leer (empty) sind:

```
if(empty($_POST['Mail']) or empty($_POST['Text']))
```

Info
Ist das der Fall, wurde das Formular nicht vollständig ausgefüllt und es soll noch mal angezeigt werden. Deshalb wird in diesem Fall `show_form_data()` aufgerufen – diesmal jedoch mit dem Parameter `true`, was `show_form_data()` veranlasst, das Formular mit der Legendenüberschrift „Fehler“ anzuzeigen. Der Anwender kann das Formular dann erneut ausfüllen.

Wurden dagegen beide Pflichtfelder ausgefüllt, gelangt das Script in den `else`-Zweig. Dort steht ein Aufruf der Funktion `mail_and_thanks()`.

Die Funktion `mail_and_thanks()` stellt zunächst in der Variablen `$header` ein paar wichtige Zeilen zusammen, die in den Kopfdaten der zu versendenden Mail stehen sollen – unter anderem das wichtige Feld `From:`, das den Absender der Mail enthält. Hier trägt die Funktion eine Zeichenkette ein, die sich aus Werten des Formulars ergibt: `{$_POST['Vorname']} {$_POST['Zuname']} <{$_POST['Mail']}>`. Das erzeugt eine Angabe wie etwa Rainer Zufall <merk@wuerd.ig>. Der Seitenanbieter erhält dann also die Mail gleich so, als wäre sie vom E-Mail-Programm des Anwenders abgesendet worden. Er kann selber ganz einfach auf die Mail antworten, indem er in seinem Mailprogramm die Funktion „Antworten“ wählt.

> Mit der PHP-Funktion `mail()` wird die E-Mail versendet.

Die Funktion kann allerdings nur dann Mails versenden, wenn die php.ini entsprechend konfiguriert wurde. Dort finden Sie – per Voreinstellung auskommentiert – die Einträge SMTP und sendmail_from für PHP unter MS Windows und den Eintrag sendmail_path für Linux/Unix-Systeme. Auf Shell-Ebene eines Linux-Rechners können Sie den Pfad mit dem Kommando `which sendmail` ermitteln.

An `mail()` müssen mindestens drei Parameter übergeben werden: 1. die Mailadresse des Empfängers, 2. das Mail-Subject und 3. der Inhalt der Mail. Als vierten Parameter können zusätzliche Kopfdaten für die Mail übergeben werden und als fünften Parameter Aufrufparameter für das `sendmail`-Programm. In unserem Beispiel werden vier Parameter übergeben. Die Mailadresse des Seitenanbieters ist in der Variablen `$mailto` gespeichert, die zu Beginn des Scripts definiert wird. Als Mail-Subject wird `$_POST['subject']` übergeben. Das Feld mit dem Namen `subject` ist im HTML-Formular ein `hidden`-Feld. Als Inhalt der Mail wird `$_POST['Text']` übergeben, also der Inhalt aus dem `textarea`-Feld in HTML.

Abschließend liest die Funktion `mail_and_thanks()` den Inhalt der Danke-Seite mit `file_get_contents()` ein und gibt ihn mit `echo` aus.

Fazit: PHP vereinfacht die Formularauswertung

Wer schon einmal versucht hat, mit POST übergebene Formulardaten in einem Script direkt von der Standardeingabe zu lesen, wird es zu schätzen wissen, dass es in PHP genügt, die gewünschten Felder einfach aus dem superglobalen Array `$_POST` zu holen. Das Dekodieren der Daten, welche bei der Übertragung nach dem Schema `www-form-url-encoded` kodiert werden, das Auseinanderdividieren von Feldnamen und Werten – all das hat PHP schon erledigt, bevor überhaupt ein Script gestartet ist, das Formulardaten auslesen möchte.

Auch das Versenden von E-Mails ist dank der praktischen Funktion `mail()` sehr bequem. Allerdings setzt die Funktion voraus, dass ein `sendmail`-Programm konfiguriert ist, das in der Lage ist, ausgehende Mails zu verarbeiten und via SMTP-Protokoll zu versenden. Auf öffentlichen Server-Rechnern mit Linux-System ist das kein großes Problem. Dort ist eigentlich immer eines der Programme sendmail oder qmail installiert und aktiv. Bei anderen Programmen als sendmail muss allerdings ein `sendmail`-Wrapper installiert werden, der PHP vortäuscht, es habe es mit sendmail selber zu tun.

Solange nur „PHP in HTML“ angewendet wird, ist kein strukturelles Umdenken nötig. Das ändert sich jedoch, wenn man PHP stattdessen in der Form „HTML in PHP“ einsetzt. Während Einsteiger eher die Variante „PHP in HTML“ bevorzugen, kommt bei größeren und professionellen Sites eher die Variante „HTML in PHP“ zum Einsatz. Das strukturelle Umdenken besteht darin, dass ein PHP-Script bei der Variante „HTML in PHP“ in der Regel ein Behälter für mehrere HTML-Dokumente ist, die auch unterschiedliche URIs haben.
Ein Beispiel soll dies demonstrieren. Dabei setzen wir unser Beispiel mit den Vorschaubildern fort. In den Links aller drei anklickbaren Vorschaugrafiken wurde dort die gleiche PHP-Datei aufgerufen, nur mit unterschiedlichem GET-String:

```
<a href="bild.php?bild=01">
<img src="bild01thumb.jpg" class="thumb_h" alt="Bild 1 Vorschau"></a>
<a href="bild.php?bild=02">
<img src="bild02thumb.jpg" class="thumb_v" alt="Bild 2 Vorschau"></a>
<a href="bild.php?bild=01">
<img src="bild03thumb.jpg" class="thumb_h" alt="Bild 1 Vorschau"></a>
```

Info
Die verlinkte Datei **bild.php** fungiert also offensichtlich als eine Art „Allgemeindokument“, das in der Lage ist, verschiedene Bilder auszugeben. Nachfolgendes Listing enthält den vollständigen Quelltext von **bild.php**:

```
<?php
$pic_number = (int) $_GET['bild'];
$pic_file = "bild".$_GET['bild'].".jpg";
$pixel = $pixel = GetImageSize($pic_file);
$css_width = $pixel[0]."px";
$css_height = $pixel[1]."px";
```

```
$csv = fopen("bilder.csv", "r");
while(!feof($csv))
$csv_lines[] = fgets($csv);
fclose ($csv);
for($i = 0; $i < count($csv_lines); $i++) {
$fields[$i] = explode(";", $csv_lines[$i]);
}
echo <<<END_1
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Bild $pic_number</title>
<style type="text/css">
td.left {
vertical-align:top;
padding-right:15px;
}
td.right {
vertical-align:top;
}
h1 {
border-bottom:1px solid black;
}
h3 {
margin-top:0;
}
</style>
</head>
<body>
<h1>Bild $pic_number</h1>
<table>
<tr>
<td class="left">
<a href="bilder.php"><img src="$pic_file" style="width:$css_width;
height:$css_height; border:solid 1px black" alt="Bild $pic_number"></a></p>
<p><b>zurück:</b> <a href="bilder.php">Vorschaubilder</a></p>
</td>
<td class="right">
<h3>Daten zum Bild:</h3>
<p>
<b>Aufnahmedatum:</b> {$fields[$pic_number][1]}
<br>
<b>Aufnahmeort:</b> {$fields[$pic_number][2]}
<br>
```

```
<b>Bildautor:</b> {$fields[$pic_number][3]}
<br>
<b>Beschreibung:</b> {$fields[$pic_number][4]}
<br>
</p>
</td>
</tr>
</table>
</body>
</html>
END_1;
?>
```

Info
Im Beispielscript ist der gesamte Inhalt in einen großen PHP-Bereich eingeschlossen. Es gibt also gar keinen „HTML-Modus“ mehr in dieser Datei, sondern nur noch Script-Anweisungen.

Für größere HTML-Passagen: here-Dokumente

Ein großer Teil des Beispielcodes besteht jedoch aus durchgängig notiertem HTML. Das liegt daran, dass mithilfe der bereits bekannten Anweisung `echo` diesmal ein so genanntes **here-Dokument** ausgegeben wird. Die Syntax dazu lautet:

```
echo <<<MARKE
<!-- viel HTML -->
MARKE;
```

Info
Dabei ist MARKE ein frei wählbarer Name. Wichtig sind die drei öffnenden spitzen Klammern vor dem Namen. Alles, was dann folgt, ist normaler HTML-Code, bis in einer Zeile alleinstehend der Name der Marke steht, abgeschlossen von einem Semikolon. Der HTML-Code eines `here`-Dokuments kann auch PHP-Variablen enthalten – der PHP-Interpreter erkennt diese und gibt an der entsprechenden Stelle den Wert der Variablen aus. Das `here`-Dokument in unserem Beispiel enthält an einigen Stellen Variablen.

`echo` ist übrigens keine PHP-Funktion. Deshalb wird der übergebene Inhalt auch nicht wie bei Funktionen sonst üblich in runden Klammern notiert. Der Inhalt kann entweder eine Zeichenkette sein, z.B.:

```
echo "<h1>Willkommen</h1>";
```

Die Zeichenkette darf auch Variablen enthalten:

```
echo "<h1>Willkommen, $name</h1>";
```

Bei Array-Variablen oder anderen komplexen Variablen müssen diese allerdings in geschweifte Klammern eingeschlossen werden:

```
echo "<h1>Willkommen, {$name[23]}</h1>";
```

Die andere Möglichkeit ist eine kommagetrennte Liste aus Zeichenketten und Variablen:

```
echo "<h1>Willkommen, ", $name[23], "</h1>";
```

Info
Oder eben ein here-Dokument: Innerhalb von here-Dokumenten gelten die gleichen Regeln wie bei der Notation in einer Zeichenkette, d.h., einfache Variablen wie $name können normal notiert werden. Komplexe Variablen wie $name[23] müssen mit geschweiften Klammern notiert werden, also in der Form {$name[23]}.

Erst Datenverarbeitung, dann Datenausgabe

Unser Beispielscript hat einen typischen Aufbau. Im oberen Teil werden Daten ermittelt und für die Ausgabe vorbereitet, im unteren Teil erfolgt die Ausgabe.

Das Script bekommt beim Aufruf Daten mittels GET-String übergeben. Im HTML-Code der Vorschaubilder sah das so aus:

```
<a href="bild.php?bild=02">
<img src="bild02thumb.jpg" class="thumb_v" alt="Bild 2 Vorschau"></a>
```

Info
Über die Superglobalvariable $_GET kann ein Script auf Daten zugreifen, die auf diesem Weg übergeben wurden. $_GET ist ein assoziativer Array. Deshalb kann das Beispielscript mit $_GET['bild'] direkt auf ein Name-Wert-Paar wie bild=02 zugreifen. In $_GET['bild'] ist dann etwa der Wert „02“ gespeichert.

Die ersten fünf Anweisungen in **bild.php** definieren verschiedene Variablen und weisen ihnen einen ermittelten Inhalt zu. Vieles davon sollte Ihnen bereits aus der **bilder.php** des vorhergehenden Abschnitts bekannt sein. Unser Augenmerk soll dem Teil gelten, der daran anschließend folgt. Die Textdaten, die im Browser neben dem Bild angezeigt werden, kommen nämlich aus einer externen Datei. Diese hat folgenden Inhalt:

```
picture_file;record_date;record_place;record_author;record_text
```

```
bild01.jpg;01.08.1999;Lettland, bei Sigulda;Darko Pipic;
Die Aufnahme zeigt die typische Feld- und Wiesenlandschaft der Umgebung von
Sigulda.
bild02.jpg;02.08.1999;Lettland, Schloss Sigulda am Gauja-Fluss;Darko Pipic;
Die Aufnahme zeigt die Seitenansicht des Schlosses.
bild03.jpg;09.08.1999;Estland, Tartu;Darko Pipic;
Die Aufnahme zeigt die zentrale Einkaufsstraße der Stadt.
```

Info
Es handelt sich also um eine typische kommagetrennte Datendatei, wie sie beispielsweise in Programmen wie Excel leicht erzeugbar ist. Jede Zeile (hier leider aus satztechnischen Gründen umgebrochen) stellt einen „Datensatz“ dar. Trennzeichen für Felder der Datensätze ist das Semikolon. Die erste Zeile in der Datei weist die Felder aus. Diese Datei haben wir unter dem Namen **bilder.csv** abgespeichert. In PHP wird die Datei eingelesen. Hier noch mal der dazu relevante Code-Ausschnitt:

```
$csv = fopen("bilder.csv", "r");
while(!feof($csv))
$csv_lines[] = fgets($csv);
fclose ($csv);
```

Info
Auf die Funktionen zum Lesen aus und Schreiben in Dateien gehen wir später noch näher ein. Hier soll es genügen, zu verstehen, dass die Datei mit `fopen()` geöffnet wird. Dann wird sie so lange (`while`) zeilenweise mit der Funktion `fgets()` eingelesen, bis das Dateiende `feof()` erreicht ist, um schließlich mit `fclose()` geschlossen zu werden. Am Ende gibt es einen Array namens `$csv_lines`, bei dem in jedem Element eine Zeile der eingelesenen Datei gespeichert ist.

Damit haben wir zwar die Datensätze schon sauber getrennt, doch noch nicht die einzelnen Datenfelder. Das geschieht in der anschließenden `for`-Schleife:

```
for($i = 0; $i < count($csv_lines); $i++) {
$fields[$i] = explode(";", $csv_lines[$i]);
}
```

Info
Für jedes Element von `$csv_lines` wird dessen Inhalt mithilfe der PHP-Funktion `explode()` auseinander genommen, und zwar an Hand des vereinbarten Trennzeichens (`;`). Am Ende der `for`-Schleife gibt es einen verschachtelten Array namens `$fields`. Jedes Element dieses Arrays stellt zwar wie in `$csv_lines` einen Datensatz dar, doch ist jedes Element diesmal selbst ein untergeordneter Array, bestehend aus den jeweiligen Datenfeldern der Zeile.

Im Ausgabeteil des `here`-Dokuments ist zu sehen, wie die Daten durch Zugriff auf diesen Array

ausgegeben werden. Der relevante Ausschnitt:

```
<b>Aufnahmedatum:</b> {$fields[$pic_number][1]}
<br>
<b>Aufnahmeort:</b> {$fields[$pic_number][2]}
<br>
<b>Bildautor:</b> {$fields[$pic_number][3]}
<br>
<b>Beschreibung:</b> {$fields[$pic_number][4]}
<br>
```

Info
Die doppelten eckigen Klammern hinter `$fields` signalisieren den „doppelten Array“. In der Variablen `$pic_number` ist die Nummer des anzuzeigenden Bildes gespeichert. Die CSV-Datei ist so sortiert, dass dadurch gleich der Zugriff auf den richtigen Datensatz möglich ist. Angenommen, in `$pic_number` ist 1 gespeichert. Dann ist `$fields[$pic_number][1]` das Gleiche wie `$fields[1][1]` und greift auf das Datum 01.08.1999 aus den CSV-Daten zu. Grund ist die 0-basierte Zählweise von Arrays in PHP, die in dieser Hinsicht denen von JavaScript gleichen.

Fazit: Umdenken lohnt sich

Die Abkehr von dem Denken „eine PHP-Datei pro Webseite“, welche typisch ist für die einfachere Programmierweise „PHP in HTML“, hat unbestreitbare Vorteile: Der PHP-Code wird zentralisiert und so organisiert, dass er für eine beliebige Menge von Webseiten wie ein Container ist. Dadurch wird der Code leichter wartbar und muss nicht mehr in vielen Dateien wiederholt werden.

Nachdem Sie nun die Grundlagen von **PHP** kennen gelernt haben, sind Sie in der Lage, PHP für eine erste Aufgabe einzusetzen. Sie alle kennen Webseiten, auf denen Sie aufgefordert werden, in bestimmten Feldern etwas einzugeben, beispielsweise Namen, Benutzernamen, E-Mail-Adresse etc. Ähnlich häufig bieten Webseiten Auswahlfelder, in denen Sie sich für eine der Optionen entscheiden, oder Kontrollkästchen, mit denen Sie durch Aktivierung eines der Kästchen eine Wahl treffen. Solche Seiten können mit einem mehr oder minder aufwändigen HTML-Formular realisiert werden. PHP übernimmt die Aufgabe, die Formulardaten zu verarbeiten und auszuwerten. Ein **HTML-Formular** in Kombination mit PHP ist der praktikabelste Weg, aus Ihrer Webseite interaktive Webseiten zu erzeugen.

Dieser Artikel wird zeigen, wie Sie ein Formular erzeugen und wie die Daten vom Browser zum Server und damit zum PHP-Script transferiert werden. Das Formular wird drei Eingabefelder enthalten sowie ein Mehrfachauswahlfeld, sodass – da ja unterschiedliche Optionen ausgewählt werden können – im PHP-Script auch eine `if`-Anweisung zum Einsatz kommt. Natürlich müssen in diesem Zusammenhang auch die beiden unterschiedlichen Methoden diskutiert werden, mit denen die Übertragung der Daten zum Server erfolgen kann: Get oder Post.

Vorüberlegungen

Im ersten Schritt erzeugen Sie das Formular mithilfe der entsprechenden **HTML-Tags** zur Darstellung von Eingabefeldern. Da Sie die Tags und die verschiedenen Elemente/Parameter für ein Formular vielleicht nicht alle parat haben, stellen wir auch die HTML-Lösung kurz vor. Das Entscheidende ist folgender Zusammenhang:

Im einleitenden `<Form>`-Tag geben Sie mit dem Attribut `action` eine Datei an, die aufgerufen wird, sobald Sie das Formular abschicken. Wird mit dem Formular auf diesem Weg eine PHP-Datei aufgerufen, stehen die Werte, die in das Formular eingegeben oder ausgewählt wurden, in der PHP-Datei als Variablen zur Verfügung. Die Namen der Variablen sind identisch mit denen der Formularelemente im Formular, also mit den Namen, die Sie mit dem Attribut `name` der einzelnen Formularelemente festlegen. Haben Sie in dem Formular ein Element mit dem Namen `nachname (name='nachname')`, wird der Wert bzw. der in das Feld eingegebene Text in dem aufgerufenen PHP-Script in der Variablen $nachname zur Verfügung stehen.

Das PHP-Script muss/sollte so geschrieben werden, dass mögliche Fehler bei der Eingabe

berücksichtigt werden, der Webseiten-Besucher also beispielsweise aufgefordert wird, die Felder auszufüllen, sofern er es nicht getan hat. Dann stellt sich die Frage: Was soll mit den Formulardaten geschehen? Denkbar ist als Verarbeitung, dass der Webseiten-Besucher nach dem Abschicken der Daten eine Seite erhält, die seine Eingaben nochmals bestätigt, also eine Art Feedback-Seite. Dies ist die einfache Variante und relativ schnell gemacht.

Der zentrale Punkt ist bei einem Kontaktformular jedoch das dauerhafte Sichern der Formulardaten. Dies geschieht in der Regel mithilfe einer separaten Textdatei oder einer Datenbank. Da Sie in diesem Kapitel jedoch die ersten konkreten Gehversuche in PHP-Programmierung machen, möchten wir hier noch nicht mit Dateien oder Datenbanken hantieren. Darüber dürfen Sie sich den Kopf zerbrechen, wenn Sie bereits ein paar Erfahrungen gesammelt haben. In diesem Beispiel sollen die Formulardaten lediglich an eine E-Mail-Adresse geschickt werden. Diese Aufgabe führt der **PHP-Befehl** `mail()` aus.

Für den Einstieg wird zunächst die eben erwähnte einfache Variante durchgespielt, in der die Eingaben nur zurückgegeben werden; anschließend wird die Kontrolle der Eingaben implementiert und danach ein neues Script erstellt, das keine Feedback-Seite erzeugt, sondern die Daten wie eben angekündigt an eine E-Mail-Adresse sendet.

Das Kontaktformular erstellen

Beginnen Sie nun damit, das HTML-Formular zu erstellen. Öffnen Sie dazu im Editor eine neue Datei und geben Sie die für ein HTML-Dokument üblichen Tags für das Grundgerüst ein:

```
<html>
<head>
```

```
<title>Ein Kontaktformular</title>
</head>
<body>
</body>
</html>
```

Im <body>-Container definieren Sie, beginnend mit dem <form>-Tag, die Formularfelder. Das Formular soll Eingabefelder für den Vornamen, den Nachnamen und die E-Mail- Adresse enthalten und außerdem ein Auswahlfeld, mit dem der Besucher die Möglichkeit hat, eine der in der Liste angebotenen Optionen auszuwählen.

Setzen Sie den Cursor unterhalb des <body>-Tags und geben Sie den HTML-Text so ein wie unten zu sehen. Speichern Sie das Dokument dann als eine HTML-Datei, also beispielsweise als Kontakt.html.

```
<html>
<head>
<title>Kontakt-Formular</title>
</head>
<body>
<p>
<h3>Geben Sie Ihre Daten ein!</h3>
<form action="antwort.php" method=post>
Vorname
<br>
<input type=text name="vorname" size=20>
<br>
Nachname
<br>
<input type=text name="nachname" size=20><p>
E-Mail
<br>
<input type=text name="email" size=30><p>
<h4>Wie hat Ihnen das Spiel gefallen?</h4><p>
<select size=1 name="rank">
<Option value="keine Angabe">keine Angabe</option>
<Option value="sehr gut">sehr gut</option>
<Option value="gut"> gut</option>
<Option value="nicht so gut">nicht so gut</option>
</select>
<p>
<input type=submit name="submit" value="abschicken">
</form>
</body>
</html>
```

Erläuterung des Scripts

Mit dem `<form>`-Tag beginnen Sie das Formular. Der Wert des dann folgenden Arguments `action=` gibt dem Server an, welches **PHP-Script** die Daten aus dem Formular erhalten wird. Die Eingabe `method=post` bestimmt die Methode, mit der die Daten weitergegeben werden. Die alternative Methode wäre `get`. Mit dem Tag `<input type=text>` wird ein Textfeld erzeugt, das Attribut `size` legt die sichtbare Größe des Textfeldes fest. Durch `<br>` wird einen Zeilenumbruch erzeugt, sodass das Formular insgesamt übersichtlicher aussehen wird.

Legen Sie Ihr Augenmerk besonders auf das Attribut `name`, beispielsweise `name="vorname"`.

> Der hier festgelegt Wert („vorname“) wird als Variable an das PHP-Script, das die Daten verarbeitet,

weitergegeben. PHP identifiziert damit den Wert des Formularelements. Es ist also wichtig, diese Namen zu erinnern und konsistent zu benutzen.

Vorsicht: PHP ist bei der Groß- und Kleinschreibung so sensibel wie ein deutscher Gymnasiallehrer – nicht hinsichtlich korrekter Schreibweise, aber hinsichtlich der gleichbleibenden Verwendung von großen oder kleinen Buchstaben! Es unterscheidet also zwischen Vorname und vorname.

Das Drop-Down-Menü

Neben den drei Textfeldern wird mit dem Element `<select>` eine Auswahlbox in Form eines Drop-Down-Menüs definiert, damit der geneigte Besucher eine Meinung abgeben kann. Die Werte der Auswahlbox werden einzeln mit

```
<option value="übermittelter Wert">Beschriftung</option>
```

eingegeben. Wird für eine Option kein `value` festgelegt, wird die Beschriftung übermittelt. Das schließende `</option>`-Tag ist laut HTML-Spezifikation nicht zwingend erforderlich, aber der Vollständigkeit halber verwenden wir es hier dennoch. Auch dieses Element braucht einen Namen, damit PHP darauf zugreifen kann. Im Beispiel wurde `rank` verwendet. Da `<select>` als Container dient, muss es durch `</select>` geschlossen werden.

Beachten Sie hier die erste Option keine Angabe. Sofern durch das Attribut `selected` keine Vorauswahl einer Option getroffen wurde, wird die erste Option automatisch im Feld angezeigt und wird – sofern der User keine andere Wahl trifft – beim Abschicken des Formulars übermittelt. Aus diesem Grund sind Sie gut beraten, wenn Sie eine Option standardmäßig aktivieren. So ist es möglich zu erkennen, ob der User aktiv eine Auswahl getroffen hat oder nicht, sodass spätere Auswertungen nicht verfälscht werden.

Die Sendeschaltfläche

Last but not least enthält das Formular eine Sendeschaltfläche. Diese Schaltfläche verwendet normalerweise den in `value` festgelegten Wert als Bezeichner. Wird als `value` nichts eingegeben, erscheint der Standardtext Submit oder Abfrage absenden – je nach Browser und Ländereinstellung. Vergessen Sie nicht, das Formular mit einem schließenden `</form>`-Tag zu beenden.

Das von uns entworfene Formular ist, wie Sie sehen werden oder bereits am HTML-Text erkennen, relativ schlicht und enthält wenig Layout-Elemente. Sie können selbstverständlich an dem Aussehen des Formulars feilen, beispielsweise durch die Verwendung einer Tabelle, der Formatierung des Textes etc. Wir wollen uns jedoch auf die Vermittlung von PHP konzentrieren und verzichten von daher in den meisten Fällen bzw. Beispielen auf eine ausgeklügelte Layout-Technik.

Überprüfen des Formulars

Sobald Sie das HTML-Formular erstellt und gespeichert haben, können Sie im Browser einen Blick darauf werfen. Rufen Sie einfach Ihren Standard-Browser und die entsprechende Datei auf.

Die Übertragungsmethoden Post und Get

Die Daten eines HTML-Formulars müssen für die Auswertung durch PHP auf den Server übertragen werden. Dabei können Sie zwischen zwei Methoden wählen: Das eine ist die Methode `Post` (nein, nicht die snail-mail!), das andere die Methode `Get`. Üblicherweise verwenden Sie bei Formularen `Post`, und nur in besonderen Fällen `Get`.

Datenübertragung mit Get

Die Datenübertragung mit der Methode `Get` ist Ihnen sicherlich schon beim Surfen durch das Netz aufgefallen. Die Werte des Formulars werden an die URL zum Aufrufen der Seite, getrennt durch ein Fragezeichen, angehängt. Die Variablen/Namen der Formularelemente werden durch ein Gleichheitszeichen von den Werten getrennt. Einzelne Namen/Werte-Paare werden durch das kaufmännische UND-Zeichen (&) zusammengekettet. Eine URL, die Daten überträgt, sieht dann folgendermaßen aus:

```
http://www.domaine.de/datei.php?variablenname1=wert1&variablenname2=wert2
```

oder bezogen auf unsere Beispieldatei

```
http://www.domain.de/antwort.php?vorname=wert1&nachname=wert2
```

Dazu müssen/sollten Sie noch wissen, dass die Übertragung mit der Get-Methode einigen Restriktionen

unterliegt: Insgesamt können auf diese Weise maximal ca. 2Kbyte Daten übertragen werden, wobei Feldnamen und Trennzeichen mitgezählt werden. Weiterhin können keine Sonderzeichen, Leerstellen etc. übermittelt werden; diese Sonderzeichen werden in Hexadezimalform übertragen. Die Methode `Get` eignet sich insbesondere dazu, Daten an ein Script zu übergeben, das Sie per Hyperlink aufrufen möchten, ohne aufwändig ein Formular zu gestalten.

Info

Um die Daten des **Hyperlinks** in die benötigte Form zu bringen, sodass keine Sonderzeichen etc. enthalten sind, bietet PHP die Befehle `urldecode` und `urlencode` an. Dazu erfahren Sie u.a. in der Befehlsreferenz mehr.

Datenübertragung mit Post

Mit `Post` werden die Formulardaten als Bestandteil des gesamten Datenblocks übertragen. Die Daten werden für den Anwender nicht in der URL sichtbar, und die Datenmenge unterliegt eigentlich keiner Größenbeschränkung – eigentlich, weil die Laufzeit der Scripte aus Sicherheitsgründen meistens auf 30 Sekunden von den Providern beschränkt wird. Alle Daten, die bis dahin von dem aufgerufenen Script nicht abgearbeitet wurden, sind verloren, sobald diese Laufzeitsperren überschritten werden. Für Formulare ist im Allgemeinen immer die Methode `Post` zu empfehlen.

Daten übermitteln per URL

Sicherlich stammen nicht immer alle Daten von einem Formular. Prinzipiell lassen sich Daten auch per URL übertragen. Bei dieser Methode rufen Sie die Datei wie üblich im Browser aus, hängen den Namen der ersten Variablen an die Adresse, und zwar abgetrennt durch ein Fragezeichen und dann weisen Sie nach einem Gleichheitszeichen den Wert zu. Im Ausschnitt sieht das so aus:

```
antwort.php?vorname=Micky
```

Weitere Variablen werden mit einem kaufmännischen UND (&) angehängt.

```
antwort.php?vorname=Micky&nachname=Maus
```

Mit dem PHP-Script können Sie nun auf die Inhalte der **Variablen** zugreifen. Sie benutzen wie üblich den `echo`-Befehl und schreiben beispielsweise:

```
echo "Hallo $vorname $nachname;
```

Das PHP-Script für eine Feedback-Seite

Im Anschluss an das Formular kreieren wir ein PHP-Script, das die Daten aus dem Formular übernimmt und dem Besucher eine neue Seite anzeigt, die seine Eingaben bestätigt. Für diese Art der Verarbeitung, d.h. die Ausgabe im Browser, benutzen Sie lediglich den PHP-Befehl `echo`.

Außerdem brauchen Sie den Befehl `if` bzw. den Befehl `if-else`, um das Auswahlfeld des Formulars auszuwerten und je nach getroffener Auswahl eine unterschiedliche Reaktion auszulösen.

- 1. Starten Sie in Ihrem Editor ein neues Dokument und geben Sie als Erstes die üblichen Standard-Elemente ein, also `<html>` etc.
- 2. Ergänzen Sie das Dokument durch den PHP-Teil, indem Sie in den `<body>`-Container die PHP-Tags `<?php` und `?>` schreiben.
- 3. Nun beziehen Sie sich auf den vom Formular übermittelten Wert des Attributes `name` im Formularelement `vorname`. PHP stellt diesen Wert in einer Variablen gleichen Namens zur Verfügung; entsprechend geben Sie den zugewiesenen Wert beginnend mit einem Dollarzeichen ein. Folglich schreiben Sie:

```
echo 'Ihr Vorname ist $vorname';
```

- 4. Analog geben Sie dann ein:

```
echo 'Ihr Nachname ist $nachname';
```

- 5. Danach schreiben Sie:

```
echo 'Ihre E-Mail-Adresse ist $email';
```

Damit der Text leserlich wird, sollte noch ein Zeilenumbruch für jede Zeile angewiesen werden. Ergänzen Sie also die ‚echo-Befehle' noch durch `<br>` in jeder Zeile. Achten Sie darauf, das `<br>` jeweils mit in die Anführungszeichen zu schreiben. Das Script sieht also bis dahin folgendermaßen aus:

```
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<?php
echo 'Ihr Vorname ist $vorname<br>';
echo 'Ihr Nachname ist $nachname<br>';
echo 'Ihre E-Mail-Adresse ist $email<br>';
?>
```

Die If-Anweisung

Nun geht es an die `if`-Anweisung, die in das Script einzubauen ist. Erinnern Sie sich an das Auswahlfeld: Der Besucher der Seite kann zwischen drei Optionen wählen: sehr gut gefallen, gut gefallen und nicht gut

gefallen. Sofern die Option nicht gut gefallen eingestellt wurde, soll ein anderer Text angezeigt werden als bei den ersten beiden Optionen. Da auch noch die Option Keine Angabe zur Auswahl steht, brauchen wir einen dritten Text. Diese Verzweigung ist mit `if-elseif-else` zu erreichen. Das heißt im Klartext: ist die `if`-Bedingung `true` (wahr), dann soll geschehen, was in der Anweisung angegeben wird; ist die `elseif`-Bedingung wahr, tritt ein anderes Ereignis ein, ansonsten folgt die Anweisung im else-Block.

Werfen Sie einen Blick auf den Code unten. Sie finden hier die `if`-Anweisung, die `elseif`-Anweisung und den else-Block. Beachten Sie, dass Sie jeweils den Vergleichsoperator, also die doppelten Gleichheitszeichen benutzen müssen. Natürlich können Sie den Text variieren. Vergessen Sie aber nicht die Anführungszeichen und das Semikolon.

Info

In dem `action`-Attribut des `<form>`-Tags können Sie wie bei Hyperlinks relative und absolute Verweise verwenden, um die Datei aufzurufen, die die Formulardaten auswerten soll. Auch lässt sich mit dem Attribut `target` angeben, in welchem Frame, sofern ein Frameset vorhanden ist, die Antwortseite ausgegeben werden soll. Mit `target=_blank` rufen Sie die Antwortseite in einem neuen Browserfenster auf.

Speichern Sie das Dokument im **PHP-Format** unter der Bezeichnung, die Sie im HTML-Formular als Wert des `action`-Attributs angegeben haben. Im Beispiel war es antwort.php. Achten Sie auch darauf, es im gleichen Verzeichnis wie das HTML-Formular zu speichern.

```
<html>
<head>
<title>Kontakte</title>
</head>
<body>
<?php
echo "Ihr Vorname ist <br>$vorname<br>";
echo "Ihr Nachname ist<br>
$nachname<br>";
echo "Ihre E-Mail ist<br>
$email<br>";
echo "Ihre Note für unser Spiel ist<br>
$rank<p>";
if ($rank=="keine Angabe")
{echo "Danke für die Teilnahme";}
elseif($rank=="nicht so gut")
{echo "Schade, dass das Spiel Ihnen nicht gefallen hat";}
else
{echo "<b>Schön, dass Ihnen das Spiel gefallen hat</b>";}
?>
</body>
</html>
```

Info
Alle Werte und Elementnamen, die das Formular an die Antwortseite überträgt, werden in einem Array gespeichert, sodass Sie Formulare sogar auswerten können, wenn Sie nicht wissen, wie die Formularelemente heißen. Das Array trägt den vordefinierten Namen `$HTTP_POST_VARS`. Es handelt sich um ein assoziatives Array. Als Schlüssel werden die Namen der Formularelemente verwendet. Auf die Werte des Formularelements vorname greifen Sie also mit `$HTTP_POST_VARS[vorname]` zu. Sie können das Array aber auch Element für Element z.B. in einer `while`– oder `for`-Schleife bearbeiten.

Das Formular testen

Sie können nun probieren, ob das Formular die gewünschte Reaktion auslöst. Rufen Sie das Formular auf und tragen Sie Daten ein. Im Auswahlfeld entscheiden Sie sich für eine Option. Danach klicken Sie auf die Schaltfläche abschicken.

Funktioniert das Script, erhalten Sie nun die entsprechende Meldung, und zwar – bedingt durch die `if`-Verzweigung – entweder eine Seite wie in Bild oder eine Seite mit einem der anderen Texte.

Ein neues Script: HTML und PHP werden kombiniert

Obwohl Sie sich zu Ihrer ersten erfolgreichen dynamischen Webseite gratulieren dürfen, ist deutlich, dass noch einige Schönheitsfehler auszubügeln sind. In der jetzigen Form ist es sehr einfach. Aber im Ernstfall werden Sie dafür sorgen wollen, dass das Formular nur vollständig ausgefüllt abgeschickt wird, und der Besucher gegebenenfalls eine entsprechende Aufforderung bzw. Bitte erhält. Dies wurde bisher nicht integriert und soll nun nachgeholt werden.

Hier kommt nun eine entscheidende Überlegung hinzu: wenn das Formular unvollständig abgeschickt wird, soll eine Meldung erfolgen und der Surfer die Möglichkeit haben, in dem Formular seine Angaben erneut einzugeben. Das heißt, PHP rekurriert auf und verarbeitet bereits getätigte Eingaben, erzeugt die Fehlermeldung und soll anschließend das Formular erneut anzeigen. Daher ist es nun sinnvoll bzw. erforderlich, nicht mehr mit zwei Dateien zu arbeiten, sondern **HTML** und **PHP** in einem Dokument zu kombinieren. (Eine andere Möglichkeit wäre es, im PHP-Script die Felder des HTML-Formulars als Felder mit dem Typ `Hidden` wieder aufzunehmen). Das Resultat wird dann eine Seite sein, die sowohl die Formularfelder zeigt als auch die Feedback-Meldungen.

> Sie machen sich am wenigsten Mühe, wenn Sie das bisherige HTML-Script sowie das PHP-Script kopieren,

im Editor eine neue Datei beginnen und die Kopien in die neue Datei einfügen. Danach wird es dann um die Elemente ergänzt, die notwendig sind, um den Besucher auf eine fehlende Eingabe aufmerksam zu machen.

Sie benutzen dazu wieder eine `if`-Anweisung, denn es geht ja wieder um eine Bedingung, die wahr oder

falsch sein kann: ist das Feld ausgefüllt (`true`) oder nicht (`false`). Der im nächsten Abschnitt vorgestellte Weg ist nicht der eleganteste, wird aber fürs Erste verwendet, damit die Eingaben nachvollziehbar bleiben, obwohl die Schreibweise im Prinzip zu umständlich ist. In dem Kapitel, in dem ein Gästebuch programmiert wird, erfahren Sie dann, wie eine ähnliche Aufgabe geschickter gelöst werden kann.

Fehlende Eingaben im Script berücksichtigen

- 1. Öffnen Sie ein leeres Dokument im Editor und fügen Sie die Kopie des HTML-Scripts und des PHP-Scripts in das neue Dokument ein.
- 2. Der HTML-Teil kann fast so bleiben wie er ist. Das Attribut action im `<form>`-Tag braucht natürlich einen anderen Wert, denn Sie verweisen jetzt ja nicht auf eine zweite PHP-Datei, die die Daten verarbeitet. Geben Sie also den Dateinamen dieser Datei ein.

```
<form action="aktuelle_Datei.php"
```

Außerdem löschen Sie vor dem Beginn des eingefügten PHP-Teils die Tags, die den Body-Container und HTML schließen.

Info

Es gibt auch die Variable $PHP_SELF. In dieser Variablen steht immer der Pfad zu der aktuellen Datei. Wenn Sie also über das `action`-Attribut das Script sich selbst aufrufen lassen wollen, können Sie diese Variable verwenden. Sie müssten dann Folgendes als `action` setzen:

```
action="<?php echo $PHP_SELF; ?>"
```

Beachten Sie, dass der Befehl, da $PHP_SELF eine PHP-Variable ist, durch die **PHP-Zeichen** abgegrenzt werden muss.

- 3. Der PHP-Teil wird mit `<?php` geöffnet. Die mit `echo` beginnenden Zeilen bleiben ebenfalls wie gehabt.
- 4. Erzeugen Sie danach eine neue Zeile und schreiben Sie eine `if`-Anweisung:

```
if (!$vorname) {echo ='Bitte geben Sie einen Vornamen ein<br>';}
```

Erklärungsbedürftig ist das Ausrufezeichen vor der Variablen $vorname. Um es zu verstehen, machen Sie sich am besten noch einmal klar, was die `if`-Anweisung auf gut deutsch sagt: wenn die Bedingung `true` (wahr) ist, dann soll ein bestimmtes Ereignis eintreten. `If($vorname)` ist `true`, wenn in der Variablen beliebiger Text enthalten ist. Wir wollen aber ein Ereignis eintreten lassen, wenn die Bedingung nicht wahr ist, was der Fall ist, wenn das Feld leer bleibt. Das Zeichen für den logischen Operator NOT ist das Ausrufezeichen. Der Wert der Bedingung wird „umgedreht“: aus `true` wird `false` und aus `false` wird `true`. Durch die `if`-Anweisung in Kombination mit dem Setzen des Ausrufezeichens vor die Variable wird somit der Ausdruck `true`, wenn kein Text in der Variablen gespeichert ist. Dadurch klappt dann wieder die

gewünschte Anweisung, die ja ausgeführt wird, wenn die Bedingung wahr ist.

- 5. In die nächsten Zeilen schreiben Sie analog:

```
if (!$nachname) {echo ='Bitte geben Sie Ihren Nachnamen ein<br>';}
if (!$email) {echo ='Bitte geben Sie Ihre E-Mail-Adresse ein<br>';}
```

- 6. Sie könnten in der Klammer jetzt auch noch kombinieren, d.h. die Variablen verketten, und für entsprechende Meldungen sorgen, wenn mehr als ein Feld nicht ausgefüllt wurde. Dann hieße es beispielsweise:

```
if (!$vorname AND !$nachname) {echo='Bitte geben Sie Ihren Vor- und Nachnamen
ein<br>';}
```

- 7. Nach diesen ersten `if`-Anweisungen folgen die `if`-Bedingungen, mit denen ausgedrückt wird, dass, abhängig von dem Votum in der Auswahlliste, bestimmte Meldungen erscheinen sollen, sofern alle Felder ausgefüllt wurden. Sie sehen im Code, dass hierfür AND-Verknüpfungen verwendet wurden.

Nach den `if`-Anweisungen schließen Sie den PHP-Teil und dann den Body-Container und HTML.

Einträge in den Formularfeldern stehen lassen

Wenn Sie das Script weiter perfektionieren möchten, können Sie nun noch dafür sorgen, dass die Eingaben nach dem Abschicken in den Formularfelder angezeigt bleiben. Das hätte für den User den Vorteil, dass er, sofern er das Formular versehentlich mit einem Leerfeld abschickt, nicht noch einmal alles eingeben muss.

Sie erreichen dies damit, dass Sie den Formularelementen einen `value` zuweisen, mit dem der jeweilige Eintrag ausgegeben wird. Dies sind in dem Fall die Variablen, die den jeweiligen Inhalt der Felder enthalten. Setzen Sie den Cursor also in das erste Formularelement nach `name=vorname` und schreiben Sie:

```
Value="<?php echo $vorname;?>"
```

Die weiteren Formularelemente ergänzen Sie dann analog.

Damit haben Sie fürs Erste genug an dem Programm-Code gebastelt. Speichern Sie die Datei, aber denken Sie daran, die Datei unter dem Namen zu speichern, den Sie als Wert des Attributs `action` angegeben haben. Wenn Sie den obigen Tipp berücksichtigt haben, sorgt PHP automatisch dafür, dass die Datei sich selbst aufruft. Sie sehen in Code, dass wir die Variable $PHP_SELF verwendet haben.

```
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<? echo $PHP_SELF; ?>" method=post>
Vorname
<br>
<input type=Text name="vorname" value="<?php echo $vorname;?>" size=20>
<br>
Nachname
<br>
<input type=Text name="nachname" value="<?php echo $nachname;?>" size=20><p>
E-Mail
<br>
<input type=text name="email" value="<?php echo $email;?>" size=30>
<p><h4>Wie hat Ihnen das Spiel gefallen?</h4><p>
<select size=1 name="rank">
<Option value="keine Angabe">keine Angabe</option>
<Option value="sehr gut">sehr gut</option>
<Option value="gut"> gut</option>
<Option value="nicht so gut">nicht so gut</option>
</select>
<input type=submit name="submit" value="abschicken"></form>
<?php
echo "Ihr Vorname ist
<br>
$vorname
<br>";
echo "Ihr Nachname ist
<br>
$nachname<br>";
echo "Ihre E-Mail-Adresse ist
<br>
$email<br>";
echo "Ihre Note für unser Spiel ist
<br>
$rank<p>";
if (!$vorname) {echo 'bitte geben Sie einen Vornamen ein<br>';}
if (!$nachname) {echo 'bitte geben Sie Ihren Nachnamen ein<br>';}
if (!$email) {echo 'bitte geben Sie Ihre E-Mail-Adresse ein<p>';}
if ($vorname AND $nachname AND $email AND $rank=="keine Angabe") {echo
"<b>Vielen Dank und bis bald</b>";}
if ($vorname AND $nachname AND $email AND $rank=="sehr gut") {echo "schön, dass
Ihnen das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="gut") {echo "schön, dass Ihnen
```

```
das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="nicht so gut") {echo "schade,
dass Ihnen das Spiel nicht gefallen hat";}
?>
</body>
</html>
```

Das Script testen

Testen Sie nun das Script, indem Sie es in Ihrem Browser öffnen. Sie erhalten, ersichtlich in Bild, das Formular, die Feedback-Meldungen und darunter die Aufforderungen, die Felder auszufüllen (da sie beim Aufruf der Seite naturgemäß alle leer sind). Wir gehen mal davon aus, dass Sie sich die Seite zu Recht etwas anders vorgestellt haben. Sie müssen also noch etwas mehr an dem Script feilen.

Auf jeden Fall soll der Besucher nicht sofort mit den Meldungen konfrontiert werden, seinen Namen etc. einzugeben. Hier gibt es eine Lösung. Zunächst können Sie aber testen, ob die eingebauten Fehlermeldungen klappen. Füllen Sie die Felder aus bis auf eins, in das Sie nichts eingeben, wählen Sie eine Option im Auswahlfeld und klicken Sie auf abschicken. Sie müssten nun eine entsprechende Meldung erhalten.

Welche Werte werden bei nicht ausgefüllten Formularelementen übertragen?

Die Frage, was eigentlich übermittelt wird, wenn leere Formularfelder abgeschickt werden, lässt sich pauschal nicht beantworten, da dies je nach Art des Formularelements variiert. Ein kurzer Überblick:

`text/hidden/textarea`: Wird nichts in das Textfeld eingetragen, existiert zwar die entsprechende Variable, aber ihr ist kein Wert zugewiesen.

`checkbox/radio`: Wird kein Wert mit Hilfe des `value`-Attributs gesetzt, wird `on` übertragen, sowie die checkbox/radio-Elemente aktiviert werden. Wird das Element nicht aktiviert, wird nichts übertragen, auch der Variablenname nicht.

In dem Array `$HTTP_POST_VARS` gibt es in diesem Fall kein Element mit dem Namen der checkbox/radio-Elemente.

Meldungen unterdrücken

Öffnen Sie gegebenenfalls erneut das PHP-Script. Der Trick, die Meldungen zu unterdrücken, damit sie beim Aufruf der Seite nicht angezeigt werden, ist ein Hilfsfeld mit dem Typ `hidden` und einem beliebigen Wert. Mit dem Wert des Feldes können Sie eine `if`-Anweisung schreiben, über die geprüft wird, ob das Formular bereits abgeschickt wurde. Sie müssen also den `<form>`-Container um folgende Zeile ergänzen:

```
<input type=hidden name=sent value=1>
```

Gleich nach dem Öffnen des PHP-Teils beginnen Sie die `if`-Bedingung

```
if ($sent==1){
```

Diese Bedingung endet mit der geschweiften Klammer nach den `if`-Anweisungen für die Meldungen, dass bestimmte Eingaben fehlen, also vor dem Ende des PHP-Teils. Speichern Sie Ihr Script nach diesen Ergänzungen ab und rufen Sie es erneut auf. Sie dürften jetzt nur ein leeres Formular erhalten, da Sie mit der Variablen `$sent` gesagt haben: Gib nur Meldungen aus, wenn das Formular abgeschickt wurde.

Die Reihenfolge des Scripts umkehren

Die Seiten sind immer noch nicht ganz nach Wunsch. Als störend empfinden Sie wahrscheinlich, dass man nun – durch die Kombination von HTML mit dem PHP-Teil – ständig das Formular vor Augen hat. Ansprechender wäre es, wenn das Formular nur erscheint, um es auszufüllen oder um Fehler zu beheben und die Meldungen auf einer Seite ohne die Formularfelder (so wie es eingangs war). Diesem Problem können Sie zu Leibe rücken, indem Sie zunächst die Reihenfolge umkehren und den PHP-Teil an den Anfang des Scripts setzen, den PHP-Teil dann beenden und den HTML-Teil beginnen.

Dadurch können Sie dann gleich zu Beginn die Programmzeilen für das Feedback und die `if`-Befehle für die Fehlermeldungen in eine `if`-Anweisung stecken mit der Prüfbedingung `if(sent==1)`, mit der Folge, dass die Rückmeldungen als auch die Fehlermeldungen nur angezeigt werden, wenn das Formular abgeschickt wurde. Dann müssen Sie auch den gesamten Formularteil – also von `<form action=... >` bis `</form>` – in eine `if`-Anweisung setzen, damit er nur angezeigt wird, wenn das Formular noch nicht abgeschickt wurde, also `$sent` noch nicht existiert `(if(!sent)`.

Beachten Sie, dass der Block der `if`-Anweisung auch den Teil des Formulars beinhaltet (obwohl der PHP-Teil beendet wurde). Deshalb muss nach `</form>` die `if`-Anweisung geschlossen werden, vergessen Sie also die schließende Klammern dieses Blocks nicht und denken Sie auch daran, diese Klammer vorher wieder als PHP zu kennzeichnen.

```
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<?php
if ($sent==1){
echo "Ihr Vorname ist
<br>
$vorname<br>";
echo "Ihr Nachname ist
<br>
$nachname<br>";
```

```
echo "Ihre E-Mail-Adresse ist
<br>
$email<br>";
echo "Ihre Note für unser Spiel ist
<br>
$rank<p>";
if (!$vorname) {echo 'Bitte geben Sie einen Vornamen ein<br>';}
if (!$nachname) {echo 'Bitte geben Sie Ihren Nachnamen ein<br>';}
if (!$email) {echo 'bitte geben Sie Ihre E-Mail-Adresse ein<p>';}
if ($vorname AND $nachname AND $email AND $rank=="keine Angabe")
{echo "<b>Vielen Dank und bis bald</b>";}
if ($vorname AND $nachname AND $email AND $rank=="sehr gut")
{echo "schön, dass Ihnen das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="gut")
{echo "schön, dass Ihnen das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="nicht so gut")
{echo "schade, dass Ihnen das Spiel nicht gefallen hat";}
}
if (!$sent){
?>
<form action="<?php echo $PHP_SELF; ?>" method=post>
<input type=hidden name=sent value=1>
Vorname
<br>
<input type=Text name="vorname" value="<?php echo $vorname;?>" size=20>
<br>
Nachname
<br>
<input type=Text name="nachname" value="<?php echo $nachname;?>" size=20><p>
E-Mail
<br>
<input type=text name="email" value="<?php echo $email;?>" size=30>
<p><h4>Wie hat Ihnen das Spiel gefallen?</h4><p>
<select size=1 name="rank">
<Option value="keine Angabe">keine Angabe</option>
<Option value="sehr gut">sehr gut</option>
<Option value="gut"> gut</option>
<Option value="nicht so gut">nicht so gut</option>
</select>
<input type=submit name="submit" value="abschicken">
</form>
<?php }
?>
</body>
</html>
```

Info
In vielen Beschreibungen zu PHP wird die Submit-Schaltfläche verwendet, um zu testen, ob das Formular bereits gesendet wurde oder nicht. Dieses Verfahren spart das versteckte Eingabefeld, allerdings können Sie mit dem versteckten Eingabefeld geschickter durch das Script navigieren, wenn Sie z.B. Informationen zusammentragen möchten, die Sie auf mehreren Seiten in unterschiedlichen Formularen vom Surfer erfragen oder bei fehlerhafter Eingabe das Formular erneut anzeigen möchten. Da Sie beliebige Werte festlegen können und nicht nur auf `true` und `false` beschränkt sind, können Sie mehrere Formulare nacheinander anzeigen lassen.

Auch dieses Script ist noch nicht ganz das Gelbe vom Ei, da das Formular nun nicht mehr angezeigt wird, wenn Felder nicht ausgefüllt wurden. Dies liegt daran, dass `$sent` nach dem Abschicken des Formulars immer 1 ist, das Formular aber nur angezeigt wird, wenn `$sent` nicht existiert. Damit das Formular bei Fehlern wieder auftaucht, muss `$sent` gelöscht werden, sodass die `if`-Bedingung (`!$sent`) zum Anzeigen des Formulars wieder zutrifft.

Eine Variable löschen Sie mit dem Befehl `unset($Variablenname)`

Diesen Befehl müssen Sie also im Falle eines Fehlers/nicht ausgefüllten Feldes verwenden. Schreiben Sie in den `if`-Zweig der drei Fehlerprüfungen die zusätzliche Zeile:

```
unset($sent);
```

Eine Fehlerprüfung sieht dann folgendermaßen aus:

```
if (!$vorname)
{
echo 'Bitte geben Sie einen Vornamen ein
<br>';
unset($sent);
}
```

Info
`int unset(mixed variablen)` Die Funktion löscht eine Variable oder ein Array.

Strings bearbeiten

Die in den Feldern eingegebenen Daten interpretiert PHP als **String-Variablen**, also als aus alphanumerischen Zeichen bestehenden Text. Strings – oder Zeichenketten – können weiter bearbeitet werden, beispielsweise um Fehlern oder Missgeschicken, die bei der Eingabe von Text in ein Formularfeld entstanden sind, zu begegnen.

Leerstellen entfernen

Es passiert beispielsweise recht häufig, dass überflüssige Leerstellen mitgeschickt werden, die dann entweder die Folgeseite merkwürdig aussehen lassen, vor allem aber bei der Sammlung der Daten in einer Datenbank stören würden. Deutlich wird das Problem, das hier angesprochen wird, wenn Sie sich vorstellen, dass ein Password mit überflüssigen Leerstellen eingegeben wird und dann logischerweise nicht mehr stimmt, wenn es ohne Leerstelle eingegeben wird. Aus diesen und anderen Gründen ist es ratsam, diese Leerstellen per PHP-Befehl zu entfernen. Ins Spiel kommt dazu die Funktion `trim()` mit der Leerstellen am Anfang und Ende des Strings abgeschnitten werden. Sie können die Funktion in das vorhandene Script einbauen:

- 1. Setzen Sie den Cursor im PHP-Teil an den Anfang des `if`-Zweiges (`if ($sent==1) {`) direkt hinter der öffnenden geschweiften Klammer in eine neue Zeile und geben Sie hintereinander ein:

```
$vorname=trim($vorname);
$nachname=trim($nachname);
$email=trim($email);
```

Info
`string trim(string dertext)` Die Funktion entfernt überflüssige Zeichen am Anfang und Ende einer Zeichenkette (dertext).

Damit weisen Sie den Variablen neue Werte zu, die von den möglichen Leerstellen am Anfang und Ende befreit sind. Hierbei ist in diesem Beispiel zu beachten, dass zunächst mit `trim($vorname)` die Variable `$vorname` mit noch vorhandenen Leerstellen dem Befehl `trim` zugeführt wird. Anschließend wird der Variablen $vorname das Ergebnis des Befehls `trim` zugewiesen, sodass danach in der Variablen $vorname die Leerstellen am Anfang und Ende nicht mehr vorhanden sind.

Strings kombinieren

Auch wenn es für das eben durchgespielte Beispiel nicht unbedingt erforderlich ist, möchten wir an dieser Stelle eine weitere Möglichkeit zeigen, wie Sie Strings bearbeiten bzw. manipulieren können. Strings lassen sich auf einfache Art zusammensetzen, um so neue Variablen festzulegen. Die Schreibeweise zur Kombination von Strings ist:

```
$neuerstring=$vorhandener_string1.$vorhandener_string2;
```

In dem Beispiel wäre es denkbar, aus den beiden Strings $vorname und $nachname eine neue Variable zu generieren, sodass darüber der Vorname und der Nachname zusammen aufgerufen und gespeichert werden können.

Um die Variable zu erzeugen, geben Sie mit einem selbst gewählten neuen Variablen-Namen einfach ein:

```
$ganzername=$vorname." ".$nachname;
```

Die Elemente werden jeweils durch den Punkt getrennt. Der Punkt ist das Textverkettungszeichen oder der Verkettungsoperator. Die Anführungszeichen in der Mitte kleiden die Leerstelle ein, die dafür sorgt, dass der Vorname und der Nachname nicht zusammenkleben. Auf diese Weise könnten Sie das Script ergänzen, um den Besucher mit Vornamen und Nachnamen anzusprechen. Dazu setzen Sie die Variable wie eben beschrieben und schreiben dann beispielsweise jeweils in den Block der `if`-Anweisung:

```
if ($vorname AND $nachname AND $email AND $rank=="sehr gut") {echo "schön, dass Ihnen das Spiel gefallen hat. Danke, $ganzername";}
```

Die Daten als E-Mail verschicken

In diesem Abschnitt erklären wir nun wie angekündigt, wie Formulardaten per E-Mail verschickt werden. Für diese Aufgaben müssten Sie eigentlich nicht unbedingt PHP bemühen. Wie viele von Ihnen wissen, können Sie auch im HTML-Formular als `action mailto:Ziel` angeben. Diese Methode öffnet den E-Mail Client des Surfers mit einer neuen Nachricht, in der Ihre E-Mail-Adresse bereits als Empfänger und die Daten des Formulars in der Nachricht eingetragen sind.

Wir wollen diesen Weg hier aber nicht weiter verfolgen, weil diese Methode bei vielen Usern nicht besonders beliebt ist, denn beim Verschicken der E-Mail per Client wird die „echte“, eventuell private E-Mail-Adresse, automatisch mit übertragen – ein Vertrauensvorschuss, den viele Surfer nicht bereit sind zu leisten. Mit PHP können Sie die Formulardaten direkt per E-Mail verschicken. Der benötigte Befehl trägt den vielversprechenden Namen `mail`. Allgemein ausgedrückt lautet der Befehl:

```
mail(Empfänger, Betreff, Nachricht, Header)
```

Die einzelnen Argumente des Befehls bedürfen eigentlich keiner großen Erklärung, aber einige Informationen sind vermutlich hilfreich.

Argument	Bedeutung
Empfänger	E-Mail-Adresse des Empfängers als String. Mehrere E-Mail-Adressen können per Komma getrennt direkt aneinander geschrieben werden.
Betreff	Betreffzeile, die im E-Mail-Client angezeigt werden soll (als String).
Nachricht	Der Text der E-Mail als String. Hier sind die unterschiedlichen Formatierungsmöglichkeiten zu beachten, die die Mail-Formate (HTML, Text) bieten.

Argument	Bedeutung
Header	In diesem Bereich geben Sie der Nachricht zusätzliche Informationen mit auf den Weg, z.B. teilen Sie dem E-Mail-Client mit Content-Type: text/html mit, dass eine Nachricht im HTML-Format folgt (Content-Type: text/plain steht hingegen für reine txt-Nachrichten). Mit Importance: High aktivieren Sie die Kennzeichnung des Mail-Clients für wichtige Nachrichten, sofern der verwendete E-Mail-Client diese Eigenschaft unterstützt.

Damit Sie die Formulardaten an sich selbst per E-Mail senden können, müssen Sie sich also zunächst die Nachricht im gewünschten Format „zusammenbauen“, den Header erzeugen und anschließend die Mail auf den Weg bringen.

Um die Nachricht im HTML-Format zu verschicken, muss das vorhandene Script demnach um folgende Zeilen erweitert werden:

```
$message="<html><body>";
$message.="<p>Vorname: ";
$message.=$vorname;
$message.="<br>Nachname:";
$message.=$nachname;
$message.="<br>E-Mail.";
$message.=$email;
$message.="<br><b>";
$message.=$rank;
$message.="</b></p></body></html>";
```

Soweit der mit Hilfe von HTML formatierte Text der E-Mail-Nachricht. Jetzt folgt der Header, der sich auf einen Eintrag beschränkt: dem E-Mail-Client mitzuteilen, dass eine HTML-Mail folgt. Weitere Angaben könnten folgen, wobei wir Ihnen empfehlen, jede Angabe mit Hilfe von `\n` – dem Zeichen für den Carriage Return – Zeile für Zeile aufzubauen:

```
$header="\n Content-Type: text/html";
```

Nun fehlt noch der Versand der E-Mail. Das ist, so gut vorbereitet, ein einfacher Einzeiler:

```
mail("ihreEMail@Adresse.de", "Eine E-Mail von meiner Webseite", $message,
$header);
```

Die Mail soll aber natürlich nur dann verschickt werden, wenn die Felder des Elements auch ausgefüllt sind. Deswegen bauen wir hier wieder eine entsprechende `if`-Anweisung ein. Das Scriptfragment zum Versenden der E-Mail sieht dann folgendermaßen aus:

```
If($vorname AND $nachname AND $email)
{
$message="<html><body>";
$message.="<p>Vorname:";
$message.=$vorname;
$message.="<br>Nachname:";
$message.=$nachname;
$message.="<br>E-Mail.";
$message.=$email;
$message.="<br><b>";
$message.=$rank;
$message.="</b></p></body></html>";
$header="\n Content-Type: text/html";
mail("ihreEMail@Adresse.de", "Eine E-Mail von meiner Webseite", $message,
$header);
}
```

Info

```
bool mail(string to, string betreff, string nachricht [, zusätzlicher header])
```

Funktion zum Versenden von E-Mails. Die Klammer nimmt Argumente auf, die den Empfänger (`to`), den Betreff und die Nachricht angeben. Optional ist der Header für zusätzliche Informationen.

Die Funktion `mail` gibt `false` zurück, wenn ein Fehler beim Versand der Mail aufgetaucht ist. Diese Eigenschaft lässt sich – mit einer weiteren kleinen `if`-Anweisung kombiniert – dazu verwenden, den Surfer über den Erfolg oder Misserfolg der Datenübertragung zu informieren.

Info

Die Funktion `mail` gibt kein `false` zurück, wenn die E-Mail-Adresse des Empfängers nicht existiert und die Mail deswegen nicht zugestellt werden konnte. False wird nur ausgegeben, wenn die Übergabe an das auf dem Server installierte Mailprogramm – in den meisten Fällen Sendmail -, das sich um den Versand der Mail kümmert, nicht ordnungsgemäß funktionierte.

Die Zeile für den Versand der Mail sieht dann folgendermaßen aus.

```
if(mail("ihreEMail@Adresse.de", "Eine E-Mail von meiner Webseite", $message,
$header))
{
echo "Ihre Informationen wurden übermittelt";}
else
{
echo "Die Informationsübermittelung ist fehlgeschlagen, bitte versuchen Sie es
später noch einmal.";}
```

Wenn Sie dem User nach einer fehlgeschlagenen Übermittlung wieder das Formular anzeigen möchten, müssen Sie das Script nochmals um eine Kleinigkeit erweitern. Ergänzen Sie den obigen `else`-Zweig um eine Zeile, die die Variable `$sent` löscht:

```
Unset($sent);
```

Das gesamte **Script** sieht dann folgendermaßen aus (da das endgültige Script mittlerweile relativ kompakt ist, finden Sie weitere Erläuterungen innerhalb des Scripts in Form von Kommentaren, also in den Zeilen mit den vorangestellten Schrägstrichen // bzw. <!– im HTML-Teil):

```
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<?php
//Mit $sent==1 wird getestet,ob das Formular abgeschickt wurde
if ($sent==1)
{
//Ausgeben der eingegebenen Formulardaten
echo "Ihr Vorname ist
<br>
$vorname
<br>";
echo "Ihr Nachname ist
<br>
$nachname
<br>";
echo "Ihre E-Mail-Adresse ist
<br>
$email<br>";
echo "Ihre Note für unser Spiel ist
<br>
$rank<p>";
//Testen, ob alle Felder ausgefüllt wurden
//und gegebenenfalls Fehlermeldungen ausgeben
//und $sent löschen, damit das Formular erneut angezeigt wird
if (!$vorname) {echo 'Bitte geben Sie einen Vornamen ein<br>';
unset($sent);}
if (!$nachname) {echo 'Bitte geben Sie Ihren Nachnamen ein<br>';
unset($sent);}
if (!$email) {echo 'Bitte geben Sie Ihre E-Mail-Adresse ein<p>';
unset($sent);}
//Ausgeben der Texte für die unterschiedlichen Bewertungen des Spiels
```

```
if ($vorname AND $nachname AND $email AND $rank=="keine Angabe") {echo
"<b>Vielen Dank und bis bald</b>";}
if ($vorname AND $nachname AND $email AND $rank=="sehr gut") {echo "schön, dass
Ihnen das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="gut") {echo "schön, dass Ihnen
das Spiel gefallen hat";}
if ($vorname AND $nachname AND $email AND $rank=="nicht so gut") {echo "schade,
dass Ihnen das Spiel nicht gefallen hat";}
//Testen, ob alle Felder ausgefüllt wurden.
//Wenn ja, die E-Mail vorbereiten und verschicken
If($vorname AND $nachname AND $email)
{
//Den Nachrichtentext der E-Mail für eine HTML-Mail zusammenbauen $message="<
html>
<body>";
$message.="<p>Vorname:";
$message.=$vorname;
$message.="<br>Nachname:";
$message.=$nachname;
$message.="<br>E-Mail.";
$message.=$email;
$message.="<br><b>";
$message.=$rank;
$message.="</b></p></body></html>";
//Den Header der E-Mail erstellen
$header="\n Content-Type: text/html";
//Die E-Mail versenden.
//Wenn kein Fehler auftritt, wird eine Erfolgsmeldung ausgegeben,
//andernfalls eine Fehlermeldung
if(mail("ihreEMail@Adresse.de", "Eine E-Mail von meiner Webseite", $message,
$header))
{
//Erfolgsmeldung ausgeben
echo "Ihre Informationen wurden übermittelt";}
else
{
//Fehlermeldung ausgeben
echo "Die Informationsübermittelung ist fehlgeschlagen, bitte versuchen Sie es
später noch einmal.";
//$sent löschen, damit das Formular erneut angezeigt wird unset($sent);}}
//schließende Klammer für If($vorname AND $nachname AND $email)}
//schließende Klammer für If($sent==1)
//Das Formular anzeigen, wenn $sent nicht existiert.
//Dies ist der Fall, wenn die Seite das erste Mal aufgerufen wird,
//oder wenn $sent oben bei einem Fehler gelöscht wird.
if(!$sent)
```

```
{
?>
<form action="<?php echo $PHP_SELF; ?>" method=post>
<!-Verstecktes Feld um zu Testen,->
<!-ob das Formular aufgerufen werden soll oder nicht->
<input type=hidden name=sent value=1>
Vorname
<br>
<input type=Text name="vorname" value="<?php echo $vorname;?>" size=20>
<br>
Nachname
<br>
<input type=Text name="nachname" value="<?php echo $nachname;?>" size=20><p>
E-Mail
<br>
<input type=text name="email" value="<?php echo $email;?>" size=30>
<p><h4>Wie hat Ihnen das Spiel gefallen?</h4><p>
<select size=1 name="rank">
<Option value="keine Angabe">keine Angabe</option>
<Option value="sehr gut">sehr gut</option>
<Option value="gut"> gut</option>
<Option value="nicht so gut">nicht so gut</option>
</select>
<input type=submit name="submit" value="abschicken">
</form>
<?php
}
//schließende Klammer für If(!$sent)
//Wichtig: Diese Klammer muss in einem php-Teil stehen
?>
</body>
</html>
```

Als Adobe 1993 die Version 1.0 des Portable Document Format (PDF) auf den Markt brachte, versuchte Adobe damit ein plattformunabhängiges Format für den Austausch digitaler Dokumente zu etablieren. Mehr als zehn Jahre später kann man sagen, dass diese Strategie gut funktioniert hat. Der kostenlose Arcobat-Reader ist auf den meisten aktuellen Computersystemen installiert.

PDF schafft es, Vektorgrafik, Text und Rasterbilder in einem exakten Layout zu verbinden. PDF-Dokumente sollen unabhängig vom Betriebssystem immer gleich aussehen, egal welche Schriftarten oder Farben verwendet werden.

Clibpdf und PDFlib

Zur Erzeugung von PDF-Dokumenten können in PHP zwei verschiedene Bibliotheken verwendet werden, wenn sie bei der Installation berücksichtigt wurden: Clibpdf und PDFlib. Für beide Produkte gibt es

unterschiedliche Lizenzen, und für den privaten Gebrauch sind beide frei verfügbar. PDFlib-Lite ist eine freie Variante der PDFlib, die unter einer Open-Source-Lizenz verwendbar ist.

Info

Wer mit den Einschränkungen dieser Lizenzen nicht leben kann, hat noch die Möglichkeit, auf eine gänzlich freie Implementierung zurückzugreifen: Unter FPDF steht eine rein in PHP programmierte Bibliothek zum Erzeugen von **PDF-Dokumenten** zur Verfügung. Die Anwendung unterscheidet sich aber wesentlich von den in den folgenden Abschnitten besprochenen Möglichkeiten.

PDFlib und PDFlib-Lite

Die PDFlib ist eine Bibliothek zum Erstellen von hochwertigen PDF-Dokumenten. Seit 2000 wird sie durch die in Deutschland ansässige PDFlib GmbH vertrieben, deren Geschäftsführer und geistiger Vater Thomas Merz ist. Wer keine kommerzielle Lizenz der Bibliothek kaufen möchte und auf einige spezielle Funktionen der **PDFlib** verzichten kann (zum Beispiel auf die Verschlüsselung von Dokumenten oder auf das Kerning von Fonts), dem steht die PDFlib-Lite zur Verfügung. Die Lizenz der Open-Source-Variante weist ausdrücklich darauf hin, dass der komplette Quellcode des eigenen Programms der Öffentlichkeit zur Verfügung gestellt werden muss.

Die Unterschiede zwischen PDFlib und PDFlib-Lite und deren Lizenzen sind auf der Internetseite der PDFlib GmbH aufgeführt.

Für die Beispiele in diesem Artikel reichen die Funktionen der Lite-Version auf jeden Fall aus.

Mit dem dritten Release-Candidate von PHP 5 wurden die PDFlib-Erweiterungen aus dem Kern von PHP entfernt und in das PECL Repository übertragen. Das bedeutet, dass die PDFlib nun als Extension für PHP zur Verfügung steht und nicht mehr bei der Installation von PHP berücksichtigt werden muss. Für Administratoren reicht der Aufruf von

```
root# pear install pecl/pdflib-2.0.tgz
```

und das Modul steht für die PHP-Anwendungen zur Verfügung (natürlich muss die PDFlib zuerst installiert werden). Leider benötigt man für diesen Vorgang root-Rechte auf dem Webserver, was in den meisten Fällen nicht üblich ist. In PEAR Forum gibt es einige Tipps, wie man PEAR-Installationen auch ohne root-Rechte schafft, das würde diesen Rahmen aber sprengen. Weiters muss, um das Modul automatisch zur Verfügung zu haben, ein Eintrag in der PHP-Konfigurationsdatei erfolgen, der PHP anweist, die Erweiterung zu laden. Unter Linux könnte das zum Beispiel so aussehen:

```
extension=pdf.so
```

Wer keine Schreibberechtigung für die Konfigurationsdatei hat, kann das Modul aber auch im **PHP-Code** laden. Dabei reicht der Aufruf für das dynamische Laden der Bibliothek (dl):

```
dl("pdf.so");
```

PDF-Logo erstellen

Um die Funktionsweise der PDFlib vorzuführen, wird ein einfaches Logo erzeugt. Der Anwender hat die Möglichkeit, einen kurzen Text einzugeben (zum Beispiel den Namen des Unternehmens) und die Schriftart auszuwählen. Danach wird der Button LOGO ERZEUGEN gedrückt und das Logo erstellt. Unser PHP-Programm soll dafür sorgen, dass sich die Größe des Logos nach der Länge der Zeichenkette richtet.

Die **HTML-Eingabemaske** enthält neben dem Textfeld noch eine Auswahlliste mit Schriftarten. Damit der PHP-Code keine Probleme macht, sollte man sicherstellen, dass die Schriftarten auch zur Verfügung stehen.

```
<FORM action="create_pdflogo.php" method="POST">
Text:<INPUT type="text" name="company_name" size="20">
Font: <SELECT name="font_name">
<option value="Times">Times</option>
<option value="Helvetica" selected>Helvetica</option>
<option value="Courier">Courier</option>
</SELECT>
<INPUT type="submit" name="logo" value="Logo erzeugen">
</FORM>
```

Der PHP-Code beginnt mit den bereits bekannten Überprüfungen, ob das Formular abgeschickt worden ist und ob die eingegebene Zeichenkette nicht zu lang ist:

```
<?php // Datei pdf/create_pdflogo.php
if ($_POST['logo'] === 'Logo erzeugen'){
$company_name = $_POST['company_name'];
if (strlen($company_name) > 35){
echo 'Es können maximal 35 Zeichen eingegeben werden';
echo 'Sie haben '.strlen($company_name). ' verwendet';
exit();
}
```

Im nächsten Schritt wird das PDF-Dokument initialisiert. Der Aufruf `pdf_new()` erzeugt das PDF-Objekt und wird in der Variable `$pdf` gespeichert. Anschließend wird die Ausgabe von Warnungen unterdrückt, da ein vorzeitiges Senden der Kopfzeilen die korrekte Anzeige des PDF-Dokuments verhindert. Der Funktion `pdf_open_file` übergibt man als ersten Parameter das PDF-Objekt und als zweiten Parameter den Namen der zu erzeugenden Datei. Wird, wie im vorliegenden Beispiel, eine leere Zeichenkette übergeben, so wird das Dokument im elektronischen Speicher erzeugt, also on-the-fly. Zusätzliche Informationen über das Dokument werden mit `pdf_set_info` festgelegt. Neben dem Autor, Titel, Erzeuger und Schlüsselwörtern können auch frei definierte Eigenschaften gespeichert werden.

```
$pdf = pdf_new();
pdf_set_parameter($pdf, 'warning', 'false');
pdf_open_file($pdf, "");
pdf_set_info($pdf, "Creator", "create_pdflogo.php (PHP: ".phpversion().")");
pdf_set_info($pdf, "Author", "Bernd Oeggl");
pdf_set_info($pdf, "Title", "Logo: $company_name");
```

Um den Schriftzug auf der Seite zu zentrieren und das Seitenformat an die Schrift anzupassen, sind einige Berechnungen notwendig. Alle Längenangaben erfolgen in Postscript-Punkt, wobei 1 Postscript-Punkt ungefähr 0.3527777778 Millimetern entspricht.

```
$border = 20;
$font_size = 16;
$font = pdf_findfont($pdf, $_POST['font_name'], 'iso8859-1', 0);
$string_width = pdf_stringwidth($pdf, $company_name, $font,
$font_size);
$width = $height = $string_width+$border*2;
$center = $width/2;
```

Durch `pdf_findfont` wird die Schriftart (aus der HTML-Auswahlliste) im richtigen Encoding gesucht und der Variable `$font` zugewiesen. Der vierte Parameter (in diesem Fall 0, also FALSE) entscheidet darüber, ob die Schriftart im PDF eingebettet werden soll oder nicht. Diese Optionen sollte man nur bei der Verwendung von ausgefallenen Schriftarten in Betracht ziehen. Anschließend wird über den Aufruf an `pdf_stringwidth` die Laufweite des Schriftzugs errechnet. Da unterschiedliche Schriftarten eine unterschiedliche Laufweite aufweisen, ist es unerlässlich, diesen Wert zu ermitteln. Nachdem die wichtigsten Längen ermittelt worden sind, kann das Zeichnen beginnen:

```
pdf_begin_page($pdf, $width, $height);
pdf_setcolor($pdf, 'fill', 'gray', .9);
pdf_rect($pdf, 0, 0, $width, $height);
pdf_fill($pdf);
pdf_setcolor($pdf, 'fill', 'gray', 1);
pdf_arc($pdf, $center, $center, $string_width/2, 0, 360);
pdf_fill($pdf);
pdf_setcolor($pdf, 'fill', 'gray', .9);
pdf_arc($pdf, $center, $center, $string_width/10, 0, 360);
pdf_fill($pdf);
```

Eine neue Seite wird mit den ermittelten Werten für Breite und Höhe begonnen. In diesem Fall ist sie quadratisch. `pdf_setcolor` stellt – wie zu erwarten – die Farbe ein, wobei hier zwischen der Füllfarbe (`fill`) und der Stiftfarbe (`stroke`) gewählt werden kann; oder man setzt Beides zugleich (`both`). Die eigentliche Farbe kann in unterschiedlichen Farbräumen (gray, rgb, cmyk, lab, spot) eingegeben werden,

wobei die Lite-Version der PDFlib nur die Standardverfahren gray, rgb und cmyk unterstützt. Je nach gewähltem Farbraum werden ein bis vier Parameter (Gleitkommazahlen von 0 bis 1) benötigt, um den Farbton zu beschreiben. Nun wird der gesamte Hintergrund mit einem Grauton von 0.9 gefüllt (`pdf_rect/pdf_fill`). Ein weißer Kreis (`pdf_arc`) mit dem Radius der halben Länge der Zeichenkette wird im Zentrum platziert und von einem kleineren grauen Kreis überlagert. Der Aufruf `pdf_fill` füllt die jeweils zuvor gezeichnete Form. Anschließend wird noch der Text, jetzt in dunklem Blau, positioniert:

```
pdf_setcolor($pdf, 'both', 'rgb', 0, .2, .6);
pdf_setfont($pdf, $font, $font_size);
pdf_set_value($pdf, "textrendering", 0);
pdf_show_xy($pdf, $company_name,
$center-$string_width/2, $center-$font_size/2);
```

Der PDF-Wert `textrendering` entscheidet über das Aussehen der Schrift. So bedeutet 0 zum Beispiel, dass die Buchstaben gefüllt werden, 1 hingegen zeichnet nur die Randlinie. Durch die Positionierung in der Hälfte der Stringlänge und auf der halben Stringhöhe vom Zentrum wird der Text in `pdf_show_xy` genau in der Mitte des Dokuments gesetzt. Zum Abschluss wird das Dokument gespeichert und an den Browser geschickt:

```
pdf_end_page($pdf);
pdf_close($pdf);
$output = pdf_get_buffer($pdf);
$length = strlen($output);
header("Content-type: application/pdf");
header("Content-Length: $length");
header("Content-Disposition: inline; filename=your_logo.pdf");
echo $output;
pdf_delete($pdf);
```

`pdf_end_page` beendet die aktuelle Seite, und `pdf_close` schließt das Dokument ab (natürlich ist es auch möglich, mehrseitiges PDF zu produzieren). Da die Datei nicht auf der Festplatte gespeichert wird, legt man den Inhalt über den Aufruf `pdf_get_buffer` in einer Variable, hier `$output`, ab. Um ganz sicherzugehen, dass sich kein Browser an dem veränderten `Content-type` verschluckt, kann man noch die Größe der Datei im Feld der `Content-Length` übergeben. Nach den drei Kopfzeilen reicht ein einfaches `echo`, um den Inhalt auszugeben. Mit `pdf_delete` werden explizit alle Ressourcen freigegeben, die in Zusammenhang mit dem PDF-Objekt stehen.

Eine PDF-Rechnung mit PHP

Bei offiziellen Dokumenten wie Rechnungen oder Bestätigungen wird Wert darauf gelegt, dass der Ausdruck auch immer das gleiche Layout aufweist. HTML ist hier nicht ausreichend, da die gleiche Seite von unterschiedlichen Browsern nicht immer gleich ausgedruckt wird. Hier kann PDF seine Stärken ausspielen, da die Objekte millimetergenau platziert sind.

Im folgenden Beispiel wird ein Web-Interface für Außendienstmitarbeiter einer Firma erstellt. Nachdem ein Techniker seine Arbeit vor Ort erledigt hat, meldet er sich an der Webseite seiner Firma an und gibt dort die gerade erledigte Arbeit ein. Aus diesen Informationen wird eine Rechnung als PDF-Dokument erzeugt, die der Mitarbeiter direkt ausdruckt und unterschreibt. Das allein würde den Aufwand noch nicht rechtfertigen, könnte doch der Techniker auch ein ausgedrucktes Formular bei sich haben und es auf herkömmliche Weise, analog sozusagen, ausfüllen.

Der zusätzliche Nutzen entsteht dadurch, dass die Daten zur Arbeitszeit und zum Auftrag des jeweiligen Mitarbeiters in der zentralen Datenbank der Firma gespeichert werden. Das erspart nicht nur den Mitarbeitern ein erneutes Ausfüllen von Stundenlisten, sondern ermöglicht auch den boomenden Human-Ressource-Management-Abteilungen einfachen Zugriff auf die zu optimierenden Daten der MitarbeiterInnen.

Die Webschnittstelle

Beim Ausfüllen der Rechnung hat der Mitarbeiter die Möglichkeit, unterschiedliche Arbeiten einzugeben, die am Ende summiert werden. Sind alle Tätigkeiten eingetragen, wird das Webformular abgeschickt und das PDF-Dokument erzeugt.

job_accounting.php enthält in erster Linie den HTML-Code, der das Formular zur Eingabe der Stunden bereitstellt. Bereits eingegebene Tätigkeiten werden auf der Webseite angezeigt und in einer versteckten Variable mitgeführt.

Der PHP-Quelltext für die Webschnittstelle enthält wenig Besonderheiten:

```
<?php
// aktuelles Datum und Zeit berechnen:
$start = localtime(time()-3600, 1);
$end = localtime(time(), 1);
if (isset($_POST['add']) || isset($_POST['done'])){
$work_done = unserialize(urldecode($_POST['work_done']));
}
```

Zu Beginn werden eine Startzeit (`$start`) und eine Endzeit (`$end`) berechnet, um die Formularfelder teilweise im Voraus auszufüllen. Die Startzeit wird dabei auf eine Stunde vor der aktuellen Zeit gestellt (`time()-3600`). Egal welcher Button angeklickt wurde (ADD oder DONE), zu Beginn muss der Inhalt der versteckten Variable `$work_done` deserialisiert werden. Die Methode des Serialisierens bietet sich an, um eine komplexe Variable in eine speicherbare Textform zu bringen. In diesem speziellen Fall muss der Inhalt zuerst noch decodiert werden (`urldecode`), da Sonderzeichen bei einem HTTP-Request verloren gehen.

```
if (isset($_POST['add'])){
$duration = (mktime($_POST['end_hour'],
$_POST['end_min']) - mktime($_POST['begin_hour'],
```

```
$_POST['begin_min']))/60;
$work_done[] = array ($_POST['category'],
$_POST['begin_hour'],
$_POST['begin_min'], $_POST['end_hour'],
$_POST['end_min'],
$_POST['anno'], $duration);
}
else if (isset($_POST['done'])){
// hier kann die Arbeit in die Datenbank gespeichert werden
require_once 'PdfInvoice.php';
$invoice = new PdfInvoice();
$invoice->init('Firstname Lastname', '99',
explode("\n", $_POST['client']), $work_done);
$invoice->show();
}
?>
```

Soll ein Eintrag hinzugefügt werden (`$_POST['add']`), wird in die Array-Variable `$work_done` ein weiteres Element eingetragen. Ist die Rechnung fertig und soll das PDF-Dokument generiert werden, wird eine Instanz der Klasse `PdfInvoice` erzeugt. In der `init`-Methode werden die Daten für die Rechnung übergeben, und anschließend wird das PDF mit `show()` ausgegeben.

Im HTML-Code fehlt noch das Serialisieren und Übergeben der versteckten Variable mit den bereits eingetragenen Arbeiten. Um beim **HTTP-Versand** keine Zeichen zu verlieren, wird der serialisierte String dem Standard konform kodiert. Diese Kodierung entspricht im Wesentlichen den Vorgaben im RFC1738.

```
<input type="hidden" name="work_done" value='<?php
echo urlencode(serialize($work_done));
?>'>
```

Natürlich kann man hier auch mit Cookies arbeiten, um die Informationen zu transportieren. In unserem Beispiel sitzt der Außendienstmitarbeiter aber an einem fremden Computer, auf dem keine Spuren hinterlassen werden sollen. Das betrifft auch Cookies, die der Browser auf der Festplatte speichert.

Die Klasse PdfInvoice

Während die Logik des Programms in der Datei job_accounting.php liegt, erzeugt die Klasse PdfInvoice.php die PDF-Rechnung. Dadurch besteht die Möglichkeit, die Klasse auch aus einer anderen Anwendung heraus aufzurufen. Da die meisten PDF-spezifischen Funktionen schon im vorangegangenen Beispiel beschreiben worden sind, werden hier nur neue Funktionen vorgestellt.

Zu Beginn werden Variablen definiert, die das Aussehen der Rechnung bestimmen, wie zum Beispiel die Dimensionen einer DIN-A4-Seite in **Postscript-Punkt** und die nach DIN-5008 vorgesehene Position des

Adressfeldes auf der Seite:

```
// DIN-A4-Seite
var $width = 595.3;
var $height = 842;
// DIN-5008-Angaben für das Adressfeld
var $addr_x = 56.7;
var $addr_y = 700.3;
var $addr_width = 241; // 85 mm
var $addr_height = 113.4; // 40 mm
```

Die variablen Inhalte der Rechnung werden der `init()`-Funktion übergeben, die im nächsten Schritt Hilfsfunktionen aufruft, um die Elemente der Rechnung zusammenzusetzen:

```
function init($creator, $invoice_nr, $client, $work){
$this->pdf = pdf_new();
$this->creator = $creator;
pdf_open_file($this->pdf, "");
pdf_set_parameter($this->pdf, 'warning', 'false');
pdf_set_info($this->pdf, "Creator", "pdf_invoice.php (PHP ".phpversion().")");
pdf_set_info($this->pdf, "Author", $this->company .": $creator");
pdf_set_info($this->pdf, "Title", "Invoice: $invoice_nr");
pdf_begin_page($this->pdf, $this->width, $this->height);
$this->_draw_tics();
$this->_write_letterhead();
$this->_write_address($client);
$this->_draw_work_table($work, $invoice_nr);
$this->_draw_footer();
}
```

Hier ist ein Ansatzpunkt für Verbesserungen des Beispiels: Soll die Ausgabe der Rechnung möglichst flexibel bleiben, so muss das Einbinden von Hilfsfunktionen wie `_draw_tics()` optional sein. Das kann zum Beispiel durch das Setzen einer Statusvariable in der **PdfInvoice**-Klasse erreicht werden, oder man gliedert den Funktionsaufruf an `_draw_tics` aus der `init`-Funktion aus.

Eine sehr kurze, aber nützliche Hilfsfunktion im vorliegenden Beispiel ist `_line_from_to`(x, y, x2, y2). Da in diesem Beispiel mehrere Linien gezeichnet werden, verkürzt sich der Aufruf von drei Funktionen der PDF-Bibliothek auf eine interne Hilfsfunktion:

```
function _line_from_to($x,$y,$x2,$y2){
pdf_moveto($this->pdf, $x, $y);
pdf_lineto($this->pdf, $x2, $y2);
```

```
pdf_stroke($this->pdf);
}
```

Eine neue Funktion aus der PDF-Implementierung wird beim Setzen der Adresse verwendet:

```
pdf_show_xy($this->pdf, $client[0], $x, $y);
for ($i=1; $i<count($client); $i++){
pdf_continue_text($this->pdf, $client[$i]);
}
```

Die Array-Variable `$client` enthält pro Element eine Zeile der Adresse des Kunden. Die erste Zeile wird mit `pdf_show_xy` an die angegebene Position gesetzt, und die folgenden Zeilen werden innerhalb der `for`-Schleife mit `pdf_continue_text` platziert. Dabei kümmert sich die Funktion selbst um die x- und y-Position des Textes und erspart dem Programmierer das lästige Berechnen der Zeilen und Zeichenhöhe.

Das Firmenlogo, ein Bild im PNG-Format, wird im Kopf der Rechnung platziert. Die PDFlib-Funktionen zum Einbinden von Grafiken sind sehr einfach:

```
function _write_letterhead(){
$letterhead = pdf_open_image_file($this->pdf, 'png',
$this->letterhead_logo);
pdf_place_image($this->pdf, $letterhead, 0,
$this->height-100, 0.6);
}
```

Nachdem das Bild geöffnet und der Variable `$letterhead` zugewiesen worden ist, positioniert `pdf_place_image` das Bild an den angegebenen Koordinaten (der Ursprung des Koordinatensystems ist die linke untere Ecke) mit der gewünschten Skalierung (0.6 in diesem Beispiel).

Verbesserungsvorschläge und Erweiterungen

Das Beispiel stellt nur die Minimalversion dieser Anwendung dar. Interessante Erweiterungen wären in mehreren Bereichen denkbar:

- Wie bereits erwähnt wurde, kann das Aussehen der Rechnung flexibler gestaltet werden, um die Klasse universeller einsetzbar zu machen. Dazu wäre es nötig, die Aufrufe an die internen Funktionen (`_draw_tics` oder `_write_letterhead`) nur auf Wunsch auszuführen. Dazu könnten Sie zum einen die Aufrufe aus der `init`-Funktion herausnehmen und diese Arbeit der aufrufenden Applikation überlassen. Standardmäßig wäre die Funktion somit deaktiviert. Zum anderen könnte die Ausführung über Variablen gesteuert werden, die den Status Ein oder Aus darstellen. Dadurch kann der Defaultstatus auf Ein sein, und die Applikation kann die Funktion über das Setzen der Variable auf Aus deaktivieren.

- Die Eingabe der Kundenadresse sollte mit der Kundenkartei abgeglichen werden. Ist ein Kunde bereits in der Adressdatei vorhanden, soll die Adresse aus der Datenbank ausgelesen werden; ist es ein Neukunde, wird die Datenbank um die Adresse ergänzt.
- Ein weiterer Mangel ist die Ausgabe von Kommentaren des Außendienstmitarbeiters auf der PDF-Rechnung. Ist die Zeichenkette zu lang, gibt es keinen automatischen Zeilenumbruch in der Tabelle. Hier wäre es zumindest notwendig, die Länge der Zeichenkette (`pdf_stringwidth`) mit der Breite der Spalte zu vergleichen und den Text im Zweifelsfall zu kürzen. Natürlich ist es auch möglich, den Text aufzuteilen und in mehrere Zeilen zu schreiben. Dabei darf man nur die Y-Position nicht aus den Augen verlieren, da sie zum weiteren Setzen der Rechnung wichtig ist.

Will man die Rechnung sehr flexibel gestalten, muss man irgendwann den Seitenumbruch in Bedacht ziehen. Werden zu viele einzelne Arbeiten aufgelistet, sollte eine neue Seite erzeugt werden, die die Rechnung vervollständigt. Für das hier angeführte Beispiel ist das wohl nicht unbedingt notwendig.

Das Ausstellen der Rechnung sollte natürlich nur für Mitarbeiter der Firma möglich sein. Um das zu garantieren, muss die Seite mit einem Passwort geschützt werden. Dazu bieten sich entweder die in String-Funktionen vorgestellten Möglichkeiten zur Validierung an, oder man verwendet eine Funktion des Webservers (`htaccess` bei Apache). Der wahre Benutzername kann dann aus der Variable `$_SERVER['PHP_AUTH_USER']` an die `init`-Funktion der PdfInvoice-Klasse übergeben werden.

PHP-Scripts erhalten die Standarddateiendung **.php**. Je nachdem, was in der Konfiguration des Webservers Apache bei **AddType application/x-httpd-php** angegeben wurde, sind auch andere dort genannte Dateiendungen erlaubt. PHP-Scripts, die mittels URI, also etwa im Browser, aufrufbar sein sollen, müssen unterhalb der Document Root abgelegt werden, also ebenso wie HTML-Dateien, die über einen Webserver aufrufbar sein sollen.

Auf Linux/Unix-Systemen sollten PHP-Dateien und die Verzeichnisse, in denen sie abgelegt sind, stets auf „ausführbar“ gesetzt werden, und zwar in der Regel für alle Berechtigungstypen (Dateieigentümer, seine Primärgruppe und Rest der Welt). Eine typische Wahl ist `chmod 755`.

In diesem Abschnitt stellen wir bereits einige praxistypische Beispiele für den Einsatz von PHP vor, ohne dabei jedoch näher auf die Sprachdetails einzugehen. Ziel ist es, ein Gefühl dafür zu bekommen, wie PHP einsetzbar ist und wie PHP-basierte Websites strukturiert werden können. Verzweifeln Sie nicht, wenn Sie einzelne Passagen in den hier vorgestellten Beispielen zu diesem frühen Zeitpunkt noch nicht ganz verstehen. Auf syntaktische Details gehen wir erst später ein.

PHP in HTML

Bereits im ersten Testscript haben wir gesehen, dass sich PHP problemlos in HTML einbetten lässt. Im Gegensatz zu JavaScript kann PHP sogar an beliebigen Stellen im HTML-Code eingebettet werden, d.h., die Syntax von HTML darf durch PHP-Bereiche verletzt werden. Wichtig ist nur, dass am Ende fehlerfreies HTML an den Browser gesendet wird. Nachfolgendes Beispiel zeigt, wie PHP-Code an den unterschiedlichsten Stellen innerhalb eines HTML-Dokuments nützliche Dienste leisten kann:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<meta http-equiv="expires" content="<?php
$in_3_days = mktime(date("h"),date("i"),date("s"),date("n"),
date("j"),date("Y")) + (72 * 60 * 60);
echo gmdate("D, d M Y H:i:s", $in_3_days)," GMT";
?>">
<title>Bilder</title>
<style type="text/css">
td.thumb {
vertical-align:top;
padding-right:15px;
padding-bottom:10px;
}
td.info {
vertical-align:top;
}
img.thumb_h {
width:120px;
height:96px;
border:solid 1px black;
}
img.thumb_v {
width:96px;
height:120px;
border:solid 1px black;
}
h1 {
border-bottom:1px solid black;
}
</style>
</head>
<body>
<h1>Bilder</h1>
<table>
<tr>
<td class="thumb">
<a href="bild.php?bild=01">
<img src="bild01thumb.jpg" class="thumb_h" alt="Bild 1 Vorschau"></a>
</td>
<td class="info">
<b>Bild 1:</b>
<br>
```

```
<?php
$bytes = filesize("bild01.jpg");
$pixel = GetImageSize("bild01.jpg");
echo "Breite × Höhe: ",
$pixel[0]," × ",$pixel[1]," Pixel<br>";
echo "Größe: ",round($bytes / 1024)," KByte";
?>
</td>
</tr>
<tr>
<td class="thumb">
<a href="bild.php?bild=02">
<img src="bild02thumb.jpg" class="thumb_v"
alt="Bild 2 Vorschau"></a>
</td>
<td class="info">
<b>Bild 2:</b>
<br>
<?php
$bytes = filesize("bild02.jpg");
$pixel = GetImageSize("bild02.jpg");
echo "Breite × Höhe:",
$pixel[0]," × ",$pixel[1]," Pixel<br>";
echo "Größe: ",round($bytes / 1024)," KByte";
?>
</td>
</tr>
<tr>
<td class="thumb">
<a href="bild.php?bild=03">
<img src="bild03thumb.jpg" class="thumb_h"
alt="Bild 3 Vorschau"></a>
</td>
<td class="info">
<b>Bild 3:</b>
<br>
<?php
$bytes = filesize("bild03.jpg");
$pixel = GetImageSize("bild03.jpg");
echo "Breite × Höhe:",
$pixel[0]," × ",$pixel[1]," Pixel<br>";
echo "Größe: ",round($bytes / 1024)," KByte";
?>
</td>
</tr>
</table>
```

```
<?php
echo "<p style=\"border-top:solid 1px black;\">";
echo $_SERVER['SERVER_NAME'],"</p>";
?>
</body>
</html>
```

Info
Das Beispiel gibt ein paar anklickbare Vorschaubilder aus (auch als **Thumbnails** oder **Thumbs** bezeichnet), die zum Betrachten ihrer größeren Varianten einladen sollen. Neben jedem Vorschaubild stehen Informationen über Bildausmaße und Dateigröße des Originalbilds. Diese Informationen stehen jedoch nicht fest im HTML-Quelltext, sondern werden von PHP dynamisch ermittelt. Würde also etwa eines der aufrufbaren Bilder durch eine andere Version ausgetauscht, so würden beim Aufruf der Seite jederzeit die aktuellen Daten zum Originalbild angezeigt.

PHP mitten im HTML-Tag: die dynamische Meta-Angabe

Bevor wir näher auf die Passagen mit den Vorschaubildern eingehen, erst einmal zum ersten notierten PHP-Bereich im HTML-Quelltext. Dieser befindet sich in den Kopfdaten und zwar innerhalb der Wertzuweisung an das `content`-Attribut eines Meta-Tags:

```
<meta http-equiv="expires" content="<?php
$in_3_days = mktime(date("h"),date("i"),date("s"),date("n"),
date("j"),date("Y")) + (72 * 60 * 60);
echo gmdate("D, d M Y H:i:s", $in_3_days)," GMT";
?>">
```

Info
Aus HTML-Sicht stellt dies eine syntaktische Verletzung dar, da innerhalb von Wertzuweisungen an Attribute HTML-eigene Zeichen wie < oder > nicht unmaskiert vorkommen dürfen. Da der PHP-Bereich jedoch nach Ausführung des Scripts durch seinen Output ersetzt wird, erscheint am Ende HTML-konformer Inhalt an der betroffenen Stelle. In der Quelltextansicht des an den Browser übertragenen HTML-Dokuments steht dann beispielsweise:

```
<meta http-equiv="expires" content="Mon, 07 Mar 2005 10:06:37 GMT">
```

Info
Die Zuweisung an das `content`-Attribut wurde dabei von PHP ermittelt. Das Script erzeugt als Ausgabe einen formatgerechten String, der den Zeitpunkt genau drei Tage nach Aufruf der Seite darstellt. Auf diese Weise erhält die Meta-Angabe zum Cache-Verfallszeitpunkt eine relative Angabe, nämlich „ignoriere das Dokument im Cache, wenn es dort älter als 3 Tage ist“, aber die Zeitangabe selbst ist eine `absolute`.

Ermittelt wird die gewünschte Ausgabe mithilfe der PHP-Funktionen

`mktime()` und `gmdate()`. Auf deren Eigenheiten und die Bedeutung ihrer Parameter gehen wir später näher ein. An dieser Stelle soll ein kurzer Abriss dessen genügen, was innerhalb des Scripts passiert: In einer Variablen namens `$in_3_days` wird der Rückgabewert der Funktion `mktime()` gespeichert. Variablen in PHP sind immer an dem führenden Dollarzeichen `$` zu erkennen. Die Parameter, die `mktime()` übergeben werden, bewirken, dass die Funktion den Zeitpunkt „in drei Tagen, von jetzt an gerechnet“ zurückgibt.

Den Zeitpunkt benötigen wir, um ihn der Funktion `gmdate()` zu übergeben. Mithilfe von `gmdate()` kann ein Zeitpunkt in Bestandteile wie Wochentag, Monat, Jahr, Uhrzeit usw. zerlegt werden. Die Funktion ermöglicht es, eine formatierte Ausgabe in gewünschter Form zu erzeugen. Das Ergebnis, das die Funktion zurückgibt, wird mit `echo` ausgegeben. „Ausgegeben“ heißt, es wird im HTML-Code, der an den Browser gesendet wird, an der Stelle eingefügt, wo der PHP-Bereich notiert ist.

PHP für dynamisch erzeugten Text: die Angaben zu den Bildern

Die Vorschaubilder im Beispiel stehen in einer unsichtbaren Tabelle, links die Vorschaubilder und rechts davon die Informationen zum Bild. Bei jedem der drei Bilder im Beispiel ist in der rechten Tabellenzelle ein gleichartiger PHP-Bereich notiert. Hier noch mal der Ausschnitt für das erste Bild:

```
<b>Bild 1:</b>
<br>
<?php
$bytes = filesize("bild01.jpg");
$pixel = GetImageSize("bild01.jpg");
echo "Breite × Höhe: ",
$pixel[0]," × ",$pixel[1]," Pixel<br>";
echo "Größe: ",round($bytes / 1024)," KByte";
?>
```

Info
Die erste Zeile ist noch statisches HTML. Danach folgt der PHP-Bereich. Die Bedeutung der ersten Anweisung ist leicht erratbar: Mithilfe der PHP-Funktion `filesize()` wird die Dateigröße der Grafik ermittelt, die angezeigt werden soll, wenn der Anwender auf das Vorschaubild klickt. Der Rückgabewert der Funktion, die Dateigröße in Byte, wird in einer Variablen namens `$bytes` gespeichert.

Die nächste Anweisung ist nach dem gleichen Schema aufgebaut: In einer Variablen `$pixel` wird der Rückgabewert der PHP-Funktion `GetImageSize()` gespeichert. Auch dieser Funktion wird wieder die Datei der Grafik übergeben, die bei Anklicken des Vorschaubilds angezeigt werden soll. Im Unterschied zu `filesize()` gibt `GetImageSize()` jedoch keinen einzelnen Wert zurück, sondern gleich mehrere Werte – in einem Array. In `$pixel` ist also am Ende ein Array gespeichert.

Damit sind alle gewünschten Daten ermittelt und sie können ausgegeben werden.

Von den Werten, die in `$pixel` gespeichert sind, interessieren uns nur die ersten beiden (Breite und Höhe). Diese geben wir mit `$pixel[0]` und `$pixel[1]` zusammen mit anderem HTML-Text aus. Auch die Dateigröße wird ausgegeben, allerdings nicht wie zunächst ermittelt in Byte, sondern in Kilobyte. Um den in `$bytes` gespeicherten Wert in Kilobyte auszudrücken, muss er durch 1024 dividiert werden, weil ein Kilobyte aus 1024 Byte besteht. Um sicherzustellen, dass eine Ganzzahl angezeigt wird, übergeben wir die Rechnung an die PHP-Funktion `round()`.

Dieses Beispiel zeigt, wie viele fertige Funktionen zu PHP gehören, die man sonst in Programmiersprachen oft vergeblich sucht bzw. aus Basisfunktionen selbst programmieren oder aus zusätzlichen Code-Bibliotheken besorgen muss. Hinter `GetImageSize()` etwa steckt eine Menge Intelligenz, nämlich die Header-Daten verschiedener Grafikformate auslesen und daraus bestimmte Werte ermitteln zu können.

PHP für die Ausgabe von globalen Daten: die Ausgabe des Servernamens

Am Ende unseres Beispiels ist noch ein kleiner PHP-Bereich notiert, in dem mit `$_SERVER['SERVER_NAME']` der Hostname des Rechners ausgegeben wird, auf dem der Webserver läuft. Diese Variable gehört zu den vordefinierten und global verfügbaren Variablen in PHP. Es gibt eine ganze Reihe solcher Variablen, unter anderem sämtliche Umgebungsvariablen des Webservers. Aber auch GET-Strings und POST-Daten kann das Script über solche Variablen ermitteln. Es handelt sich um die so genannten **Superglobals**, ein Set von Arrays, in denen Umgebungsdaten gespeichert sind.

Fazit: PHP in HTML ist praktisch

Das Beispiel zeigt, warum PHP so beliebt ist. Ohne viel Aufwand lassen sich HTML-Dokumente aufpeppen oder optimieren. Die Strukturierung einer Website bleibt die gleiche wie beim Arbeiten mit statischen HTML-Dateien. Es gibt eine Reihe von Dateien, die untereinander verlinkt sind.

Das Kapitel über HTML und CSS hat gezeigt, wie sich durch gezielten Einsatz von CSS Inhalt und Layout von Webseiten sinnvoll trennen lassen. Mit PHP können wir noch ein Stück weiter gehen: Wir trennen nicht nur HTML-Struktur und CSS, sondern auch HTML-Grundgerüst, Navigation und Seiteninhalte. Dadurch erreichen wir eine noch sauberere Trennung der Bereiche mit erheblichen Vorteilen:

- Das HTML-Grundgerüst, das ja zumindest die Basiselemente für das Layout mit CSS enthält, muss ebenso wie die CSS-Definitionen nur ein einziges Mal notiert werden, und nicht, wie bei statischen Seiten, in jeder einzelnen HTML-Datei. Selbst bei radikalen Layout-Relaunches müssen dann nur noch die HTML-Grundgerüstdatei und die CSS-Datei bearbeitet werden.
- Die Navigation muss ebenfalls nur noch an einer Stelle notiert werden und nicht mehr in jeder einzelnen Seite.
- Die eigentlichen Seiteninhalte bestehen aus schlichtem HTML und können durch kleine Änderungen im PHP-Script aus ganz variablen Quellen stammen. In unserem Beispiel wird es für jede Seite eine Inhaltsdatei geben. Ebenso gut könnten die Inhalte aber aus einer Datenbanktabelle, aus einer XML-Datei oder aus einer anderen Quelle stammen.

Auch im Template-Beispiel werden wir anstelle von statischen URIs wie schon im Abschnitt zuvor wieder mit dynamischen URIs arbeiten, d.h. also den anzuzeigenden Seiteninhalt aus dem übergebenen GET-String ermitteln. Eine einzige PHP-Datei ist der „Container“ für sämtliche möglichen Inhalte.

Insgesamt benötigen wir also folgende Dateien:

- Eine Datei mit dem HTML-Grundgerüst ohne Inhalte,
- eine Datei mit dem HTML-Code der Navigation,
- eine CSS-Datei für zentrale Layout- und Formatierungsdefinitionen,
- je eine Datei für den Inhalt jeder Seite,
- ein PHP-Script, das alles zusammenklebt und die gewünschten Inhalte ausgibt.

Unser Template-Beispiel greift auf das Beispiel aus „Winkel-Layout“ zurück.

HTML-Templates: Grundgerüst und Navigation

Die Datei mit dem HTML-Grundgerüst nennen wir im Beispiel linkweb.tpl. Der Dateiname ist natürlich frei wählbar. Die Endung .tpl vergeben wir einfach als Abkürzung für „Template“. Der Quelltext lautet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>[%title%]</title>
<link rel="stylesheet" type="text/css" href="linkweb.css">
</head>
<body>
<div id="top_section">
<img src="logo.gif" id="logo" alt="Logo">
<div id="ticker"></div>
<form id="search" action="search.php" method="get">
<span id="search_label">Suche nach:</span>
<input type="text" id="search_text" name="search_text" value="Suche">
<input type="submit" src="ok.gif" id="search_button" value="OK">
</form>
</div>
<div id="main_section">
<div id="navigation">
[%navigation%]
</div>
<div id="content">
[%content%]
```

```
</div>
</div>
</body>
</html>
```

Info
Es handelt sich um eine gewöhnliche HTML-Datei, welche das HTML-Grundgerüst mit den nötigen Kopfdaten enthält sowie die `div`-Bereiche, die für CSS zur optischen Bereichsverteilung im Browser dienen. Auffällig sind die Zeichenfolgen wie z.B. `[%content%]`. Dies sind selbst definierte Platzhalter, die wir später in PHP durch ermittelte Inhalte ersetzen werden.

Noch trivialer ist die Template-Datei für die Navigation – wir nennen sie **linkweb-nav.tpl**:

```
<a class="navi" href="linkweb.php?page=home">HOME</a>
<a class="navi" href="linkweb.php?page=impressum">Impressum</a>
<a class="navi" href="linkweb.php?page=themen">Themen</a>
<a class="navi" href="linkweb.php?page=branchen">Branchen</a>
<a class="navi" href="linkweb.php?page=auskunft">Auskunft</a>
<a class="navi" href="linkweb.php?page=geo">Länder und Städte</a>
<a class="navi" href="linkweb.php?page=wissen">Wissenschaft</a>
<a class="navi" href="linkweb.php?page=literatur">Literatur</a>
<a class="navi" href="linkweb.php?page=lexika">Lexika/Wörterbücher</a>
<a class="navi" href="linkweb.php?page=glossare">Glossare</a>
<a class="navi" href="linkweb.php?page=wikis">Wikis</a>
<a class="navi" href="linkweb.php?page=ftp">FTP-Verzeichnisse</a>
<a class="navi" href="linkweb.php?page=wais">WAIS-Services</a>
<a class="navi" href="linkweb.php?page=newsgroups">Newsgroups</a>
<a class="navi" href="linkweb.php?page=foren">Web-Foren</a>
```

Info
Die Datei enthält ausschließlich die Hyperlinks der Navigation. Die Formatierung der Navigationsleiste, welche aus diesen Links gebildet wird, wird in der CSS-Datei vorgenommen. Deren Quelltext sparen wir uns jedoch, weil sie nichts enthält, was im Zusammenhang mit dem Funktionieren des Beispiels von Bedeutung ist.

Alle Navigationslinks rufen das zentrale, als „Container“ fungierende PHP-Script `linkweb.php` auf, und zwar jeweils mit einem GET-String, in dem ein Parameter namens `page` einen jeweils unterschiedlichen Wert erhält.

Datendatei: schlicht und einfach

Zum besseren Verständnis noch der Quelltext einer Datendatei, also einer Datei mit eigentlichen Seiteninhalten. Wie erwähnt speichern wir die Seiteninhalte in unserem Beispiel in jeweils eigenen

Dateien, die wir 0001.txt, 0002.txt, 0003.txt usw. benennen. Die Inhalte könnten aber auch in anderer Form gespeichert werden. Der Inhalt der 0001.txt lautet beispielsweise:

```
<h1>LinkWeb</h1>
<p>Bevor es große Suchmaschinen wie Google gab, waren es Linkverzeichnisse,
welche die zentrale Vernetzung im Web besorgten. Auch heute noch, im Zeitalter
der Suchmaschinen, gibt es nach wie vor gepflegte Linkverzeichnisse, meist auf
einen bestimmten Themenbereich beschränkt. Dazu kommen durchsuchbare
Spezialverzeichnisse, etwa für Adressen und Telefonnummern, aber auch für
Internet-Inhalte jenseits des World Wide Web.</p>
<p><b>LinkWeb</b> ist das Verzeichnis solcher Verzeichnisse.</p>
<h2>Die Kategorien</h2>
<ul>
<li>Themenverzeichnisse sind Linkverzeichnisse zu bestimmten Fachgebieten.</li>
<li>Branchenverzeichnisse sind nach Branchen sortierte Verzeichnisse von
Unternehmen und Geschäften.</li>
<li>Auskunft-Services sind durchsuchbare Verzeichnisse wie Telefonbücher.</li>
<li>usw.</li>
</ul>
```

Info
Datendateien dieser Art enthalten nichts als strukturelles HTML für die darzustellenden Inhalte. Keine Spur von Layout und auch das HTML-Grundgerüst mit Kopfdaten usw. ist nicht nötig, da es ja in eine Template-Datei ausgelagert wurde. Wichtig ist für unser Beispiel nur eine Konvention: In der ersten Zeile einer Datendatei muss eine h1-Überschrift mit dem Titel der Seite notiert sein.

Das PHP-Script: Zusammenkleben und Ausgeben der Inhalte

Die zuvor behandelten Listings lassen bereits erkennen, welche Aufgaben das zentrale PHP-Script hat:

- Es muss die Template-Datei mit dem HTML-Grundgerüst einlesen.
- Es muss die Datei mit der HTML-Navigation einlesen.
- Es muss den Code der in Schritt 2 eingelesenen Datei innerhalb des Codes der in Schritt 1 eingelesenen Datei dort einfügen, wo der Platzhalter `[%navigation%]` notiert ist.
- Es muss den übergebenen GET-String auswerten, um herauszufinden, welche Seite angezeigt werden soll. Dazu muss es natürlich auch eine Liste möglicher Seiten kennen oder wissen, wie es an die Daten von Seiteninhalten herankommt.
- Es muss den Inhalt der gewünschten Datendatei einlesen und dann an der Stelle einfügen, wo der Platzhalter `[%content%]` notiert ist.
- Es muss der vereinbarten Konvention gemäß die h1-Überschrift der ersten Zeile der Datendatei auslesen und ihren Textinhalt dort einfügen, wo der Platzhalter `[%title%]` notiert ist. Das ist im `title`-Element in den Kopfdaten des Grundgerüsts der Fall.
- Es muss die so zusammengefügte Seite ausgeben.

Der Quelltext des PHP-Scripts lautet:

```
<?php
#-----------------------------------------
# Templates:
$layout_template = "linkweb.tpl";
$navigation_template = "linkweb-nav.tpl";
#-----------------------------------------
# Inhaltsdateien:
$content_files = array();
$content_files['home'] = "0001.txt";
$content_files['impressum'] = "0007.txt";
$content_files['themen'] = "0003.txt";
$content_files['branchen'] = "0009.txt";
$content_files['auskunft'] = "0005.txt";
$content_files['geo'] = "0002.txt";
$content_files['wissen'] = "0004.txt";
$content_files['literatur'] = "0008.txt";
$content_files['lexika'] = "0006.txt";
$content_files['glossare'] = "0010.txt";
$content_files['wikis'] = "0014.txt";
$content_files['ftp'] = "0011.txt";
$content_files['wais'] = "0015.txt";
$content_files['newsgroups'] = "0012.txt";
$content_files['foren'] = "0013.txt";
#=========================================
# Templates einlesen:
$layout_content = file_get_contents($layout_template);
$navigation_content = file_get_contents($navigation_template);
#-----------------------------------------
# Seite aus Templates zusammenfügen:
$page = $layout_content;
$page = preg_replace("/\[\%navigation\%\]/",
$navigation_content, $page);
#-----------------------------------------
# Inhalt seitenabhängig einlesen:
$get_page = "";
if(isset($_GET['page']))
$get_page = $_GET['page'];
else
$get_page = "home";
$content_lines = file($content_files[$get_page]);
$content = implode("", $content_lines);
#-----------------------------------------
# Inhalt in Seite einfügen:
```

```
$page = preg_replace("/\[\%content\%\]/", $content, $page);
#----------------------------------------
# Titel ermitteln und einfügen:
preg_match("/<h1>(.*)<\/h1>/", $content_lines[0], $matches);
$page_title = strip_tags($matches[0]);
$page = preg_replace("/\[\%title\%\]/", $page_title, $page);
#----------------------------------------
# Fertige Seite ausgeben:
echo $page;
?>
```

Info
Das PHP-Script enthält zur besseren Übersicht Kommentarzeilen, die auch zur optischen Trennung von Bereichen im Script genutzt werden. Kommentare in PHP können unter anderem mit dem Gatterzeichen (#) beginnen.

Vom Aufbau her betrachtet, ist in diesem Beispiel ein Script zu sehen, das nicht einmal mehr dem Typus „HTML in PHP“ entspricht, sondern eher wie ein reines Programm ohne Bezug zu HTML aussieht. Dennoch tut es letztlich nichts anderes, als den HTML-Code einer Webseite zu erzeugen.

> Zunächst werden einfach die beiden Namen der Template-Dateien

für Grundgerüst und Navigation in den Variablen `$layout_template` und `$navigation_template` gespeichert. Auch der nächste Abschnitt besteht aus Definitionen. Ein Array namens `$content_files` wird erzeugt. Mittels Zuweisung von Schlüsselnamen und Dateinamen als Werten merkt sich das Script die einzelnen Datendateien. Die Schlüsselnamen wie etwa home in `$content_files['home'] = "0001.txt"` sind übrigens die gleichen, die auch beim GET-Parameter `page` übergeben werden können. Wenn dem Script also beispielsweise im GET-String `page=auskunft` übergeben wird, so kann es mit `$content_files['auskunft']` ohne Umwege die zugehörige Datendatei ermitteln.

Im Script-Abschnitt „Templates einlesen“ werden die beiden Dateien mit dem HTML-Grundgerüst und mit dem HTML-Code für die Navigation eingelesen. Dazu verwenden wir die praktische PHP-Funktion `file_get_contents()`, die das sonst übliche Öffnen, Einlesen und Schließen der Datei in einem Schritt erledigt. Der gesamte eingelesene Dateiinhalt befindet sich anschließend in der Variablen, welcher der Funktionsaufruf zugewiesen ist. Im Beispiel steht also in `$layout_content` anschließend der komplette Inhalt der Datei mit dem HTML-Grundgerüst, und in `$navigation_content` der Inhalt der Datei mit dem HTML-Code für die Navigation.

Das „Zusammenkleben“ der Inhalte beginnt im Script-Abschnitt „Seite aus Templates zusammenfügen“: Dort wird zunächst der Inhalt aus `$layout_content`, also der HTML-Code des Grundgerüsts, in eine neue Variable `$page` kopiert. Dies wird die Arbeitsvariable, die so versorgt wird, dass sie am Ende den kompletten HTML-Code der Seite enthält und ausgegeben werden kann.

Mit der PHP-Funktion `preg_replace()` lassen sich Zeichenketten suchen und ersetzen.

In unserem Beispiel nutzen wir die Funktion zum Ersetzen der Platzhalterzeichenfolge `[%navigation%]`, die ja zum eingelesenen HTML-Grundgerüstcode gehört und folglich nun auch im Inhalt der Variablen `$page` vorkommt, durch den tatsächlichen HTML-Code der Navigation. Dieser ist ja in der Variablen `$navigation_content` gespeichert.

Der erste an `preg_replace()` übergebene Parameter lautet im Beispiel `/\[\%navigation\%\]/`, obwohl die zu suchende Zeichenkette doch eigentlich nur `[%navigation%]` lautet. Das etwas seltsame Konstrukt ist ein so genannter regulärer Ausdruck. Reguläre Ausdrücke benutzen Sonderzeichen, um Muster (`Regularien`) zu definieren, die auf eine oder auch mehrere Zeichenketten zutreffen. Die Funktion `preg_replace()` erwartet beim ersten Parameter das, wonach gesucht werden soll, in Form eines regulären Ausdrucks. Da reguläre Ausdrücke ein wichtiges, auch in ganz anderen Zusammenhängen immer wieder benötigtes Hilfsmittel sind, gehen wir später gesondert darauf ein.

Nach diesem Suche-Ersetze-Vorgang ist in `$page` bereits der HTML-Code des Seitengrundgerüsts inklusive des HTML-Codes mit den Navigationslinks gespeichert. Fehlt nur noch der Seiteninhalt. In diesem Zusammenhang muss das PHP-Script erst einmal ermitteln, welche Seite angezeigt werden soll.

Dazu wird die superglobale Variable `$_GET` ausgelesen,

über welche PHP einem Script eventuell übergebene GET-Parameter zur Verfügung stellt. Da unser vereinbarter Parameter `page` heißt, fragen wir mit `if(isset($_GET['page']))` ab, ob ein Parameter page übergeben wurde. Falls ja, wird in der Variablen `$get_page` der Wert des übergebenen `page`-Parameters gespeichert. Falls nicht, muss das Script ja auch irgendwie reagieren. Es reagiert so, dass `$get_page` in diesem Fall den Wert `home` erhält. Das ist der Wert, den das Script übergeben bekommen würde, wenn die Startseite angezeigt werden soll. Im Klartext: Wenn die Script-Datei ohne `page`-Parameter aufgerufen wird, sorgt es dafür, dass die Startseite des Webangebots angezeigt wird.

Über `$content_files[$get_page]` lässt sich dann die zugehörige Datendatei ermitteln, da `$content_files` ja ein assoziativer Array mit den entsprechenden Schlüsselnamen ist. Wird also etwa als Wert des `page`-Parameters `auskunft` ermittelt und in `$get_page` gespeichert, dann ist `$content_files[$get_page]` so viel wie `$content_files['auskunft']`.

Die Datendatei wird etwas umständlicher eingelesen:

```
$content_lines = file($content_files[$get_page]);
$content = implode("", $content_lines);
```

Info
Die PHP-Funktion `file()` ist wieder eine von den praktischen Funktionen, die uns das Öffnen, Lesen und

Schließen der gewünschten Datei erspart. Die angegebene Datei wird einfach direkt eingelesen, jedoch anders als bei `file_get_contents()` nicht in eine einzige Zeichenkette, sondern zeilenweise, wobei ein Array der einzelnen Dateizeilen zurückgegeben wird. In der zweiten oben gelisteten Anweisung werden die einzelnen Zeilen dann mit der PHP-Funktion `implode()` wieder zu einer einzigen Zeichenkette zusammengefügt.

Dass wir nicht gleich `file_get_contents()` verwendet haben, liegt daran, dass wir nun noch den Array `$content_lines` mit den einzelnen Zeilen haben. Diesen brauchen wir nämlich, da wir ja aus der ersten Datenzeile den Überschriftentext der dort erwarteten `h1`-Überschrift extrahieren und ins `title`-Element einsetzen sollen.

Bevor das geschieht, wird der eingelesene Inhalt der Datendatei

jedoch erst einmal wieder in die am Ende auszugebende Variable `$page` eingefügt. Dazu wird wieder die Funktion `preg_replace()` verwendet. Eingesetzt wird der Inhalt der Datendatei an der Stelle, wo der Platzhalter `[%content%]` notiert ist.

Im vorletzten Abschnitt „Titel ermitteln und einfügen“ extrahiert das Script dann noch wie schon erwähnt aus der ersten Zeile der Datendatei, gespeichert in `$content_lines[0]`, die `h1`-Überschrift, genauer, deren Elementinhalt. Diesen speichert sie in der Variablen `$page_title` zwischen und setzt deren Inhalt schließlich dort in `$page` ein, wo der Platzhalter `[%title%]` notiert ist (Elementinhalt des `title`-Elements).

Die Ausgabe der gesamten Seite ist dann nur noch eine einzige kleine `echo`-Anweisung, die am Ende des Scripts steht.

Fazit: Teilen und Herrschen ist professionell

Das Script linkweb.php ist vom grundsätzlichen Aufbau her ein Beispiel dafür, wie PHP in professionellen Kreisen eingesetzt wird. PHP wird hierbei ausschließlich zum Interpretieren von PHP-Code benutzt und muss nicht dauernd zwischen Interpreter-Modus und HTML-Modus hin- und herschalten. Der PHP-Code ist auch nicht voll von `echo`-Anweisungen, mit deren Hilfe direkt HTML-Code erzeugt wird. Das Script selber fügt nur aus Template-Dateien mit selbst definierten Platzhaltern und Datendateien dynamisch den Inhalt einer anzuzeigenden Webseite zusammen. Das Script erlaubt nicht nur die Trennung zwischen Markup und Layout, sondern auch die Trennung zwischen Navigation und Inhalt.

Gefahren bei Formulardaten Sicherheitsbedenken

Kaum eine Woche vergeht, ohne dass man in einschlägigen Foren von neuen Sicherheitslücken in Webapplikationen liest. Ob SQL-Injection oder Cross-Site-Scripting (XSS), viele dieser Fehler gehen auf die schlampige Verarbeitung von Formulardaten zurück. Dabei wäre es sehr einfach, die gröbsten Fehler in der Behandlung von Formulardaten zu vermeiden:

- Verwenden Sie keine Felder zur Eingabe von Texten (`input type="text"` oder `textarea`). Leider ist das eine etwas zu große Einschränkung für die meisten Anwendungen und deshalb nicht realistisch.
- Geben Sie nie die Benutzereingabe auf einer Webseite aus. Der Grund dafür ist einfach: Für einen Angreifer wird es so möglich, Script-Code auf dem Client auszuführen.

Denkt man an weit verbreitete **PHP-MySQL-Anwendungen** wie zum Beispiel ein Gästebuch, merkt man schnell, dass beide Anforderungen unrealistisch sind. Das bedeutet, das Thema betrifft praktisch jeden Webentwickler.

Datenverlust

Bei Fehlern in der **Formularbehandlung** kann nicht nur der Server angegriffen werden; es gibt noch subtilere Probleme im Zusammenhang mit Zeichensätzen. Datenverlust kann nämlich auch auftreten, wenn die Daten nach der Eingabe falsch kodiert werden und es nicht mehr nachvollziehbar ist, was der Benutzer ursprünglich abgeschickt hat.

> Das im englischen Sprachraum oft vernachlässigte Thema der unterschiedlichen Zeichensätze

kann zu großen Problemen führen, speziell wenn die Anwendung noch mehrsprachig sein soll. Jeder Webentwickler, der schon mehrsprachige Applikationen entwickelt hat, weiß, wovon hier gesprochen wird. Die wirklich richtige Lösung ist eine Zeichenkodierung, die alle Sprachen unterstützt, Unicode. Leider ist die Unterstützung von Unicode noch nicht in allen Programmen ganz fehlerlos implementiert. Wie schon erwähnt wurde, kümmert sich der englischsprachige Teil der Computerwelt nicht so engagiert darum, was dazu führt, dass es bei der Verwendung von Unicode hin und wieder zu unerwarteten Ergebnissen kommt.

Formulardaten richtig verarbeiten

Probleme mit Anführungszeichen und HTML-Sonderzeichen (&, < und >) treten auf, sobald man eingegebene Daten wieder anzeigen möchte, mit ihnen rechnet oder sie etwa in einer Datenbank speichert. Diese Problematik wird durch die automatischen Quotierungsmaßnahmen des PHP-Interpreters (**Magic Quotes**) noch etwas verkompliziert.

Formularauswertung

Die abgeschickten Daten eines Formulars stehen, je nachdem, ob die Aktion auf GET oder POST eingestellt ist, in der globalen Array-Variable `$_GET` beziehungsweise `$_POST` zur Verfügung. Wird eine Variable aufgerufen, die nicht belegt ist (weil der Anwender das Feld leer gelassen hat), kommt es zu einer Warnung. Um solchen Warnungen vorzubeugen, sollte man überprüfen, ob das Array den Schlüssel auch enthält. PHP hat dazu die Funktion `array_key_exists` vorgesehen, der als erster Parameter der zu suchende Schlüssel und als zweiter das gesamte Array übergeben wird. Eine kurze Hilfsfunktion liest gültige Werte aus dem Array aus:

```
function array_item($ar, $key) {
if(array_key_exists($key, $ar)) return ($ar[$key]);
}

// Formulardaten in Variable kopieren
$submitbutton = array_item($_POST, 'submitbutton');
$fname = array_item($_POST, 'fname');
```

Die Variablen `$submit` und `$fname` können jetzt ohne Warnungen überprüft werden.

Info

Im `<form>`-Tag des **HTML-Codes** eines Formulars geben Sie durch `method="post"` oder `method="get"` an, wie die Formulardaten übertragen werden sollen. Welche Methode ist besser? Die Antwort hängt davon ab, welchen Zweck das Formular hat.

`get` bedeutet, dass die **Formulardaten** als Teil der URL an die nächste Seite übertragen werden. Die Daten sind also im Adressfeld des Browsers sichtbar. Üblicherweise wird `get` nur für Suchformulare verwendet. `get` bietet da den Vorteil, dass die Formulardaten bei einer Rückkehr von einem Suchergebnis zur Ergebnisliste weiterhin vorhanden sind und nicht neu übertragen werden müssen. Diese Methode sollten Sie nur verwenden, wenn die Datenmengen klein sind und wenn die Formulardaten keinen Speicher- oder Bestellvorgang auslösen. `get` ist hingegen nicht geeignet, wenn Sie große Datenmengen erwarten, wenn die Daten vertraulich sind (Passwörter etc.) oder wenn Unicode-Zeichen vorkommen können!

`post` bedeutet, dass der Webserver die Formulardaten auf eine für den Benutzer unsichtbare Weise überträgt. In der Praxis werden Sie für die meisten Formulare `post` einsetzen. `post` ist für Daten gedacht, die anschließend nur einmal weiterverarbeitet werden sollen, also z.B. Speicher- oder Bestellvorgänge. Wenn ein Benutzer den ZURÜCK-Button seines Browsers verwendet, um später nochmals zur Seite zurückzukehren, mit der die Formulardaten verarbeitet wurden, müssen die Daten neu übertragen werden. Die meisten Browser zeigen vorher eine Warnung an (damit der Benutzer nicht unabsichtlich eine Bestellung doppelt durchführt).

Formulardaten neuerlich in das Formular einsetzen

Oft ist es nötig, Daten, die in ein Formular eingegeben wurden, noch einmal zu bearbeiten, sie also zu editieren. Dazu werden die Werte in das **HTML-Formular** als `value` eingesetzt. Im praktischen Beispiel sieht das so aus (der PHP-Code innerhalb des HTML-Codes ist fett hervorgehoben):

```
<form method="POST" action="test.php">
<p>Titel:<input type="text" name="title" size="30" maxlength="30" value="<?php
echo $title; ?>" /></p>
<p>Name:<input type="text" name="name" size="30" maxlength="30" value="<?php
echo $name; ?>" /></p>
<p><input type="submit" name="submitbutton" value="OK" /></p>
</form>
```

Eigentlich ist das eine sehr praktische Methode, um den Benutzern die Arbeit abzunehmen, alles noch einmal eintippen zu müssen. Was aber passiert, wenn die Zeichenkette der Variable ein Anführungszeichen enthält? Angenommen, jemand gibt in dem Feld für den Namen seinen Spitznamen in Anführungszeichen ein: Michael „Monty“ Widenius. Der resultierende HTML-Code könnte folgendermaßen aussehen:

```
<p><input type="text" name="lname" size="30" maxlength="30" value="Michael
"Monty" Widenius" /></p>
```

Der Browser interpretiert das Anführungszeichen vor dem Spitznamen als Ende der Zeichenkette. Der restliche Teil wird in der Regel als unerkanntes HTML-Tag einfach weggelassen. Man kann sich schon ausmalen, dass man hier mit versteckten HTML-Tags (vielleicht noch mit etwas JavaScript angereichert) die Webseite ziemlich verändern kann. Einem Angreifer öffnet eine derart falsche Behandlung der Eingabe in jedem Fall die Tür.

Um dieses Problem in den Griff zu bekommen, sieht PHP eine Funktion vor, die jegliche HTML-Sonderzeichen unschädlich macht: `htmlspecialchars`. Die Funktion ersetzt die Sonderzeichen ", &, < und > durch folgende HTML-Zeichenketten:

```
" &quot;
& &amp;
< &lt;
> &gt;
```

Der **Code** zur Darstellung des Formulars sieht jetzt so aus:

```
<form method="POST" action="test.php">
<p><input type="text" name="name" size="30" maxlength="30" value="<?php echo
htmlspecialchars($title);?>" /></p>
<p><input type="text" name="name" size="30" maxlength="30" value="<?php echo
htmlspecialchars($name);?>" /></p>
<p><input type="submit" name="submitbutton" value="OK" /></p>
</form>
```

Magic Quotes

Wie eingangs erwähnt, bietet der **PHP-Interpreter** selbst auch die Möglichkeit, Zeichenketten, die von einer externen Datenquelle stammen, eigenmächtig zu ändern. Diese Funktion wird als `magic quote` bezeichnet und wirkt sich auf die Sonderzeichen `'`, `"`, `\` und das `0-Byte` aus. Sie werden durch einen vorangestellten Backslash ergänzt. (Diese Operation können Sie übrigens auch manuell mit der PHP-Funktion `addslashes` durchführen.)

Magic Quotes betreffen generell Daten, die von außen kommen, also z.B. aus

GET/POST-Formularen, Cookies, SQL-Abfragen etc. In manchen Fällen erleichtert diese automatische Veränderung der Zeichenketten die Weiterverarbeitung, oft stiftet sie aber auch Verwirrung.

PHP kennt drei `magic-quote`-Modi, die in php.ini voreingestellt werden:

- **magic_quotes_gpc**: GPC steht für Get/Post/Cookie, d.h., wenn die Variable gesetzt ist, werden Daten aus diesen drei Quellen verändert. Beachten Sie, dass dies alle Formulardaten betrifft, die in PHP-Scripts verarbeitet werden!
- **magic_quotes_runtime**: Wenn diese Variable gesetzt wird, quotiert PHP in den meisten Fällen Zeichenketten, die aus externen Datenquellen (z.B. MySQL) stammen.
- **magic_quotes_sybase**: Das Zeichen `'` wird durch seine Verdoppelung quotiert (also durch `''` statt durch `\'`). Das gilt aber nur, wenn eine der beiden anderen `magic_quote`-Konfigurationsvariablen gesetzt ist.

Magic-quote-Defaulteinstellung

In der Defaulteinstellung von PHP 4.0 bis 5.0 ist `magic_quotes_gpc` auf On gesetzt, die beiden anderen Modi auf Off. Diese Einstellung werden Sie auch bei den meisten ISPs vorfinden. Sie hat folgende Konsequenzen für die Programmierung:

- Aus Formularen übermittelte Daten können ohne Veränderungen in `INSERT`-Kommandos verwendet werden. Die kritischen Zeichen `'`, `"` und `\` sind schon quotiert.

```
$sql = "INSERT ... VALUES('$formVar')";
```

- Wenn Sie Formulardaten mit Daten aus SQL-Abfragen vergleichen, müssen Sie die Formulardaten vorher mit der PHP-Funktion `stripslashes()` von den `\`-Zeichen befreien.

```
if(stripslashes($formVar) == $sql_result_row[0]) ...
```

Wenn Sie den Vergleich dagegen nicht in PHP, sondern mit **MySQL** durchführen (als Teil einer `WHERE`-Bedingung), sind keine besonderen Maßnahmen erforderlich:

```
$sql = "SELECT ... WHERE authorsName = '$formVar'";
```

- Wenn Sie Formulardaten in einem HTML-Dokument (oder neuerlich in einem Formular, etwa zur Korrektur) anzeigen möchten, müssen Sie zuerst die Quotierungszeichen mit `stripslashes` entfernen und dann alle HTML-spezifischen Sonderzeichen mit `htmlentities` neu (und anders!)

quotieren:

```
echo htmlentities(stripslashes($formVariable));
```

Info
Die PHP-Entwickler schlagen aus Geschwindigkeitsgründen vor, alle drei Modi abzuschalten.

Bevor Sie mit der Entwicklung eigenen Codes beginnen, sollten Sie sich unbedingt darüber informieren, welche Magic-Quote-Einstellungen auf dem Server Ihres Internet Service Providers gelten!

Magic-quote-Modi im PHP-Script ermitteln und verändern

Teilweise können Sie den Zustand der drei Konfigurationsvariablen im Code Ihres **PHP-Scripts** ermitteln oder ändern:

- **magic_quotes_gpc**: `get_magic_quotes_gpc()` ermittelt den aktuellen Zustand. Eine Änderung ist nicht möglich.
- **magic_quotes_runtime**: `get_magic_quotes_runtime()` ermittelt den aktuellen Zustand, `set_magic_quotes_runtime()` verändert ihn.
- **magic_quotes_sybase**: Der Zustand kann weder ermittelt noch verändert werden.

Magic Quotes entfernen

Magic Quotes wurden ohne Zweifel mit der Intention geschaffen, das Leben für Programmierer einfacher zu machen. In der Praxis ist aber leider oft das Gegenteil der Fall: Gerade wenn Sie Code entwickeln, in dem viele Daten aus Formularen ausgelesen und später in dasselbe oder ein anderes Formular übergeben werden, stört die automatische Quotierung von Sonderzeichen oft mehr, als sie nutzt.

Ab PHP-Version 4.1 können Sie die Quotierung durch einen Aufruf der folgenden Funktion abstellen:

```
function no_magic() { // PHP >= 4.1
if (get_magic_quotes_gpc()) {
foreach($_GET as $k=>$v) $_GET["$k"] = stripslashes($v);
foreach($_POST as $k=>$v) $_POST["$k"] = stripslashes($v);
foreach($_COOKIE as $k=>$v) $_COOKIE["$k"] = stripslashes($v);
}
}
```

Unicode in Formularen

Ein **Anmeldeformular** für einen internationalen Kongress sollte dafür gerüstet sein, dass die Namen der Teilnehmer unterschiedlichste Zeichen enthalten. Werden Gäste aus dem Fernen Osten erwartet, reicht der in Westeuropa verbreitete latin1-Zeichensatz (oder eigentlich ISO-8859-1-Zeichensatz) nicht mehr aus.

Die Lösung für diesen Missstand ist ein Zeichensatz, der alle Zeichen enthält, Unicode.

Unicode wurde als Standard entwickelt, um alle Zeichen uns bekannter Schrift mit einer einmaligen Zahl abzubilden. Aus diesem großen Vorhaben stammt die Zeichenkodierung UTF-8, das 8-bit Unicode Transformation Format. UTF-8 wurde im September 1992 von Ken Thompson auf einem Tischset (englisch: placemat) entworfen und tags darauf mit Rob Pike implementiert.

> Das Besondere an UTF-8 ist, dass es eine Zeichenkodierung mit variabler Länge ist

und dass die ersten 128 Zeichen mit dem US-ASCII-Zeichensatz kompatibel sind (diese werden mit nur einem Byte kodiert). Wer mehr als diese 128 Zeichen verwendet, muss ein zweites Byte zur Kodierung heranziehen. Dadurch ergeben sich 1920 Möglichkeiten, die für romanische, griechische, kyrillische, koptische, armenische, hebräische und arabische Buchstaben vorgesehen sind. Um chinesische und japanische Zeichen zu speichern, sind schon drei Bytes nötig, die Platz für 63.488 neue Zeichen geben. Die weiteren Plätze, die mit 4 Bytes kodiert werden können, sind noch nicht vergeben, vielleicht werden ja hier die elvischen Zeichen Platz finden.

Bei Computerprogrammen setzt sich UTF-8 nur langsam durch, da die variable Länge einzelner Zeichen bei der Programmierung für wesentlich größeren Aufwand sorgt. Da UTF-8 als Standard-Zeichensatz für XML gewählt wurde, wird an vielen Stellen unter Hochdruck an einer stabilen UTF-8-Implementierung gearbeitet. Jedes Glied in der Kette muss mit dem Zeichensatz umgehen können, im Fall von PHP und MySQL also der Browser, PHP, der Webserver und die Datenbank.

Um dem Browser verständlich zu machen, welchen Zeichensatz die Seite enthält, wird eine **META-Zeile** im Kopf von HTML mitgeliefert:

```
<html>
<head>
<title>RSS Feed Reader</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
```

Oft reicht das aber nicht aus, um den Browser davon zu überzeugen, dass wirklich UTF-8-Text gesendet wird. Der Webserver sendet, bevor er den eigentlichen Seitentext schickt, einige Informationen über die Seite als Kopfzeilen zum Browser. Wenn der Server so konfiguriert ist, dass der Standardzeichensatz nicht UTF-8, sondern ISO-8859-1 ist (was bei gängigen Linux-Distributionen aus Sicherheitsgründen der Fall ist), wird der Browser versuchen, die UTF-8-Informationen mit dem falschen Zeichensatz darzustellen, und es erscheinen die wohl allen erfahrenen Internetbenutzern bekannten merkwürdigen Zeichen statt der korrekten Umlaute. Zwar kann der Browser manuell umgestellt werden, doch will man das seinen Besuchern sicher nicht zumuten.

PHP bietet eine Möglichkeit, diese Header-Information des Webservers zu beeinflussen:

die `header`-Funktion. Sie muss vor jeglichen anderen Ausgaben gemacht werden und sieht für die UTF-8-Umstellung zum Beispiel so aus:

```
header("Content-Type: text/html; charset=utf-8");
```

Damit erkennt der Browser nun automatisch, dass es sich um den **UTF-8-Zeichensatz** handelt. Interessant wird es bei Formularen, die der Benutzer abschickt. Je nachdem, welchen Zeichensatz der Browser eingestellt hat, werden die Daten in den Formularfeldern interpretiert. Hat der Browser also UTF-8 eingestellt und schickt er ein Formular ab, enthalten die Variablen alle in UTF-8 kodierte Strings. Das ist zu berücksichtigen, wenn diese Variablen in eine Tabelle oder in eine Datei geschrieben werden. Das folgende Beispiel wird darauf eingehen.

Beispiel: Anmeldung zum Kongress

Das kurze Beispiel gibt ein Anmeldeformular aus, in dem Benutzer ihre Daten eingeben und abschicken. Anschließend werden Sie nach der Richtigkeit der Daten gefragt und haben die Möglichkeit, sie bei Bedarf zu verändern. Experimentieren Sie ruhig mit den Eingabefeldern, und versuchen Sie, Sonderzeichen wie äöü, \ oder ganze HTML-Tags (<b>) einzugeben.

```
<?php // Datei: forms/register.php
header("Content-Type: text/html; charset=utf-8");
function array_item($ar, $key) {
if(array_key_exists($key, $ar))
if (get_magic_quotes_gpc()) {
return stripslashes($ar[$key]);
}
else {
return $ar[$key];
}
}
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
// HTML mit Titel und Überschrift
```

Das **PHP-Script** beginnt mit der Anweisung, den Browser auf den UTF-8-Zeichensatz umzustellen, und definiert die schon besprochene Funktion zur Überprüfung der gesendeten Array-Variablen. Eine kleine Änderung gibt es dabei in diesem Beispiel: Sind Magic Quotes eingeschaltet (`get_magic_quotes_gpc()`), werden die dabei erzeugten Backslashes gleich entfernt. Würde man

diesen Schritt nicht einbauen, verdoppeln sich die Backslashes bei jedem neuerlichen Abschicken der Variablen.

```
<form method="POST" action="register.php">
<?php
$submit = array_item($_POST, 'submitbutton');
$fname = array_item($_POST, 'fname');
$lname = array_item($_POST, 'lname');
$email = array_item($_POST, 'email');
$cdate = array_item($_POST, 'cdate');
if ($submit == 'Register') {
if ($fname == '' || $lname == '' || $email == '') {
print "<b>You forgot one of the required fields!</b>";
} else {
print "<p>Your date is complete:</p>\n";
print "<pre>\n";
print "Firstname: ". htmlspecialchars($fname)."\n";
print "Lastname:  ". htmlspecialchars($lname)."\n";
print "Email:     ". htmlspecialchars($email)."\n";
print "Congressday: ".$cdate."\n\n";
print "<input type=\"submit\" name=\"submitbutton\"
value=\"Confirm\"> ";
print "<input type=\"submit\" name=\"submitbutton\"
value=\"Correct\"></pre>\n";
print "<input type=\"hidden\" name=\"fname\"
value=\"".htmlspecialchars($fname)."\">";
// ... Ausgabe der restlichen Felder
print "</form></body></html>";
exit();
}
```

Im Hauptteil des Programms werden die abgeschickten Variablen ausgelesen, sofern diese vorhanden sind. Über den Status von `$submit` wird ermittelt, ob das Formular zum ersten Mal abgeschickt wird (`$submit == 'Register'`) oder schon zur Korrektur vorlag (`$submit == 'Confirm'`, siehe unten). Nur wenn alle drei Felder ausgefüllt sind, werden die Daten mit einer nichtproportionalen Schrift (`<pre>`) angezeigt. Wichtig ist dabei, dass die Variablen bei der Ausgabe immer in `htmlspecialchars` geklammert sind, wodurch sichergestellt wird, dass kein HTML-Code eingeschleust werden kann. Da das Datum (`$cdate`) kein Texteingabefeld ist, können hier keine unerwarteten Zeichen auftreten.

Das Formular zur Bestätigung übergibt die **Variablen** als `input type hidden`, wodurch sicher gestellt wird, dass der Benutzer die geprüften Daten nicht mehr verändert hat. Sollen Veränderungen vorgenommen werden, muss dies über einen neuerlichen Aufruf des Scripts geschehen. Die `exit()`-Anweisung bricht den Programmablauf ab, weshalb hier noch der korrekte HTML-Abschluss eingefügt wird.

```
}
else if ($submit == 'Confirm') {
$mailtext = "$fname\t$lname\t$email\t$cdate\n";
mail("register@server.com", "New registration", $mailtext);
print "<p>Thank You!</p>";
print "<p>Your registration has been sent to us.</p>";
print "<p><a href=\"/\">Home</a>\n</body></html>";
exit();
}
?>
```

Hat der Benutzer die Schaltfläche Confirm angeklickt, werden seine Daten per E-Mail an die Adresse register@server.com geschickt.

```
<table border="2">
<tr>
<td>Firstname</td>
<input type="text" name="fname" value="<?php echo htmlspecialchars($fname); ?>"
size="30">
// restlichen Text für die Eingabefelder hier
<td>Congress day:</td>
<select name="cdate">
<option value="20041105" <?php if ($cdate == '20041105') print "selected"; ?>
>2004-11-05</option>
<option value="20041106" <?php if ($cdate == '20041106') print "selected"; ?>
>2004-11-06</option>
<option value="20041107" <?php if ($cdate == '20041107') print "selected"; ?>
>2004-11-07</option>
</select>
</tr>
</table>
<i>(all Fields are mandatory)</i>
<p><input type="submit" name="submitbutton" value="Register"></p>
</form>
</body>
</html>
```

Der restliche Code ist im Wesentlichen die HTML-Formatierung der Eingabefelder. Zu beachten ist, dass auch hier die Variablen, die als `value` in die Textfelder eingesetzt werden, mit `htmlspecialchars` ausgegeben werden, um unerwünschten Ergebnissen vorzubeugen.

Während des gesamten Programmablaufs wird ausschließlich der UTF-8-Zeichensatz verwendet, so kommt auch die gesendete E-Mail als UTF-8-kodierte Nachricht an. Sollte Ihr E-Mail-Programm das nicht von selbst

erkennen, so gibt es meist die Möglichkeit, den Zeichensatz von Hand einzustellen. Damit steht Ihren Geschäftspartnern also (fast) nichts mehr im Wege, ihren Namen auf Japanisch einzutragen.

Mit der in diesem Kapitel beschriebenen Aufgabe werden einige neue Funktion eingeführt und deren Gebrauch demonstriert, außerdem benötigen Sie zum Mitspielen des Beispiels erneut **Arrays** und eine `if`-Kontrollstruktur. Geplant ist eine Seite, die unterschiedliche Texte ausgibt, und zwar eine wechselnde Begrüßung und wechselnde „Fußballerweisheiten“.

Während die Begrüßung, wie Sie sich denken können, abhängig ist von der Tageszeit, erscheint das sich ändernde Fußballerzitat aufgrund der Generierung einer Zufallszahl. Die Tageszeit wird der Systemzeit des Servers entnommen. Die Seite sieht in etwa so aus wie in Bild:

> PHP bietet diverse Funktionen für Datumsangaben, die unterschiedliche Aufgaben übernehmen.

So lässt sich die aktuelle Zeit (Datum und Uhrzeit) auslesen, oder beispielsweise eine bestimmte Datumsinformation festlegen.

Mit diesem **Script** möchten wir auch demonstrieren, dass Scripte so, aber auch anders geschrieben werden können. Im vorliegenden Beispiel sind wir so vorgegangen, dass zunächst viele Variablen definiert wurden, die alle notwendigen Werte enthalten. Diese Methode macht dann den Teil des Scripts, der für die Ausgabe sorgt, sehr einfach und übersichtlich, da Sie im Prinzip nur noch schreiben müssen: `echo "$Variable so und so"`.

Es hätte ohne Zweifel auch alternative Wege gegeben, z.B. ließe sich die Anweisung, welcher Begrüßungstext erscheinen soll, auch direkt in den Block der `if`-Bedingung einbauen. Die gewählte Methode ist jedoch die professionellere, weil ein Script auf diese Weise sehr flexibel bleibt.

Vorüberlegungen

Für eine Seite, die auf Basis der aktuellen Tageszeit den einen oder anderen Text ausgibt, brauchen Sie die Funktion `date()`.

Mit dieser Funktion wird das aktuelle Datum/die aktuelle Zeit als Zeichenkette zurückgegeben, wobei es vom Parameter abhängt, was wie ausgegeben wird. (Eine Übersicht über die Formate finden Sie in einer Tabelle in Vordefinierte Funktionen).

Für die Aufgabe, eine Tageszeit bestimmen zu lassen, eignet sich der Parameter `date(H)`. Damit berechnet **PHP** die Stunde aus einem 24-Stunden-Format, also zweistellig in der Form: 06 oder 14. Praktisch ist außerdem die Funktion `date(A)`; Die Rückgabe dieser Funktion ist je nach Tageszeit entweder AM oder PM.

Info

```
string date(string format [, int timestamp])
```

Die Funktion formatiert mittels der angegebenen Parameter eine angegebene Zeit bzw. ein angegebenes Datum. Ohne `timestamp` wird die aktuelle Zeit zurückgegeben.

Sie können am Script – abgebildet weiter unten – erkennen, dass die `date`-Funktion als Prüfbedingung des `if`-Befehls eingesetzt wird. Darüber hinaus ergibt sich die Möglichkeit, die Zeitausgabe zu differenzieren, indem mit den Operatoren größer als bzw. kleiner als gearbeitet wird.

Für die Generierung einer Zufallsauswahl benutzen Sie die Funktion `mt_rand()`.

Eine nähere Erklärung zu dieser Funktion finden Sie in dem Abschnitt Erläuterung des Scripts. Das Script, das die beiden beschriebenen Aufgaben erfüllt, sieht in der gewählten Variante so aus):

```
<?php
$begr[0]="Guten Morgen!";
$begr[1]="Guten Tag!";
$begr[2]="Guten Abend!";
$meld[]="Ein Spiel dauert 90 Minuten und am Ende gewinnen die Deutschen";
$meld[]="Schwach wie eine Flasche leer";
$meld[]="Mailand oder Madrid Hauptsache Italien (Andy
Möller)";
$meld[]="Man hetzt die Leute auf mit Tatsachen, die nicht der Wahrheit
entsprechen (Toni Polster)";
$meld[]="Wir wollten in Bremen kein Gegentor kassieren, das hat bis zum Gegentor
auch ganz gut geklappt (Thomas Hässler)";
$meld[]="Die Breite an der Spitze ist dichter geworden (Berti Vogts)";
$meld[]="Das nächste Spiel ist immer das nächste (Matthias Sammer)";
$meld[]="Nach dem Spiel ist immer vor dem Spiel";
$meld[]="Das wird doch alles von den Medien hochsterilisiert! (Bruno Labbadia)";
$meld[]="Der Jürgen Klinsmann und ich sind schon ein tolles Trio, ....aeh
Quartett. (Fritz Walter) ";
$meld[]="Ich glaube, daß der Tabellenerste jederzeit den Spitzenreiter schlagen
kann. (Berti Vogts) ";
$meld[]="Zwei Chancen, ein Tor - das nenne ich hundertprozentige
Chancenauswertung. (Roland Wohlfahrt) ";
$meld[]="Wir werden nur noch Einzelgespräche führen, damit sich keiner verletzt.
(Frank Pagelsdorf) ";
$meld[]="Wenn der Ball am Torwart vorbei geht, ist es meist ein Tor. (Mario
Basler)";
$meld[]="Was der Rudi Bommer heute mit seinen 800 Jahren geleistet hat, war
```

```
schon phänomenal. (Dragoslav Stepanovic)";
$meld[]="Zuerst hatten wir kein Glück und dann kam auch noch Pech dazu. (Uwe
Wegmann)";
$meld[]="Wenn die deutsche Mannschaft gut spielt, wird sie Weltmeister - wenn
nicht, kommt sie ins Endspiel. (WM-Organisator Michel Platini)"; $meld[]="Lieber
Gott, wenn ich Wimbledon gewinne und Kroatien Weltmeister wird, wird das ganze
Land bis zum Jahresende betrunken sein - und ich mit ihm. (Wimbledon-Finalist
Goran Ivanisevic) ";
mt_srand((double)microtime()*1000000);
$zufall=mt_rand('0',count($meld)-1);
$meldung=$meld[$zufall];
if(date("A")=="AM")
{$be=$begr[0];}
elseif(date("H")>=12 AND date("H")<18)
{$be=$begr[1];}
else
{$be=$begr[2];}
?>
<html>
<head>
<title>hallo</title>
</head>
<body bgcolor="yellow">
<p><h2><center>Willkommen auf unserer Seite</center></h3><p>
<?php echo "<b><font size=+1>$be</font></b>"; ?>
<p><?php echo "<b><font size=+1>$meldung</font></b>";
?>
</body>
</html>
```

Erläuterung des Scripts

Das Script beginnt wie schon die vorherigen Beispiele mit der **PHP-Auswertung**. Da Sie Arrays bereits kennen gelernt haben, können Sie ohne Mühe erkennen, dass zunächst ein Array definiert wird, das die unterschiedlichen Begrüßungstexte enthält. Das Array beginnt mit:

```
$begr[0]="Guten Morgen";
```

Das zweite Array legt die unterschiedlichen Texte fest, die bei Aufruf der Seite nach dem Zufallsprinzip erscheinen. Der Array-Bezeichner oder Schlüssel ist ein Paar eckiger Klammern ohne Indexwert. In dem Fall sorgt PHP für die Indizierung. Denken Sie aber daran, dass die Indizierung bei Null und nicht bei Eins beginnt. Die erste Zeile des Arrays heißt in unserem Beispiel:

```
$meld[]="Ein Spiel dauert 90 Minuten und am Ende gewinnen die Deutschen";
```

Bei den nächsten Zeilen geht es um die Anweisung, nach dem Zufallsprinzip die im Array $meld[] gespeicherten Texte auszugeben. Wir verwenden dafür zunächst die Funktion `mt_srand()`.

> Das „rand“ steht für random, was übersetzt nichts anderes heißt als „Zufallsprinzip“.

Mit dieser Funktion wird der Startwert für den Zufallszahlengenerator gesetzt. Die Argumente in der Klammer legen die interne Berechnung fest. Sie können die Funktion einfach so übernehmen wie abgebildet und sie auch genauso bei anderen Projekten, bei denen es um Zufallszahlen geht, verwenden.

```
mt_srand((double)microtime()*1000000);
```

Dann wird die Variable $zufall definiert und ihr ein Wert zugewiesen. Die Funktion `mt_rand` gibt eine Zufallszahl zurück.

```
$zufall=mt_rand('0',count($meld)-1);
```

Info

```
int mt_rand(int min [, int max])
```

Die Funktion gibt eine Zufallszahl zwischen `min` und `max` zurück.

```
void mt_srand(int seed)
```

Die Funktion generiert einen internen Startwert für den Zufallsgenerator. Es wird kein Ergebniswert zurückgegeben.

Mithilfe der Argumente erreichen wir, dass eine Zufallszahl erzeugt wird, die zwischen 0 und der Anzahl der Elemente des Arrays $meld minus 1 liegt. Die Funktion `count` zählt die Elemente des Arrays. Diese Begrenzung ist sinnvoll, da über die Zufallszahl das anzuzeigende Element des Arrays $meld ausgewählt wird. Das ausgewählte Element wird der neuen Variablen $meldung zugewiesen.

```
$meldung=$meld[$zufall];
```

Info

```
int count(mixed arrayvariable)
```

Die Funktion zählt die Menge der Elemente innerhalb eines Arrays und gibt die Anzahl zurück.

Die Grenzen, in der die Zufallszahl liegen darf, erklären sich folgendermaßen: Der Bereich muss bei 0 beginnen, da der Indexwert 0 das erste Element des Arrays $meld anspricht. Dies heißt: wenn $zufall gleich 0 ist, somit der kleinste mögliche Wert, wird mit $meld[$zufall] (entspricht $meld[0]) das erste Element des Arrays $meld angesprochen. Der Maximalwert der Zufallszahl darf nicht größer sein als es Elemente im Array $meld gibt. Genauer betrachtet, ist es die Anzahl der Element minus 1, da die Elemente bei 0 beginnend indiziert werden (Indexwert=0).

Damit haben Sie den ersten Teil programmiert. In $meldung steht jetzt ein über die Zufallszahl mit jedem Aufruf variierendes Element aus dem Array $meld.

Im zweiten Teil des Scripts geht es um die Begrüßung des Users, die je nach Tageszeit wechseln soll.

Die Texte wurden weiter oben bereits festgelegt. Sie müssen PHP jetzt mitteilen, was wann passieren soll. Dies können Sie mit einer recht einfachen `if-else`-Bedingung lösen.

Beginnend mit

```
if(date("A")=="AM")
{$be=$begr[0];}
```

sagen Sie zunächst, dass, wenn die Funktion `date(A)` aufgrund der Systemzeit des Servers AM als wahr zurückgibt, eine **Variable** definiert wird, der als Wert das nullte Element des Arrays $begr zugewiesen wird. Beachten Sie also bitte: die Anweisung in der `if`-Verzweigung sagt hier noch nicht: gebe das und das aus, sondern weist der Variablen $be den Wert eines Arrayelementes zu.

Ähnlich ist die `elseif`-Verzweigung aufgebaut, nur, dass Sie hier als Parameter der Funktion `date` das Format („H“) angeben und danach die zu prüfende Tageszeit formulieren, indem Sie Operatoren einsetzen. Für die Verknüpfung brauchen Sie `AND`:

```
elseif(date("H")>=12 AND date("H")<18)
{$be=$begr[1];}
```

Nach der `else`-Anweisung, die der Variablen $be mit

```
else {$be=$begr[2];}
```

den Wert des zweiten Elementes der Array-Liste zuweist, schließen Sie zunächst den PHP-Teil. In $be steht jetzt also je nach Tageszeit ein anderer Text.

Nach dem Text „Willkommen auf unserer Seite“, durch den **HTML-Tag** <h3> formatiert als Überschrift der Ebene drei, öffnen Sie erneut PHP, um den Wert der Variablen mit `echo` auszugeben. Da Sie für die Begrüßung und für die unterschiedlichen Texte Variablen gesetzt haben, ist dieser Teil schnell geschrieben. Wir bauen lediglich noch eine Formatierung für beide Textausgaben ein (fett und einen etwas größeren Schriftgrad):

```
<?php echo "<b><font size=+1>$be</font></b>";
?>
<p>
<?php echo "<b><font size=+1>$meldung</font></b>";
?>
```

Speichern Sie das Dokument und testen Sie es in Ihrem Browser. Je nach Tageszeit mit einem unterschiedlichen Begrüßungstext müssten Sie nun eine Seite erhalten, die in etwa so aussieht wie in Bild 2 oder Bild 3. Lassen Sie die Seite mehrmals aktualisieren, um zu überprüfen, ob der Text der „Fußballersprüche“ sich jeweils ändert. Wenn das geschieht, hat die Programmierung der Seite geklappt und wir gratulieren.

Ausblick – Texte aus einer Datei übernehmen

Möchten Sie die unterschiedlichen Texte nicht wie im obigen Beispiel per Zufallsprinzip aus einem Array auswählen, sondern ‚per Hand‘ jeweils in eine Textdatei schreiben (also beispielsweise jeweils den ‚Spruch des Tages‘), die Sie bei Bedarf per FTP-Programm auf den Server laden oder in einem weiteren Schritt mit einem Uploadformular auf den Server zaubern, können Sie den Text der Datei ganz einfach in Ihre Webseite integrieren. Dies zeigt das nachfolgende Script:

```
<html>
<head>
<title>Textdatei auslesen</title>
</head>
<body>
<h1>Der wechselnde Text:</h1>
<h3>
<?php
readfile('textdatei.txt');
?>
</h3>
</body>
</html>
```

Mit der Funktion `readfile()` wird der Inhalt der angegebenen Datei automatisch an den Browser ausgegeben. Sie benötigen keinen `echo`– oder `print`-Befehl. Im Beispiel muss die Datei im gleichen Verzeichnis wie die Webseite liegen, Sie können aber auch relative Pfadangaben verwenden oder auf Dateien anderer Webserver mithilfe einer vollständigen URL zugreifen (http://www.domain.de/textdatei.txt).

Der Euro hin oder her – schnell mal einen bestimmten Betrag in eine andere Währung umzurechnen, steht auch in Zeiten einer gemeinsamen Währung nach wie vor oft auf der Tagesordnung. Außerdem – Hand aufs Herz – werden wir zunächst auch die neuen Euro-Beträge klammheimlich in die alt-vertraute Währung übersetzen wollen, damit sich ein besseres Gefühl für die Preise einstellt!

Dieser Artikel ist in zwei Teile unterteilt. Zunächst soll ein einfacher Umrechner kreiert werden, bei dem unterschiedliche Ausgangswährungen festgelegt werden können, die Zielwährung aber immer Euro ist. Im zweiten Teil wird demonstriert, wie Sie einen Umrechner programmieren, mit dem es möglich ist, Beträge in unterschiedlichen Ausgangswährungen in unterschiedliche Zielwährungen umzurechnen. Dabei haben wir aus Gründen der Überschaubarkeit beispielhaft nur ein paar Währungen ausgewählt – für einen ernsthaft anzuwendenden Umrechner müsste die Liste der Währungen natürlich deutlich verlängert werden!

> Auf der Seite mit einem einfachen Euro-Umrechner gibt es ein Auswahlfeld mit diversen Währungsoptionen,

ein Feld, um den Betrag einzugeben und die Schaltfläche zum Abschicken der Berechnung. Die Seite sieht in etwa so aus wie in Bild 1. Bedenken Sie bitte, dass wir wieder wenig Wert auf die optische Gestaltung gelegt haben. Es bleibt Ihnen überlassen, mit den Gestaltungsmitteln von **HTML** am Aussehen des Formulars zu feilen.

Bei der zweiten Variante gibt es naturgemäß zwei Auswahlfelder, eins für die Ausgangswährung und eins für die Zielwährung. Diese Seite kann etwa so aussehen wie Bild 2.

Beide Aufgaben werden mit jeweils nur einem Dokument gelöst, das als PHP-Datei abgespeichert wird. Die PHP-Teile des Scripts werden jeweils durch den php-Tag <?php geöffnet und mit ?> geschlossen; diesen Wechsel können/könnten Sie prinzipiell so häufig wiederholen wie es erforderlich ist. Alle Elemente des Scripts, die nicht zwischen den **PHP-Zeichen** stehen, sind HTML-Codes.

Vorüberlegungen zur ersten Variante: Euro-Umrechner

Geplant ist als Erstes ein Umrechner, der Beträge in verschiedenen Währungen in Euro konvertiert. Dazu brauchen Sie ein Auswahlfeld zur Bestimmung der Ausgangswährung und ein Feld, in das der Betrag eingegeben wird.

Für die Verarbeitung durch PHP legen Sie teils Variablen fest und weisen ihnen Werte zu, teils werden die Inhalte/Werte von den Formularfeldern übergeben. Die Formularfelder müssen also eindeutige Namen

erhalten, die dann als Variablennamen verwendet werden. Dann kommt ein neues Element ins Spiel: statt der `if`-Anweisung, die Sie bereits kennen gelernt und angewendet haben, ist es in diesem Fall günstiger, die `switch`-Bedingung zu benutzen, da Sie mehrere aufeinander folgende Bedingungen (unterschiedliche Ausgangswährungen) gegen eine Variable prüfen möchten. Sie könnten bei der Sache auch den `if`-Befehl verwenden, aber – wie gesagt – die Variante mit dem `switch`-Befehl ist in diesem Fall eleganter.

Gebrauch der Switch-Bedingung

Wie haben die `switch`-Bedingung im Kontrollstrukturen bereits erklärt. Hier nochmals zur Erinnerung: Die Schreibweise der `switch`-Bedingung ist folgendermaßen:

```
switch ($variablenname)
{
case "wert1": Anweisung1; break;
case "wert2": Anweisung2; break;
default: Anweisung2;break;
}
```

Die `switch`-Bedingung funktioniert so: wenn **PHP** den Fall (`case`) entdeckt, der mit dem Wert der angegebenen Variable bzw. dem `switch`-Ausdruck korrespondiert, wird die Anweisung ausgeführt. PHP setzt dann die Abarbeitung des restlichen Codes innerhalb des `switch`-Blocks fort und zwar entweder solange, bis das Ende der Anweisungen in den geschweiften Klammern erreicht ist oder bis die Ausführung unterbrochen wird. Dies geschieht durch die Eingabe `break`. Deswegen ist es wichtig, dass jeder `case` mit `break` geschlossen wird. Nur in ganz seltenen Fällen ist es beabsichtigt, die Ausführung des Scripts nicht zu unterbrechen und durch mehrere Cases laufen zu lassen.

Die `switch`-Bedingung benutzt also den Wert der Variablen als seine Bedingung und prüft, ob die Variable dem Wert1 (erster case) entspricht; ist dies der Fall, wird die Anweisung ausgeführt. Trifft keine der Bedingungen zu, tritt die default-Anweisung ein.

Sie können sich denken, dass diese `switch`-Bedingung passend für den geplanten Euro-Konverter ist. Für das Auswahlfeld wird eine Variable festgelegt, die mit dem Namen des Feldes korrespondiert, im `case`-Teil taucht der Wert auf und die Anweisung gibt an, was jeweils geschehen soll. Wir erwähnten bereits, dass auch eine `if`-Bedingung stattdessen eingesetzt werden könnte, aber die `switch`-Bedingung sorgt unserer Meinung nach in diesem Fall für mehr Klarheit und überdies geht es ja auch darum, eine Programmier-Methode durchzuspielen, die Sie bisher noch nicht probiert haben.

Das Script für den einfachen Euro-Umrechner

Werfen Sie einen Blick auf den Code unten. In diesem Script stecken das Formular und die PHP-Verarbeitung nach der eben beschriebenen Methode. Beachten Sie, dass wir den PHP-Teil an den Anfang gesetzt haben. Dadurch wird die Berechnung als Erstes ausgeführt und das entsprechende Ergebnis vor dem Formular angezeigt.

```
<html>
```

```
<head>
<title>Euro-Rechner</title>
</head>
<body>
<?php
$ergebnisDM=$betrag*0.51;
$ergebnisPes=$betrag*0.006010;
$ergebnisFranc=$betrag*0.15245;
switch($rate)
{
case "DM":
echo "dies sind $ergebnisDM Euro"; break;
case "Peseten":
echo "das sind $ergebnisPes Euro"; break;
case "Franc":
echo "dies sind $ergebnisFranc Euro"; break;
}
?>
<h2>Ausgangswährung</h2>
<p>
<form action="euroumrechner.php" method=post> <select size=1 name=rate>
<Option value="DM">
DM
<Option value="Franc">
Franc
<Option value="Peseten">
Peseten
<Option value="Euro">
Euro
</select>
<p>
<b>Betrag</b>
<br>
<input type=text name=betrag size=20>
<p>
<input type=submit Name="submit" Value="berechnen">
</body>
</html>
```

Erläuterung zum Script in Schritten

Das Script beginnt nach dem üblichen HTML-Header mit dem PHP-Teil.

- 1. Zunächst werden mit $ergebnisDM etc. die Namen der Variablen festgelegt.
- 2. Diesen Variablen weisen Sie Werte zu. Die Variable $betrag korrespondiert mit dem Namen des Formularfeldes für die Eingabe des Betrages, der Variablenname muss also auf jeden Fall identisch

sein mit dem weiter unten eingegebenen Namen im Tag `<input type=text name=betrag>`. Die Berechnung, die der Variablen als Wert zugewiesen wird, enthält die Multiplikation des Betrages mit dem jeweiligen Euro-Kurs. Der arithmetische Operator für eine Multiplikation ist – wie Sie wissen – das Sternchen *.

- 3. Im Switch-Befehl steht die zu testende Variable. Der Variablenname korrespondiert mit dem Namen des im HTML-Teil definierten Auswahlfeldes, im Beispiel ist dies `rate`. Beim Festlegen des Auswahlfeldes muss also `rate` als Name im **HTML-Tag** `<Select>` auftauchen.
- 4. In den `case`-Abschnitten stehen die Testwerte (jeweils ein Wert des Auswahlfeldes) und danach, also in der geschwungenen Klammer, formulieren Sie, was geschehen soll, wenn das Prüfergebnis `true` ergibt. Dann wird mit der Ausführung des Codes begonnen. Mit `echo` geben Sie an, dass das Ergebnis der Berechnung im Browser ausgegeben wird.
- 5. Nach dem PHP-Teil wird der HTML-Teil geöffnet. Zunächst geben Sie mit `<form>` an, dass Sie ein Formular anlegen. Der ausführende Befehl spricht das aktuelle PHP-Script an. Als Methode zur Datenübertragung wurde wieder `Post` gewählt.
- 6. Das Auswahlfeld legen Sie mit `<select>` fest; hier müssen Sie darauf achten, dass der Wert des Attributs `name` identisch ist mit der Variablen in der Switch-Bedingung (im Beispiel `rate`).
- 7. Die Werte der Optionen entsprechen den Elementen im case. Für das Feld zur Eingabe des Betrages benötigen Sie den Tag `<input type=text>` zusammen mit dem Attribut `name`. Der Name des Feldes wird im PHP-Teil als Variable verwendet. Da wir zu Beginn des PHP-Teils bereits $betrag festgelegt haben, muss es hier also auch heißen: `<input type=text name=betrag>`.
- 8. Zu guter Letzt benötigen Sie wieder eine Schaltfläche zum Abschicken der Eingaben.
- 9. Speichern Sie das Script als PHP-Datei und rufen Sie es in Ihrem Browser auf. Testen Sie es, indem Sie eine Währung auswählen und einen Betrag eingeben. Nach einem Klick auf BERECHNEN müssten Sie die Antwort angezeigt bekommen. In dem Bild sehen Sie das Ergebnis.

Ergänzung: eine Fehlermeldung ausgeben

So weit so gut! Es fehlen aber noch ein paar Kleinigkeiten. Zunächst: Damit der Besucher daran erinnert wird, einen Betrag einzugeben, falls er auf die Schaltfläche BERECHNEN klickt, ohne einen Betrag eingetragen zu haben, müssen Sie das Script um eine `if-else`-Bedingung ergänzen. Die `if`-Anweisung lautet:

```
if (!$betrag) {echo 'bitte geben Sie einen Betrag ein <br>';}
else {;
}
```

Diese Anweisung setzen Sie über die `switch`-Bedingung. Mit !$betrag in der Klammer wird die Variable $betrag negiert, d.h. die Prüfung der Bedingung gibt `true` zurück, wenn das Feld leer ist. Dann folgt als Ereignis die Meldung. Im `else`-Block steht die ganze `switch`-Bedingung, denn wenn das Feld nicht leer ist, soll keine Meldung erfolgen, sondern die Berechnung und die Ausgabe. Nach dem Ende der `switch`-Bedingung wird der `else`-Block mit der geschweiften Klammer geschlossen.

```
<?php
```

```
$ergebisDM=$betrag*0.51;
$ergebnisPes=$betrag*0.006010;
$ergebnisFranc=$betrag*0.15245;
if (!$betrag) {echo '<b>Bitte geben Sie einen Betrag ein</b>
<br>';}
else {
switch($rate)
{
case "DM":
echo "<b>Dies sind $ergebnisDM Euro</b>";
break;
case "Peseten":
echo "<b>Das sind $ergebnisPes Euro</b>";
break;
case "Franc":
echo "<b>Dies sind $ergebnisFranc Euro</b>";
}
}
?>
```

Speichern Sie das Script ab und rufen Sie es in Ihrem Browser auf. Schicken Sie es dann einmal ab, ohne einen Betrag eingegeben zu haben. Sie erhalten, wie Bild zeigt, eine entsprechende Meldung.

Es gibt einen weiteren Schönheitsfehler. Die Zahlen, die als Ergebnis erscheinen, werden nicht abgerundet und formatiert. Außerdem wäre es sinnvoll, wenn die gewählte Währung und der eingegebene Betrag auch nach der Berechnung im Formular erhalten blieben. Diesen Problemen rücken wir in dem nächsten Script zu Leibe.

Ein Währungsumrechner

Das zweite Beispiel ist komplexer. Wir programmieren einen Umrechner, der Beträge in verschiedenen Ausgangswährungen in unterschiedliche Zielwährungen konvertiert, und nicht nur in den Euro.

Vorüberlegungen

Offensichtlich brauchen Sie diesmal zwei Auswahlfelder, das eine für die Bestimmung der Ausgangswährung, das andere für die Bestimmung der Zielwährung. Als drittes muss die Seite ein Textfeld für die Eingabe des Betrages enthalten und natürlich wieder eine Schaltfläche zum Abschicken der Einaben.

> Für die Ausführung durch PHP kommen nun mehrere neue Elemente ins Spiel.

Die Auswahlfelder enthalten relativ lange Listen mit den verschiedenen Währungen. Diesen Währungen müssen Werte zugewiesen werden. Dies würde eine Fülle von „normalen“ Variablen mit jeweils einem Wert erforderlich machen. Um hier geschickter vorzugehen, werden in diesem Fall Arrays verwendet, also die Sonderform von Variablen, die viele Werte speichern können (dazu im nächsten Abschnitt mehr).

Da nicht Hunderte von Wechselkursen zwischen den unterschiedlichen Währungen definiert werden können, brauchen wir einen Kurs, auf den alle anderen Währungen bezogen werden. Dafür bietet sich der Euro an. Die Umrechnung eines Betrages von einer Währung in eine andere erfolgt dann über den Zwischenschritt, zunächst in den Euro zu konvertieren und dann gegebenenfalls in die gewünschte andere Währung. Auf diese Weise wird die Umrechnung praktikabel.

Die Arbeit mit Arrays

Arrays sind eine Sammlung beliebig vieler unterschiedlicher Werte, die in einer einzigen Variable zusammengefasst werden. Jeder Wert wird durch eine Zahl oder Zeichenkette indiziert. Auf diese Weise lässt sich die Liste als Ganzes ansprechen, ebenso können einzelne Array-Elemente ausgewählt und bearbeitet werden.
Info

Beachten Sie bitte: Was ArraysArrays sind, haben wir bereits erklärt. An dieser Stelle finden Sie nur eine kurze Zusammenfassung.

Noch einmal zur Syntax: Genauso wie Variablen haben Arrays einen Namen, dem ein Dollarzeichen vorangestellt wird, aber im Unterschied zu einer normalen Variablen besteht ein Array aus beliebig vielen indizierten Elementen. Daher hat jedes Element eines Arrays einen Schlüssel und einen Wert mit der Schreibweise:

```
$arrayname[Schlüssel]=Wert;
```

Über den Schlüssel, der aus einer Zahl oder einem String bestehen kann, erfolgt der Zugriff auf den Wert. $arrayname würde sich also auf das gesamte Array beziehen, während sich $arrayname[Montag] auf das eine spezielle Element des Arrays bezieht.

Um ein Array zu erstellen, bieten sich zwei Schreibweisen an:

Entweder Sie benutzen die Funktion `array()` oder den sogenannten Array-Bezeichner []. Die letztere Schreibweise ist nach unserem Geschmack etwas übersichtlicher und vielleicht auch komfortabler, da Sie den Arraynamen einfach kopieren können und dann die einzelnen Werte untereinander eingeben.

Um zu dem Beispiel Währungsumrechner zu kommen: Hierfür wird ein Array gebraucht mit den einzelnen Währungen und dem jeweiligen Wechselkurs. Wie oben schon erwähnt, wird den einzelnen Währungen als Wert der jeweilige Euro-Kurs zugewiesen. Sie geben also beispielsweise ein:

```
$arrayname[DM]=0.51;
```

Analog dazu werden dann die weiteren Array-Elemente festgelegt.

Bild 2 zeigt das Script für den Währungsumrechner. Schauen Sie sich das Script gut an. Im nächsten Abschnitt werden wir es Schritt für Schritt erläutern.

```
<html>
<head>
<title>Währungsumrechner</title>
</head>
<body>
<?php
if($betrag)
{
$rate[DM]=0.51;
$rate[Franc]=0.15245;
$rate[Peseten]=0.006010;
$rate[Euro]=1;
$rate1[DM]=1/$rate[DM];
$rate1[Franc]=1/$rate[Franc];
$rate1[Peseten]=1/$rate[Peseten];
$rate1[Euro]=1;
$ergebnis=$rate[$geld]*$betrag*$rate1[$geld1];
echo "<br>$betrag $geld sind $ergebnis $geld1<br>"; }//ENDE IF BETRAG
echo "<p><h3>Währungs-Umrechner!</h3>";
echo "<FORM ACTION="berechnung2.php" METHOD=post>Ausgangswährung<p>";
echo "<select size=1 name=geld>";
echo "<option";
if($geld=='DM'){echo " selected ";}
echo ">DM";
echo "<option";
if($geld=='Franc'){echo " selected ";}
echo ">Franc";
echo "<option";
if($geld=='Peseten'){echo " selected ";}
echo ">Peseten";
echo "<option";
if($geld=='Euro'){echo " selected ";}
echo ">Euro";
echo "</select><p>";
echo "Zielwährung<p>";
echo "<select size=1 name=geld1>";
echo "<option";
```

```
if($geld1=='DM'){echo " selected ";}
echo ">DM";
echo "<option";
if($geld1=='Franc'){echo " selected ";}
echo ">Franc";
echo "<option";
if($geld1=='Peseten'){echo " selected ";}
echo ">Peseten";
echo "<option";
if($geld1=='Euro'){echo " selected ";}
echo ">Euro";
echo "</select><p>";
echo "Betrag <input type=text name=betrag value=$betrag>";
echo "<input type=submit Name='submit' Value='berechnen'> </form>
</body>
</html>";
?>
```

Erläuterung

Nach dem PHP-Anfangszeichen schreiben Sie zunächst die `if`-Anweisung

```
if($betrag) {
```

Damit geben Sie an, dass alles, was im folgenden Block steht, nur geschieht, wenn das Feld BETRAG ausgefüllt wurde. Sie erinnern sich: die Prüfung einer **Variablen** gibt wahr zurück, wenn sie nicht leer ist.

Zunächst erstellen Sie ein Array mit dem Namen `rate`. In eckigen Klammer steht der jeweilige Schlüssel oder Array-Bezeichner und nach dem Gleichheitszeichen der Wert, in diesem Fall der jeweilige Euro-Kurs.

Das zweite Array trägt den Namen $rate1. Mit den Werten dieses Arrays wird die Berechnung definiert und gespeichert, die notwendig ist, um den Betrag in Euro zu konvertieren. $rate1[DM] steht beispielsweise für: 1 wird geteilt durch den Euro-Kurs für DM.

In der Variablen $ergebnis wird festgelegt, welche Berechnung durchgeführt wird, um den Betrag in die gewünschte Zielwährung umzurechnen. Die Variable $geld korrespondiert mit dem Namen des HTML-Auswahlfeldes für die Ausgangswährungen, gleichermaßen korrespondiert die Variable $geld1 mit dem Namen des Auswahlfeldes für die Zielwährungen.

```
$ergebnis=$rate[$geld]*$betrag*$rate1[$geld1];
```

Gerechnet wird folgendermaßen: der Wert, der in $rate für die im Auswahlfeld geld ausgewählte Währung festgelegt ist, multipliziert mit dem Betrag; das Ergebnis (dies ist dann der Betrag in Euro) wird

multipliziert mit dem Wert, der mit $rate1 definiert ist. Gesagt wurde, dass $rate1 der Umkehrwert des jeweiligen Kurses der ausgewählten Zielwährung, also der Rückrechenkurs von Euro in die Zielwährung ist.

Nach der Festlegung der Berechnung geben Sie mit `echo` und den oben festgelegten Variablen an, was als Ergebnis angezeigt werden soll:

```
echo "<br>$betrag $geld sind $ergebnis $geld1<br>";
```

Im Klartext also: ausgegeben werden der Betrag, die Ausgangswährung und das Ergebnis in der neuen Währung.

Die Darstellung auf der Seite

Der zweite Teil des Scripts definiert die Darstellung, also den Text und die Felder der Seite. Denken Sie daran: Da der PHP-Teil nicht beendet wurde, muss für die Ausgabe aller Elemente jeweils die Funktion echo benutzt werden. Die Zeile:

```
echo "<p><h3>Währungs-Umrechner!</h3>";
```

bildet die Überschrift der Seite. Danach wird mit dem Form-Tag das Formular eingeleitet und mit `Post` die Übertragungsmethode festgelegt. Da das Formular angezeigt bleibt, muss hier die aktuelle Datei angesprochen werden.

Die Zeile:

```
echo "<select size=1 name=geld>";
```

öffnet die erste Auswahlliste und legt mit dem Attribut `name` den Namen des Feldes fest, der von PHP als Variablenname verwendet wird.

Die nächsten Zeilen verbinden die Elemente der Auswahlliste, die durch den Tag `<option>` eingeleitet werden, mit einer `if`-Bedingung. Dies mag etwas verwirrend sein, aber hat natürlich einen Grund. Mit dieser Schreibweise lässt sich erreichen, dass auch nach der Antwort, also nachdem der Betrag in der gewünschten Währung angezeigt wurde bzw. wird, die jeweilige Option (sprich: Währung) in der Liste ausgewählt bleibt. Dass eine Standardauswahl erzwungen wird, lässt sich mit dem Attribut `selected` im Option-Tag erreichen. Wir haben nun erst mit `echo` die Option ausgegeben, dann die `if`-Verzweigung eingebaut und im Block angewiesen, das Attribut `selected` auszugeben:

```
echo "<option";
if($geld=='DM'){echo " selected ";}
```

```
echo ">DM";
```

Wenn also die Prüfung des Variableninhalts $geld durch den Vergleich (==) `true` ergibt, dann soll das Attribut `selected` wirksam werden. Nach der Unterbrechung durch die `if`-Bedingung wird `<option>` mit dem vorangestellten `echo` geschlossen und das Element der Liste eingegeben.

```
echo ">DM";
```

Diese Reihe wird genauso fortgesetzt für die anderen Währungen.

Nach dem Schließen des Tags `<select>` durch `</select>` bauen Sie die Auswahlliste für die Zielwährung genauso auf, nur dass Sie einen anderen Namen, im Beispiel geld1, vergeben. Dies hat den gleichen Effekt: Die gewählte Zielwährung bleibt ausgewählt.

```
echo "<option";
if($geld1=='DM'){echo " selected ";}
echo ">DM";
```

Zu guter Letzt fügen Sie noch das Feld für den Betrag ein. Dazu schreiben Sie

```
echo "Betrag <input type=text name=betrag value=$betrag>";
```

Mit der Festsetzung des `value= $betrag` bleibt der eingegebene Betrag auch nach dem Absenden der Berechnung im Feld stehen.

Nun fehlt nur noch die Schaltfläche zum Absenden der Berechnung.

```
echo "<input type=submit Name='submit' Value='berechnen'>
```

Dann wird der Form-Container beendet und erst dann auch die PHP-Auswertung.

Speichern Sie das Script und testen Sie es in Ihrem Browser, indem Sie die Felder ausfüllen und eine Berechnung abschicken.

> Wenn alles richtig eingegeben wurde, müssten Sie ein korrektes Ergebnis erhalten.

Aber es gibt nach wie vor einen deutlichen Schönheitsfehler, oder? Das Ergebnis wird weder auf-noch

abgerundet und mit viel zu vielen Nachkommastellen angezeigt.

Diese Crux sollte noch behoben werden. Mit anderen Worten: im nächsten Abschnitt werfen wir einen Blick auf die Möglichkeiten, Zahlen zu formatieren und zu manipulieren und werden dann das Script entsprechend ergänzen.

Zahlen manipulieren

PHP bietet einige vordefinierte mathematische Funktionen, um das Verhalten von Zahlen zu beeinflussen. Einige davon sollen hier vorgestellt werden.

Eine der mathematischen Funktionen ist die Abrundungsfunktion `round();`

Entsprechend der Standard-Konvention wird mit dieser Funktion jede Zahl über .50 (und .50 selbst) zur nächsten Ganzzahl aufgerundet, jede Zahl unter .50 zur nächsten Ganzzahl abgerundet.

Es gibt zwei Möglichkeiten, die Funktion `round()` anzuwenden. Sie können eine Dezimalzahl auf eine Ganzzahl auf- oder abrunden. Dazu legen Sie eine Variable fest und weisen mit der Funktion einen Wert zu, z.B.:

```
$zahl = round(8.52);
```

$zahl wäre dann 9. Probieren Sie das in einem kleinen Zweizeiler aus. Das Bild zeigt das Script und die Ausgabe:

Sie können aber auch auf eine bestimmte Anzahl von Dezimalstellen auf- oder abrunden. In diesem Fall benötigen Sie in der Klammer ein zweites Argument, mit dem Sie angeben, wie viele Dezimalstellen die Zahl haben soll.

```
$zahl = round(8.52, 1);
$zahl wäre dann = 8.5
```

Das heißt: die 2 von der .52 wird abgerundet, eine Nachkommastelle bleibt, ausgedrückt mit dem Argument 1. Achten Sie auf die **Syntax**. Zur Erinnerung: Die Argumente in der Klammer einer Funktion werden durch Kommata abgetrennt.
Info

```
double round(double val[int precision])
```

Der Befehl rundet eine Zahl (`val`) auf oder ab. Mit dem Parameter `precision` kann man die Nachkommastellen festlegen, nach denen gerundet wird.

Eine weitere Funktion, die im Script noch verwendet werden soll, ist die Funktion

```
number_format();
```

Mit dieser Funktion können Zahlen formatiert werden. In der Klammer müssen/können stehen: der zu formatierende Wert, die Anzahl der Nachkommastellen, das Zeichen vor den Nachkommastellen, das Tausenderzeichen. Je nach Parameter lassen sich mit dieser Funktion unterschiedliche Zahlenformate durchsetzen. Ein Beispiel:

```
echo number_format(1000,2,",",".");
```

Die Ausgabe wäre die folgende: 1.000,00.
Info

```
string number_format(float zahl [, int dezimalstellen [, string
Dezimaltrennzeichen [, string Tausendertrennzeichen]]])
```

Die Funktion formatiert eine Zahl. Die Parameter bestimmen die Formatierung: Anzahl der Nachkommastellen, das Zeichen vor den Nachkommastellen und das Tausendertrennzeichen.

Das Script wird also um zwei Zeilen ergänzt. Da es ja um die Ausgabe des Ergebnisses geht, ist die Variable die bereits definierte Variable $ergebnis.

```
$ergebnis=round($ergebnis,'2');
$ergebnis=number_format($ergebnis,2,",",".");
```

Diese Zeilen müssen vor dem `echo`, mit dem die Anzeige des Ergebnisses ausgelöst wird, eingefügt werden. Im Script sieht das dann so aus:

```
$ergebnis=$rate[$geld]*$betrag*$rate1[$geld1];
$ergebnis=round($ergebnis,'2');
$ergebnis=number_format($ergebnis,2,",",".");
echo "<br>$betrag $geld sind $ergebnis $geld1<br>";
```

Testen Sie nun den Währungsumrechner erneut. Sofern die Anweisung geklappt hat, würde ein Ergebnis nun etwa so dargestellt werden:

Weitere Vereinfachungen

Info
In diesem Abschnitt werden wir u.a. eine `while`-Schleife einsetzen. Sofern Sie sich an dieses Konstrukt noch nicht herantrauen, können Sie den Abschnitt natürlich überspringen, und später nach Lust und Laune zu diesem Beispiel zurückkehren.

Das bisherige **Script** berechnet den Betrag in der Zielwährung in einer kurzen Zeile, allerdings werden zuvor zwei Arrays definiert $rate und $rate1, wobei Array $rate1 immer der Umkehrwert des entsprechenden Elements im Array $rate zugewiesen wird. Diesen Schritt haben wir eingefügt, um die Lesbarkeit zu erhöhen. Aufgrund des feststehenden Zusammenhangs von $rate zu $rate1 kann man sich die Definition und Wertzuweisungen des Arrays $rate1 schenken, wenn man die Berechnung des Ergebnisses noch einmal ändert:

```
$ergebnis=$rate[$geld]*$betrag/$rate[$geld1];
```

Eine weitere Vereinfachung ist es, die Optionen der beiden `<select>`-Tags per Script aus den Angaben des Arrays $rate zu generieren. Mit dieser Methode reicht es, das Array $rate um ein neues Element zu ergänzen; die Auswahloptionen der `<select>`-Felder werden dann automatisch angepasst. Um diese Automatisierung zu realisieren, wird für jedes Element des Arrays $rate das `<option>`-Tag ausgegeben.

Es bietet sich also an, eine `while`-Schleife zum Einsatz zu bringen, um alle Elemente des Arrays $rate „abzuarbeiten“. Als Text soll bei den einzelnen Optionen der Schlüssel des entsprechenden Arrayelements angezeigt werden, also DM bei $rate[DM], Euro bei $rate[Euro]. Um die Schlüssel eines assoziativen Arrayelements zu erhalten, steht die Funktion `key()` parat. Die Zeilen lauten:

```
while($schluessel=key($rate))
{
echo "<option>$schluessel";
next($rate);
}
```

Zu Beginn wird mit $schluessel=key($rate) der Schüsselwert des ersten Elements des Arrays $rate der Variablen $schluessel zugewiesen. In $schluessel steht jetzt also das Währungskürzel des ersten Elements des Arrays $rate. Sofern Sie dem Beispiel bis hierher genau gefolgt sind, ist es „DM“. Diese Zuweisung dient auch gleichzeitig als Abbruchbedingung der `while`-Schleife, aber dazu gleich mehr.

Anschließend wird mit `echo "<option> $schluessel"` die erste Option des `<select>`-Tags ausgegeben – im Beispiel also `"<option>DM"`. Auf das schließende `</option>`-Tag wird hier verzichtet, da es in HTML nicht zwingend vorgeschrieben ist. Die Ausgabe des `value`-Attributs können Sie weglassen, da der angezeigte Text als Wert übertragen werden soll. In der nächsten Zeile wird mit dem Befehl `next()` das nächste Element des Arrays $rate angesprungen.

Anschließend ist der Anweisungsblock der `while`-Schleife beendet, die Abbruchbedingung der Schleife wird erneut getestet, um festzustellen, ob die Schleife nochmals durchlaufen werden soll. In diesem Fall wird mit $schluessel=key($rate) der Schlüssel des zweiten Elements des Arrays $rate der Variablen $schluessel zugewiesen, da zuvor mit `next($rate)` das nächste Element des Arrays ausgewählt wurde. Im Beispiel enthält $schluessel dann „Franc“, entsprechend wird also `"<option>Franc"` ausgegeben.

Ist die `while`-Schleife so häufig durchlaufen, dass das letzte Element des Arrays $rate abgearbeitet wurde, wird $schluessel=key($rate) `false`, da das Ende des Arrays $rate erreicht wurde und kein Element mehr vorliegt. Die `while`-Schleife wird also abgebrochen.

Damit nach dem Absenden die getätigte Auswahl der Währungen wieder angezeigt wird, muss die `if`-Anweisung noch eingefügt werden, die das Attribut `selected` bei der zuvor gewählten Option in das `<option>`-Tag ausgibt. Die gesamte Schleife sieht dann folgendermaßen aus:

```
while($schluessel=key($rate))
{
echo "<option ";
if($geld==$schluessel){echo " selected ";}
echo " >$schluessel";
next($rate);
}
```

Bedenken Sie bitte für den weiteren Programmablauf, dass nach dieser `while`-Schleife das Ende des Arrays $rate erreicht ist. Möchten Sie noch einmal auf das Array $rate zugreifen, müssen Sie es zurücksetzen. Im Beispiel werden sowohl die Optionen für das `<select>`-Auswahlfeld mit dem Namen geld als auch die Optionen für das `<select>`-Auswahlfeld geld1 aus den Schlüsselwerten des Arrays $rate erzeugt, sodass das Array zweimal durchlaufen werden muss. Um nach dem ersten Durchlauf wieder auf das erste Element eines Arrays zu verweisen, steht der Befehl `reset()` bereit. Fügen Sie also vor dem nächsten Zugriff auf das Array $rate die Zeile `reset($rate);` ein.

Beachten Sie bitte, dass noch eine kleine Änderung am Ausgangsbeispiel notwendig ist, damit die Optionen mit Hilfe der `while`-Schleifen erzeugt werden können. Das Array $rate muss immer erzeugt werden (nicht nur, wenn das Formular abgeschickt worden ist), damit die Schlüsselwerte aus dem Array gewonnen werden können und die `while`-Schleifen entsprechend der Anzahl der Elemente durchlaufen werden. Aus diesem Grund muss die `if`-Anweisung, die testet, ob $sent gesetzt ist, nach der Zuweisung der Arraywerte einsetzen.

```
<html>
<head>
<title>Währungsumrechner</title>
</head>
<body>
<?php
```

```
$rate[DM]=0.51;
$rate[Franc]=0.15245;
$rate[Peseten]=0.006010;
$rate[Euro]=1;
if($betrag)
{
$ergebnis=$rate[$geld]*$betrag/$rate[$geld1]; $ergebnis=round($ergebnis,'2');
$ergebnis=number_format($ergebnis,'2',',','.');
echo "<br>$betrag $geld sind $ergebnis $geld1<br>"; }//ENDE IF BETRAG
echo "<h3>Währungs-Umrechner!</h3>";
echo "<FORM ACTION='berechnung2.php' METHOD=post>Ausgangswährung<p>";
echo "<select size=1 name=geld>";
while($schluessel=key($rate))
{
echo "<option ";
if($geld==$schluessel){echo " selected ";}
echo " >$schluessel";
next($rate);
}
echo "</select><p>";
echo "Zielwährung<p>";
echo "<select size=1 name=geld1>";
reset($rate);
while($schluessel=key($rate))
{
echo "<option ";
if($geld1==$schluessel){echo " selected ";}
echo " >$schluessel";
next($rate);
}
echo "</select><p>";
echo "Betrag <input type=text name=betrag value=$betrag>";
echo "<input type=submit Name='submit' Value='berechnen'>
</form>
</body>
</html>";
?>
```

Eine Währung hinzufügen

Um die Flexibilität des Umrechners zu testen, ergänzen Sie den Umrechner einfach um eine weitere Währung, indem Sie ein neues Array-Element hinzufügen, z.B. $rate[Lire]= 0.0005165.

Rufen Sie die Datei dann erneut auf bzw. aktualisieren Sie sie und prüfen, ob die Ergänzung geklappt hat. In beiden Auswahlfeldern müsste, wie in Bild zu sehen, die Lire auftauchen:

Ausblick

Der Euro wird die Umrechnung der einzelnen Währungen untereinander in absehbarer Zeit überfällig werden lassen. Dennoch müssen Sie das kleine Script nicht in die ewigen Jagdgründe schicken. Wenn Sie bereit sind, die schwankenden Wechselkurse zu aktualisieren, können Sie x-beliebige Währungen einsetzen. Achten Sie aber darauf, dass sich die Wechselkurse im Array $rate immer auf eine „Basiswährung“ beziehen müssen. Haben Sie das Script so angepasst, wie es im letzen Abschnitt beschrieben wurde, reicht es, das Array $rate zu aktualisieren.

Wenn es Ihnen zu mühselig ist, die Kurse tägliche im Script zu aktualisieren, können Sie noch eine Erweiterung einbauen: sie schreiben die Wechselkurse jeder Währung in eine TXT-Datei und dann lesen Sie im Währungsumrechner den Wert dieser Datei in das entsprechende Array-Element von $rate ein. (Der Hinweis soll an dieser Stelle genügen.

Anschließend erstellen Sie ein kleines Script, um die Einträge der Dateien zu aktualisieren. Hier können Sie z.B. die Uploadfunktion nutzen, um Dateien mit aktualisierten Werten per Browser auf den Webserver zu legen.

Wenn Sie ein Anhänger von Datenbanken sind, können Sie die Wechselkurse auch in einer Datenbanktabelle speichern und für das Umrechnungsscript aus der Datenbank auslesen. Zum Aktualisieren der Wechselkurse benötigen Sie dann noch ein Formular, das Ihnen die Wechselkurse der Datenbank auflistet und Änderungen in die Datenbank schreibt. Dies ist sicherlich die eleganteste Methode, allerdings auch mit einigem Aufwand verbunden.

Nach den ersten praktischen Eindrücken von PHP wird es nun Zeit, sich mit den Details der Sprache zu beschäftigen. Denn in der Praxis benötigen Sie eine genaue Kenntnis der Sprachsyntax und Sie müssen über den Geltungsbereich von Variablen, über Datentypen und viele andere Sprachbesonderheiten Bescheid wissen.

Anweisungen, Blöcke, Ausdrücke und Kommentare

Ein PHP-Script oder ein PHP-Script-Bereich besteht in einer Folge von Anweisungen. Anweisungen können auch in Blöcken stehen, die nur unter bestimmten Bedingungen ausgeführt werden, z.B. in einem Funktionsblock, der nur ausgeführt wird, wenn die Funktion aufgerufen wird, oder in einem Block, der von einer `if`-Abfrage abhängig ausgeführt wird. Jede Anweisung wird mit einem Semikolon (`;`) beendet. Nur bei der letzten Anweisung eines Scripts oder Script-Bereichs darf das Semikolon entfallen.

Anweisungsblöcke werden in geschweifte Klammern `{` und `}` eingeschlossen. Bei Funktionen sind diese Klammern in jedem Fall Pflicht. Bei `if-else`-Konstrukten oder bei `while`– oder `for`-Schleifen dürfen die geschweiften Klammern dann entfallen, wenn nur eine abhängige Anweisung folgt.

Anweisungen können sehr unterschiedlich aussehen, z.B.:

```
$name = "Stefan";
```

```
include_once("andere.php");
echo "Server: ", $_SERVER['SERVER_NAME'];
$bytes = filesize("daten.txt");
$print_code = htmlspecialchars(file_get_contents("daten.html"));
```

Im ersten Fall wird einer Variablen ein Wert zugewiesen. Im zweiten Fall wird eine PHP-Funktion aufgerufen und einfach ausgeführt. Im dritten Fall wird ein so genanntes Sprachkonstrukt ausgeführt, nämlich `echo`. Übergeben wird eine kommagetrennte Aufzählung von Werten, die auszugeben sind. Im vierten Fall wird ein Funktionsaufruf einer Variablen zugewiesen. Diese Art von Anweisung ist immer dann sinnvoll, wenn Funktionen Werte ermitteln und zurückgeben. Der zurückgegebene Wert kann dann in der Variablen gespeichert werden. Auch die letzte Beispielanweisung gehört zu diesem Typ. Hier wird die PHP-Funktion `htmlspecialchars()` aufgerufen. Diese erwartet als Parameter eine Zeichenkette. Statt ihr eine feste Zeichenkette zu übergeben, können wir ihr jedoch auch eine Variable übergeben, in der eine Zeichenkette gespeichert ist, oder sogar, wie im Beispiel, den Aufruf einer anderen Funktion, die eine Zeichenkette zurückgibt.

Ausdrücke

In der Programmierpraxis ist vielfach auch von Ausdrücken die Rede. Bereits in Zusammenhang mit JavaScript wurde dies kurz behandelt. Ausdrücke sind Teile einer Anweisung, in denen ein Wert erzeugt wird. Dazu sind in aller Regel Operatoren erforderlich.

```
$name = $vorname." ".$zuname;
$x = 4 * (7 + 14);
```

In beiden Beispielen ist der Ausdruck das, was rechts vom Zuweisungsoperator, also vom Gleichheitszeichen steht. Im ersten Fall wird eine Zeichenkette aus mehreren anderen zusammengesetzt und im zweiten Fall wird eine Berechnung durchgeführt. Viele Anweisungen haben diesen Aufbau: Ein Ausdruck wird einer Variablen zugewiesen. Damit wird der vom Ausdruck erzeugte Wert gespeichert.

```
$x = 3 < 4;
echo $x;
```

Ausgegeben wird in diesem Beispiel 1, weil in $x die Bewertung des Ausdrucks 3 < 4 gespeichert wird. Da < ein Vergleichsoperator ist, ist das „Ergebnis“ des Ausdrucks ein Wahrheitswert. Denn der Vergleich trifft entweder zu oder nicht. In unserem Beispiel trifft er zu, weil 3 tatsächlich kleiner ist als 4. In $x wird also „wahr“ gespeichert. Die numerische Entsprechung für „wahr“ ist in PHP wie in vielen anderen Programmiersprachen 1.

Kommentare

Zu PHP kommen auch viele, die schon in anderen Sprachen programmiert haben. Deshalb ist PHP sehr großzügig, was die Syntax von Kommentaren betrifft: Jeder darf seinen früheren „Stil“ beibehalten, ob er nun von Perl herkommt oder von C.

Einzeilige Kommentare werden entweder durch ein Gatterzeichen (#) oder durch zwei Schrägstriche (//) eingeleitet. Steht dieses Kommentarsignal am Zeilenanfang, wird die gesamte Zeile auskommentiert. Steht sie an einer späteren Stelle, wird der Teil davor als ausführbarer Code betrachtet.

```
# ----------------------------------------
// Alles Kommentar
# ----------------------------------------
$x = "foo"; // Programmierer-Slang, Teil 1
$y = "bar"; # Programmierer-Slang, Teil 2
```

Mehrzeilige Kommentare können Sie durch /* einleiten und mit */ beenden.

```
/* ----------------------------------------
Alles Kommentar
---------------------------------------- */
```

Kommentare können selbstverständlich auch PHP-Code enthalten. Dadurch können Sie beispielsweise momentan nicht benötigten oder noch unfertigen Code ausblenden.

Auswahlelemente erleichtern dem Benutzer die Bedienung. Dadurch verringern sie gleichzeitig die Möglichkeit, Fehler bei der Eingabe zu machen. Falls möglich, sind sie den Textfeldern vorzuziehen, da der Mehraufwand im PHP-Programm für das Abfangen der fehlerhaften Eingaben in keinem Verhältnis zum Mehraufwand der **HTML-Codierung** der Auswahlfelder steht. Bei den Auswahlelementen unterscheidet man:

- einfache Auswahlelemente wie Radiobutton-Gruppen oder das einfache Auswahlmenü, bei denen der Benutzer genau einen Eintrag auswählen kann
- mehrfache Auswahlelemente wie Kontrollkästchen oder das mehrfache Auswahlmenü, bei denen der Benutzer mehrere Einträge auswählen kann

Einfache **Auswahlelemente** sollten Sie vorbelegen. Auf diese Weise können Sie verhindern, dass ein Formularelement ohne Wert übertragen wird. Dies verringert gleichzeitig den Aufwand im PHP-Programm.

Radiobutton-Gruppe

Eine Auswahl kann zum Beispiel über eine Gruppe von Optionsschaltfeldern, auch Radiobuttons genannt (`<input type="radio" />`), getroffen werden.

```
<html>
<body>
<p>Bitte treffen Sie jeweils eine Auswahl
<br />
und senden Sie das Formular ab:</p>
<form action = "radio.php" method = "post">
<h3>Reiseziel</h3>
<p><input type="radio" name="rziel" value="Gomera" checked="checked" />
Wandern auf Gomera
<br />
<input type="radio" name="rziel" value="Lanzarote" />
Sonnen auf Lanzarote
<br />
<input type="radio" name="rziel" value="Fuerteventura" />
Surfen auf Fuerteventura</p>
<h3>Hoteltyp</h3>
<p><input type="radio" name="htyp" value="Drei" checked="checked" />
Drei-Sterne-Hotel
<br />
<input type="radio" name="htyp" value="Vier" />
Vier-Sterne-Hotel</p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Datei radio.htm

In diesem Formular werden zwei Gruppen von **Radiobuttons** dargestellt. Die Elemente einer Gruppe haben den gleichen Namen, dadurch wird eine logische Zusammengehörigkeit hergestellt. Optisch werden die beiden Gruppen durch Überschriften voneinander getrennt.

Innerhalb der ersten Gruppe kann der Betrachter ein Reiseziel auswählen, innerhalb der zweiten Gruppe einen Hoteltyp. In beiden Gruppen ist ein Element vorbelegt: Dafür sorgt die Eigenschaft `checked` mit dem Wert `checked`.

Nach Absenden des Formulars erhält der Benutzer eine Antwort vom Webserver mit der Anzahl der Angebote für die von ihm gewählten Kriterien. Das Formular, nach Bedienung durch den Benutzer, sehen Sie in Abbildung 1.

Die Antwort wird durch das folgende Programm geliefert:

```
<html>
```

```
<body>
<?php
echo "Sie möchten also nach " . $_POST["rziel"];
echo " in ein " . $_POST["htyp"] . "-Sterne-Hotel<br />";
if ($_POST["rziel"] == "Gomera")
{
if ($_POST["htyp"] == "Drei") $ang = 7;
else $ang = 1;
}
else if ($_POST["rziel"] == "Lanzarote")
{
if ($_POST["htyp"] == "Drei") $ang = 12;
else $ang = 2;
}
else {
if ($_POST["htyp"] == "Drei") $ang = 5;
else $ang = 4;
}
echo "Dazu haben wir $ang Angebote";
?>
</body>
</html>
```

Datei radio.php

Der gemeinsame Name (Eigenschaft `name`) der ersten Optionsgruppe ist `rziel`. Nach Absenden des Formulars steht dadurch die **Variable** `$_POST["rziel"]` mit dem Wert (`value`) des vom Benutzer ausgewählten Eintrags im PHP-Programm zur Verfügung. Falls er zum Beispiel Wandern auf Gomera auswählt, wird `$_POST["rziel"]` der Wert Gomera zugewiesen.

Der gemeinsame Name der zweiten Optionsgruppe ist `htyp`. Falls der Benutzer beispielsweise Drei-Sterne-Hotel auswählt, wird der Variablen `$_POST ["htyp"]` der Wert Drei zugewiesen.

Aus den Informationen in den Variablen wird im PHP-Programm mit Hilfe einer geschachtelten Verzweigung die Anzahl der vorliegenden Angebote ermittelt und in der Variablen $ang gespeichert. Der Wert dieser Variablen wird dem Betrachter zusammen mit einer Bestätigung seiner Eingabedaten zurückgesandt. Die Antwort auf die obige Eingabe zeigt Abbildung 2.

Einfaches Auswahlmenü

Einfache Auswahlmenüs (`select`-Menüs) erfüllen den gleichen Zweck wie Gruppen von Radiobuttons. Besonders bei zahlreichen Auswahlmöglichkeiten zeichnen sie sich durch ihren geringeren Platzbedarf innerhalb eines Formulars aus. Zum Vergleich soll daher nun das oben genannte Beispiel mit Hilfe von Auswahlmenüs dargestellt werden. Es kann in beiden Fällen das gleiche **PHP-Programm** angefordert

werden.

```
<html>
<body>
<p>Bitte treffen Sie jeweils eine Auswahl
<br />
und senden Sie das Formular ab:</p>
<form action = "radio.php" method = "post">
<p><select name="rziel">
<option value="Gomera">
Wandern auf Gomera </option>
<option value="Lanzarote" selected="selected">
Sonnen auf Lanzarote</option>
<option value="Fuerteventura">
Surfen auf Fuerteventura </option>
</select> Reiseziel</p>
<p><select name="htyp">
<option value="Drei" selected="selected">
Drei-Sterne-Hotel </option>
<option value="Vier">
Vier-Sterne-Hotel </option>
</select> Hotel-Typ</p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Datei select.htm

Zu den Unterschieden: Im Dokument erscheinen zwei aufklappbare Menüs (`<select> </select>`), in denen jeweils schon eine Auswahlmöglichkeit voreingestellt ist (Eigenschaft `selected` mit dem Wert `selected`). Die Namen der beiden Auswahlmenüs sind `rziel` und `htyp`, der Wert (Eigenschaft `value`) wird über die jeweils vom Benutzer ausgewählte Option eingestellt. Das PHP-Programm radio.php verarbeitet diese Informationen genauso wie bei den Radiobuttons.

Info

Falls im **Formular** die Eigenschaft `value` weggelassen wird, wird als Wert der dargestellte Text der ausgewählten Option (zwischen `<option>` und `</option>`) übermittelt.

Das Formular sieht im Startzustand wie in Abbildung 3 aus.

Kontrollkästchen

Mit Hilfe eines Kontrollkästchens (`<input type="checkbox" />`) kann der Benutzer eine einfache Ja-

/Nein-Auswahl treffen. Soll ein Kontrollkästchen bereits vorbelegt sein, so wird die Eigenschaft checked mit dem Wert checked hinzugefügt. Falls mehrere Kontrollkästchen zusammen verwendet werden, hat der Benutzer die Möglichkeit, keinen, einen oder mehrere Einträge aus einer zusammengehörigen Gruppe auszuwählen. Ein Beispiel:

```
<html>
<body>
<p>Wünschen Sie in Ihrem Zimmer:</p>
<form action = "check.php" method = "post">
<p><input type="checkbox" name="cb" value="Bad" checked="checked" />
Bad</p>
<p><input type="checkbox" name="cm" value="Meeresblick" />
Meeresblick</p>
<p><input type="checkbox" name="cz" value="Zimmertresor" />
Zimmertresor</p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Datei check.htm

In diesem Formular kann der Betrachter drei voneinander unabhängige Eigenschaften seines Hotelzimmers auswählen. Nach Absenden des Formulars bekommt er eine Antwort vom **Webserver** mit einer Bestätigung seiner Auswahl. Abbildung 4 zeigt das Formular.

Die Antwort liefert das folgende Programm:

```
<html>
<body>
<?php
echo "<p>Danke für Ihre Anfrage, wir reservieren:</p>";
if (isset($_POST["cb"]))
echo "Zimmer mit " . $_POST["cb"]. ", Aufpreis 10 €/Tag<br />";
if (isset($_POST["cm"]))
echo "Zimmer mit " . $_POST["cm"]. ", Aufpreis 15 €/Tag<br />";
if (isset($_POST["cz"]))
echo "Zimmer mit " . $_POST["cz"]. ", Aufpreis 5 €/Tag";
?>
</body>
</html>
```

Datei check.php

Die Namen der drei Kontrollkästchen werden wiederum zu Variablen des PHP-Programms. Sie haben hier eine doppelte Funktion.

- Zum einen wird mit Hilfe der Funktion `isset()` überprüft, ob das jeweilige Kontrollkästchen vom Benutzer ausgewählt wurde. Falls ja, existiert die betreffende Variable beziehungsweise das betreffende Element des assoziativen Feldes für das PHP-Programm. Das Ergebnis der Abfrage `if (isset($_POST["checkboxname"]))` ist wahr, und die darauf folgende Anweisung wird ausgeführt.
- Zum anderen beinhaltet die Variable einen Wert (Eigenschaft `value`). Dieser Wert kann im Programm zum Beispiel zur Ausgabe genutzt werden, wie im oben gezeigten Programm geschehen.

Falls das oberste Kontrollkästchen als einziges angekreuzt wurde, sieht die Antwort aus wie in Abbildung 5.

Mehrfaches Auswahlmenü

Den gleichen Zweck wie Gruppen von Kontrollkästchen erfüllen mehrfache **Auswahlmenüs** (`<select multiple="multiple"> ... </select>`). Auch hier gilt: Besonders bei zahlreichen Auswahlmöglichkeiten zeichnen sie sich durch ihren geringeren Platzbedarf innerhalb eines Formulars aus. Zum Vergleich soll das oben gezeigte Beispiel nun mit Hilfe eines mehrfachen Auswahlmenüs dargestellt werden:

```
<html>
<body>
<p>Wünschen Sie in Ihrem Zimmer:</p>
<form action = "multiple.php" method = "post">
<p><select multiple="multiple" name="zusatz[]">
<option value="Bad, Aufpreis 10 €/Tag">
Bad</option>
<option value="Meeresblick, Aufpreis 15 €/Tag" selected="selected">
Meeresblick</option>
<option value="Zimmertresor, Aufpreis 5 €/Tag" selected="selected">
Zimmertresor</option>
</select></p>
<p><input type = "submit" />
<input type = "reset" /></p>
</form>
</body>
</html>
```

Datei multiple.htm

Bei einem mehrfachen Auswahlmenü kann der Benutzer mit Hilfe der (`Strg`)-Taste (getrennte Einträge)

beziehungsweise (Shift) (benachbarte Einträge) seine Wahl treffen. Damit eine Auswertung durch PHP möglich ist, muss das Formularelement mit name= "zusatz[]" als Feld gekennzeichnet werden. Das Formular sehen Sie in Abbildung 6.

Die Antwort dazu sieht aus wie in Abbildung 7.

Die Auswertung durch ein PHP-Programm:

```
<html>
<body>
<?php
echo "<p>Danke für Ihre Anfrage, wir reservieren:</p>";
echo "<p>";
for($i=0; $i<sizeof($_POST["zusatz"]); $i++)
{
if (isset($_POST["zusatz"][$i]))
echo "Zimmer mit ".
$_POST["zusatz"][$i]. "<br />";
}
echo "</p>";
?>
</body>
</html>
```

Datei multiple.php

Auch hier wird die Funktion `isset()` verwendet, um die Existenz einer Variablen zu prüfen. Bei `$_POST` handelt es sich bekanntlich um ein assoziatives Feld. Die Daten des **Formularelements** zusatz werden außerdem in einem numerisch indizierten Feld geliefert. Daher ist das Feldelement `$_POST["zusatz"]` wiederum ein Feld.

Jedes Element dieses Feldes ist nur über die Angabe von zwei Indizes erreichbar: Zunächst muss der Name des assoziativen Feldelements angegeben werden (zusatz), anschließend der Index des numerisch indizierten Feldes (0, 1, 2, ...).

Falls einer oder mehrere Einträge ausgewählt wurden, existiert für PHP das Feldelement `$_POST["zusatz"]`. Dessen aktuelle Größe kann mit der Funktion `sizeof()` ermittelt werden.

- Falls genau ein Eintrag ausgewählt wurde, existiert für PHP nur das **Feldelement** `$_POST["zusatz"][0]`. Es beinhaltet den Wert des ausgewählten Eintrags.
- Falls mehrere Einträge ausgewählt wurden, existieren für PHP die Feldelemente `$_POST["zusatz"][0]`, `$_POST["zusatz"][1]` usw. – entsprechend der Anzahl der ausgewählten Einträge. Sie beinhalten der Reihe nach die Werte der ausgewählten Einträge.

Was sind Cookies?

Cookies stellen einen Mechanismus dar, durch den der Webserver die Möglichkeit hat, Informationen auf dem Client (in der Regel dem Webbrowser) zu speichern und abzufragen. Als Informationen bieten sich Einstellungen an, die der Benutzer bei einer Sitzung gemacht hat und beim nächsten Besuch wieder vorfinden soll. Im **Cookie** werden neben dem eigentlichen Inhalt Informationen zum Gültigkeitsbereich, der Gültigkeitsdauer und dem Absender gespeichert. Die maximale Größe eines Cookie liegt bei 4 Kilobyte.

> Auf dem Client werden Cookies im Profil des jeweiligen Benutzers gespeichert.

Damit ist gewährleistet, dass der Zugriff auf das Cookie nicht vom Arbeitsplatz, sondern von dem angemeldeten Benutzer abhängig ist. Zu Problemen kann das Ablegen von Informationen auf dem Client in Situationen führen, in denen unterschiedliche Benutzer unter dem gleichen Benutzerprofil arbeiten (wie das in Internetcafes oft der Fall ist): Werden hier Benutzername und Passwort in einem Cookie abgelegt, das nicht nach der Sitzung gelöscht wird, haben andere Benutzer Zugriff auf den passwortgeschützten Bereich.
Info
Sollten Sie an einem öffentlichen Computer auf private Bereiche (Webmail, oder gar Internetbanking) zugegriffen haben, empfiehlt es sich, nach der Arbeit nicht nur den Verlauf im Browser zu löschen, sondern auch die Cookies.

Anwendungsbereiche von Cookies

Die Verwendung von Cookies bringt vor allem bei längeren Arbeitsabläufen Vorteile. Sollen Informationen über mehrere Seiten transportiert werden (wie zum Beispiel bei einem mehrseitigen Fragebogen), so gibt es mehrere Möglichkeiten, die Daten **zwischenzuspeichern**:

- Die Eingaben werden über versteckte Eingabefelder (`input type="hidden"`) von Seite zu Seite mitgeschleppt – eine aufwändige und fehleranfällige Variante.
- Es werden Cookies verwendet, um die Informationen zu transportieren. Dazu wird vorausgesetzt, dass der Browser auch mit Cookies umgehen kann und dass die Sicherheitseinstellungen Cookies zulassen.
- Die im nächsten Abschnitt beschriebenen Sessions sind noch flexibler als Cookies, da sie nicht auf die clientseitigen Sicherheitseinstellungen angewiesen sind.

Im einfachen Fall wird in einem Cookie genau ein Wert gespeichert (zum Beispiel der Benutzername john). **PHP** bietet aber auch die Möglichkeit, eine `array`-Variable als Cookie abzulegen, was den Vorteil bringt, dass man die Werte in einer Schleife abarbeiten kann. Für den Fall, dass noch komplexere Daten in einem Cookie gespeichert werden sollen, gibt es die Möglichkeit, den Variableninhalt zu serialisieren (Methode: `serialize`), indem eine Zeichenkette aus der komplexen Variable erstellt wird. Bei all den Verwendungsmöglichkeiten darf man aber die maximale Größe von 4 Kilobyte nicht aus den Augen verlieren.

Nachteile von Cookies

Der große Nachteil von Cookies ist wohl, dass man sich nicht auf sie verlassen kann. Ob ein Browser die Cookies wirklich nach der Sitzung löscht, wie es ihm aufgetragen wurde, ob sie ein Benutzer nicht während einer Sitzung löscht, weil sie ihm als **Sicherheitsrisiko** erscheinen, oder ob sie überhaupt deaktiviert wurden, lässt sich im Vorhinein nicht sagen. Darum ist es generell keine gute Idee, bei sicherheitskritischen Aufgaben (wie passwortgeschützten Bereichen) auf Cookies zurückzugreifen.

Eine weitere Einschränkung von Cookies ist, dass sie nur vor dem Ausliefern des Seiteninhalts

erstellt oder verändert werden können. Diese Limitierung ist in der Definition des HTTP verankert und hängt nicht direkt mit PHP zusammen. Trotzdem muss man bei der Programmierung sehr aufmerksam sein, denn eine einzige Leerzeile reicht, um die Funktion der Cookies zu verhindern. Seit PHP 4 gibt es die Möglichkeit, die Ausgabe zu puffern, um so sicherzustellen, dass der Cookiemechanismus funktioniert.

Cookies werden erst nach dem neuerlichen Laden einer Seite sichtbar. Dieses Verhalten sorgt oft für Verwirrung, z.B. wenn man nach dem Setzen des Cookies gleich prüfen will, ob es auch gesetzt ist.

Cookies und Privatsphäre

Um eine standardisierte Aussage über Informationen machen zu können, die in Cookies gespeichert werden, hat das World Wide Web Consortium (W3C) eine Recommendation dazu herausgegeben (eine Recommendation des W3C können Sie sich wie einen Internet-Standard vorstellen; die Spezifikation von HTML 4.01 ist z.B. eine W3C-Recommendation). Die **P3P 1.0-Spezifikation** (Platform for Privacy Preferences) wurde im April 2002 in diesen Status erhoben und ist damit ein noch recht junger Standard.
Info

Auf den Internetseiten des W3C finden Sie ausführliche Informationen zu P3P. Wenn Sie vorhaben, auf Ihren Internetseiten personenbezogene Daten aufzunehmen, sollten Sie sich einen Überblick über diesen Standard verschaffen. W3C-Informationen

Ab der Version 6 des Internet Explorer hat sich das Standard-Cookie-Verhalten geändert. Damit Third-Party-Cookies akzeptiert werden, muss die Website entsprechende P3P-Informationen zur Verfügung stellen. Third-Party-Cookies sind Cookies, die beim Aufruf von Ihrer Seite durch eine andere Seite gesetzt werden. Eine häufige Anwendung für dieses Szenario sind Banner-Werbungen, die so ihre Spuren auf dem Client hinterlassen.

Sie erkennen eine Seite, deren Third-Party-Cookies nicht akzeptiert wurden,

an dem Auge mit dem roten Minus-Zeichen in der Status-Leiste des Internet Explorer. Natürlich können

diese Einstellungen vom Anwender beeinflusst werden, so dass diese Cookies auch ohne derartige Informationen akzeptiert werden. Aber Erfahrungswerte zeigen, dass nur sehr wenige Anwender das Defaultverhalten ändern.

Info

Wenn Sie `sessions` oder `cookies` verwenden, sollten Sie auf den betroffenen Seiten explizit darauf hinweisen, vielleicht mit einem Link auf eine Seite, die kurz erklärt, warum Cookies hier erforderlich sind und dass dadurch die Privatsphäre nicht gestört wird. Wenn Ihre Seite dann bei einem Anwender nicht funktioniert, weiß dieser zumindest, was die Ursache sein könnte, und kann entsprechend reagieren.

Cookies im Einsatz

Als Beispiel soll die Startseite für einen mehrseitigen Fragebogen programmiert werden. Es werden ein Benutzername und ein Passwort abgefragt, an einer lokalen Benutzertabelle überprüft und in einem Cookie abgelegt. Ein zusätzliches Eingabefeld ermöglicht es dem Benutzer, sich eine Hintergrundfarbe aussuchen, in der die weiteren Seiten angezeigt werden. Auch diese Information wird in einem Cookie transportiert.

Alle Dateien des Beispiels befinden sich im Verzeichnis `cookie` und haben folgende Struktur:

- **login_cookie.php** Erzeugt die Eingabemaske für Benutzername und Passwort. Kann über den Parameter `func=logout` die Cookies löschen.
- **check_cookie_user.php** Überprüft das Passwort und setzt die Cookies.
- **page_1.php** Die erste Seite des Fragebogens

Um standardkonformen **HTML-Code** zu erzeugen, kommt noch die Bibliothek html.inc.php zum Einsatz, die schon in PHP-Authentifizierung beschrieben wurde.

login_cookie.php

```
<?php // Datei cookie/login_cookie.php
if ($_GET['func'] === 'logout') {
setcookie("my_user", "");
setcookie("my_pass", "");
setcookie("my_bgcolor", "");
header('Location: login_cookie.php');
}
require_once "../lib/html.inc.php";
if (!isset($_COOKIE['my_user'])) {
html4head("Bitte anmelden");
echo '<body><form name="user_login" method="post"
action="check_cookie_user.php">';
echo '<table>';
echo '<tr>
<td>User</td>
<td><input type="text" name="user_id" value=""></td>';
```

```
echo '<tr>
<td>Pass</td>
<td><input type="password" name="user_pass" value=""></td>';
echo '<tr>
<td>Hintergrundfarbe</td>
<td>
<select name="bg_color">';
echo '<option value="lavender">lavender</option>';
echo '<option value="beige">beige</option>';
echo '<option value="tan">tan</option>';
echo '<option value="white">white</option>';
echo '</select>
</td>
</tr>
</table>';
echo '<input type="submit" value="abschicken" name="submit">';
echo '</form>';
}
else {
html4head("Bereits angemeldet");
echo '<body>
<p>Sie sind bereits angemeldet</p>';
echo '<p>Wenn Sie nicht <b>'.$_COOKIE['my_user'].'</b> sind, können Sie sich <a
href="login_cookie.php?func=logout">hier abmelden</a>.</p>';
echo '<a href="page_1.php">Zur 1. Seite</a>';
}
htmlend();
?>
```

Die Datei login_cookie.php ist die Startseite dieses Beispiels. Wird diese Datei in der Form `login_cookie.php?func=logout` aufgerufen, so wird die erste `if`-Abfrage ausgeführt und es werden alle drei Cookies gelöscht (durch Setzen auf einen leeren Wert). Im Anschluss wird das Script noch einmal ohne Parameter aufgerufen. Ist das Cookie mit dem Benutzernamen noch nicht vorhanden, wird der HTML-Code zur Anmeldung angezeigt. Sollte das Cookie noch vorhanden sein, wird man darauf hingewiesen und kann die **Passwortüberprüfung** überspringen oder sich neu anmelden.

check_cookie_user.php

```
<?php // Datei cookie/check_cookie_user.php
require_once "../lib/html.inc.php";
$user_passwd = array (
'john' => 'pu88jksd',
'marc' => 'lks31lsk'
);
```

```
if (isset($_POST['submit'])) {
if ($_POST['user_pass'] === $user_passwd[$_POST['user_id']]) {
/* Cookies sind 1 Stunde gültig */
setcookie("my_user", $_POST['user_id'], time()+3600);
setcookie("my_pass", md5($_POST['user_pass']), time()+3600);
setcookie("my_bgcolor", $_POST['bg_color'], time()+3600);
header('Location: page_1.php');
}
else {
html4head("Login falsch");
echo '<body>Falsch: Username oder Password. ';
echo '<a href="login_cookie.php">Neuer Versuch</a>';
htmlend();
exit();
}
}
if (!isset($_COOKIE['my_user'])) {
html4head("Bitte anmelden");
echo '<body>
<a href="login_cookie.php">Bitte Anmelden</a>';
}
else {
html4head("Bereits angemeldet");
echo '<body>
<p>Sie sind bereits angemeldet</p>';
echo '<p>Wenn Sie nicht <b>'.$_COOKIE['my_user'].'</b> sind, können Sie sich <a
href="login_cookie.php?func=logout">hier abmelden</a>.</p>';
echo '<a href="page_1.php">Zur 1. Seite</a>';
}
htmlend();
?>
```

In diesem Programm werden nach erfolgreicher Passwortüberprüfung die Cookies gesetzt. Die Zeitangabe (`time()+3600`) wird als UNIX-Timestamp erwartet, also in Sekunden seit dem Beginn der UNIX-Epoche. Üblicherweise macht man sich die PHP-Funktion `time()` zunutze, die die aktuelle Sekundenzahl der Epoche ausgibt, und addiert die gewünschte Anzahl an Sekunden. Um beispielsweise ein Cookie für eine Woche gültig zu halten, wäre die Anweisung `time()+3600*24*7` ausreichend.

Nach dem Setzen der Cookies wird man auf die erste Seite des Fragebogens weitergeleitet. Ist der Benutzername oder das Passwort falsch, erscheinen eine Fehlermeldung und der Link zum neuerlichen Anmelden. Auch in dieser Datei wird auf ein bestehendes Cookie überprüft und man hat die Möglichkeit, sich neu anzumelden.

page_1.php

```
<?php // Datei cookie/page_1.php
require_once '../lib/html.inc.php';
if (!isset($_COOKIE['my_user'])) {
header('Location: login_cookie.php');
}
$bg_color = "";
if (isset($_COOKIE['my_bgcolor'])) {
$bg_color = 'bgcolor="'.$_COOKIE['my_bgcolor'].'"';
}
html4head("Seite 1");
echo "<body $bg_color>";
?>
<h1>Willkommen bei unserer Befragung</h1>
<p><a href="login_cookie.php?func=logout">abmelden</a></p>
</body>
</html>
```

Der Quelltext der ersten Seite des Fragebogens wertet die Cookies aus: Ist der Benutzername nicht im Cookie gespeichert, wird man zur Anmeldung weitergeleitet. Wenn das Cookie mit der Hintergrundfarbe gesetzt ist, wird diese über das **HTML-Tag** `<body bgcolor="XXX">` eingebaut und die Seite erscheint in der gewünschten Farbe. Abschließend wird noch ein Link zum Abmelden angezeigt.

Das Beispiel zeigt in wenigen Schritten, wie Cookies gesetzt, verändert und gelöscht werden können. Will man den Fragebogen ausbauen, so kann man sich den erweiterten Möglichkeiten von Cookies widmen, wie etwa dem Speichern von Array-Variablen in einem Cookie.

Bereits in PHP 4 flossen mit der Zeit immer mehr objektorientierte Funktionen in die Programmiersprache ein. In Version 5 wurde die **Objektorientierung** komplett überarbeitet. Der neue, übersichtlichere und schnellere Ansatz besitzt von der Syntax her gewisse Ähnlichkeit mit der Programmiersprache Java.

Mitte der neunziger Jahre, wohl zeitgleich mit der wachsenden Popularität der Programmiersprache Java, wurde objektorientierte Programmierung (OOP) zu einem Modewort in der IT-Branche. Die Lösung nahezu aller Programmierprobleme wurde in der Objektorientierung gesehen.

Heute wie damals gibt es Kritiker an dem OOP-Ansatz. Einer der Punkte, die an OOP kritisiert werden ist, dass die Theorie sehr gut klingt, die praktische Umsetzung aber oft zu kompliziert und zu umständlich ist.

Was ist OOP?

Vereinfacht gesagt dreht sich bei **OOP** alles um Objekte. Ein Objekt ist dabei als Ansammlung von Daten und Funktionen zu verstehen. Darin besteht ein wesentlicher Unterschied zur prozeduralen Programmierung, wo Daten und Funktionen getrennt verwaltet werden.

> Nehmen wir einmal an, Sie schreiben ein Programm, das mit geometrischen Formen arbeitet:

Da Sie das Rechteck oft verwenden, programmieren Sie eine Klasse vom Typ Rechteck (die Klasse ist der PHP-Code des fertigen Objekts). Ihre Rechteck-Klasse hat die Eigenschaften Breite und Höhe (die Daten Ihres Objekts). Für geometrische Berechnungen benötigen Sie immer wieder die Fläche und den Umfang.

Ihrer Rechtecke. Die Klasse bekommt zwei Funktionen, `getArea` und `getPerimeter`. Ihre Klasse besteht nun aus zwei Variablen, die die Daten zu Ihrem Rechteck halten, und zwei Funktionen, die aus diesen Daten neue Werte errechnen. Jedes neue Rechteck-Objekt, das Sie erzeugen, kann von sich aus die Fläche und den Umfang zurückgeben. Ein zugegebenermaßen stark vereinfachtes Beispiel, aber es vermittelt die Grundidee von der objektorientierten Struktur.

OOP ist zu komplex, um es hier in zwei Absätzen zu erklären. Wenn Sie über die spezifische **PHP-Syntax** hinaus eine ernsthafte Einführung in OOP suchen, empfehlen wir einschlägige Bücher aus der OOP-Abteilung Ihres Buchgeschäfts.

Wer braucht OOP?

Die Vorteile von OOP liegen im Bereich von großen Anwendungen, also für Projekte, bei denen Sie von vornherein wissen, dass eine große Codebasis zustande kommen wird. Meist sind an solchen Projekten auch mehrere Entwickler beteiligt. OOP sieht fest definierte Schnittstellen der einzelnen Objekte vor, was in einer verteilten Entwicklung unerlässlich ist.

Mit OOP wird Ihr Code nicht kürzer oder weniger komplex. Oft ist sogar das Gegenteil der Fall. Verbessern sollte sich aber die Wartbarkeit und die Erweiterbarkeit des Codes.

> Speziell der Umstieg vom prozeduralen Programmieren zu OOP fällt vielen Entwicklern schwer,

da sich das Konzept stark unterscheidet. Wenn Sie noch nie mit OOP zu tun hatten, werden Sie in diesem Kapitel vermutlich nicht alles auf Anhieb verstehen. Das macht aber nichts.

Das Gute an **PHP** ist, dass Sie den Einstieg in OOP ganz gemütlich angehen können. Starten Sie damit, in Ihrem Programm ein paar zusammengehörende Funktionen in einer Klasse zu kapseln. Vielleicht merken Sie später, dass diese Funktionen immer mit gewissen Daten rechnen; dann lagern Sie auch die Variablen für diese Daten in den Code der Klasse aus.

Der Spagat von PHP: OOP und prozedural

PHP ist vor allem dafür bekannt, dass alles schnell geht – sowohl die Programmausführung als auch die Anwendungsentwicklung. Schnelle Anwendungsentwicklung bedeutet oftmals weniger Struktur und kurze

unzusammenhängende Scripts. Das muss nicht immer schlecht sein. Ein kleines **PHP-Script**, das seinen Zweck erfüllt, ist oft die ideale Ergänzung.

Die objektorientierte Programmierung hingegen zeichnet sich durch klare Strukturen, Schnittstellen und ein durchdachtes Konzept aus. Wie können diese unterschiedlichen Ansätze zusammenpassen? Anders als in Java wird man in PHP nicht zur Objektorientierung gezwungen. Sie können PHP 5 wie ältere PHP-Versionen verwenden und prozedurale Anwendungen schreiben. Noch besser ist: In PHP können Sie die beiden Ansätze vermischen.

Für PHP 5 wurde der schon in PHP 4 vorhandene objektorientierte Ansatz von Grund auf neu geschrieben. Während in älteren PHP-Versionen Objekte wie primitive Typen (z.B. Integer, String) verwendet wurden, hat die Zend Engine II, die in **PHP 5** zum Einsatz kommt, eine wesentlich ausgefeiltere Implementierung. Der neue Ansatz erlaubt außer einer fein graduierten Rechteverteilung innerhalb der Objekte auch eine deutlich gesteigerte Performance.

PHP basiert letztlich auch auf C und C ist die Sprache der Funktionen. Das klassische C-Ideal besteht in einer Programmierung, die vorwiegend aus Funktionen besteht, von denen jede eine bestimmte, generische Aufgabe erfüllt und daher wiederverwendbar ist. Funktionen erwarten meistens einen oder mehrere Parameter als `Input` und liefern einen Return-Wert (Rückgabewert) als `Output`. Im Prinzip also soll eine typische Funktion ein kleines EVA-Programm innerhalb des Gesamtprogramms sein (EVA = Eingabe, Verarbeitung, Ausgabe – das Paradigma der klassischen Datenverarbeitung aus den Zeiten bildschirmloser, magnetbandgefütterter Rechner).

> Nun stellt PHP schon sehr viele Funktionen bereit.

Dennoch bieten sich bei der Programmierung genügend Gelegenheiten, eigene Funktionen zu definieren. Es genügt schon, wenn ein Set aus Anweisungen an mehreren Stellen im Script-Code wiederholt werden muss. In diesem Fall bietet es sich an, das Anweisungs-Set in eine Funktion zu schreiben und diese bei Bedarf aufzurufen. Die Funktion hat dann die Aufgabe eines Unterprogramms, einer Sub-Routine.

Schema

Eine typische Funktion hat folgenden Aufbau:

```
function Funktionsname($Parameter) {
# Anweisungen;
return($Wert)
}
```

Eingeleitet wird eine Funktionsdefinition durch das Schlüsselwort `function`. Dahinter folgt, durch

Leerzeichen getrennt, ein frei wählbarer Funktionsname. Der Name kann wie ein Variablenname aus Buchstaben, Ziffern und Unterstrich bestehen, wobei der Name mit einem Buchstaben oder Unterstrich beginnen muss. Groß- und Kleinschreibung von Buchstaben werden nicht unterschieden, d.h., es ist egal, ob Sie eine Funktion mit `test()` oder `TEST()` aufrufen. Funktionsnamen müssen script-weit eindeutig sein. Wenn Sie beispielsweise andere Scripts über `include_once()` oder ähnliche Funktionen einbinden, dann dürfen in den anderen Scripts keine gleichnamigen Funktionen vorkommen. Nur Methoden verschiedener Objekte dürfen gleichnamig sein.

> Der gesamte Funktionskörper muss in geschweifte Klammern eingeschlossen werden,

auch dann, wenn die Funktion aus nur einer Anweisung besteht.

Falls die Funktion an die aufrufende Anweisung einen Wert zurückgeben will, kann sie dazu die PHP-Funktion `return()` verwenden. Als Parameter wird dieser Funktion übergeben, was zurückgegeben werden soll. Meistens ist das eine Variable, in der ein ermittelter Wert steht. Es kann jedoch auch eine literale Zeichenkette oder eine aus literalen Teilen und Variablen zusammengesetzte Zeichenkette sein. Wenn mehrere Werte zurückgegeben werden sollen, können diese in einem Array gesammelt werden. Zurückgegeben wird dann die Array-Variable. Durch `return()` wird die Funktion sofort verlassen.

Parameter und lokale Variablen

Variablen, die innerhalb eines Funktionskörpers definiert werden, gelten nur innerhalb dieser Funktion und sind außerhalb davon nicht bekannt. Innerhalb einer Funktion dürfen also durchaus Variablennamen vorkommen, die auch außerhalb davon vorkommen. Die Variablen außerhalb der Funktion werden durch diejenigen innerhalb der Funktion nicht berührt.

```
$i = 0;
function set_i() {
$i = 23; }
set_i();
echo $i;
```

Das Beispiel gibt 0 aus, weil die Funktion lost `set_i()`, die in ihrem Funktionskörper einer Variablen `$i` einen Wert zuweist, keinen Einfluss auf die Variable `$i` hat, die zuvor global mit 0 initialisiert wurde.

Auch Parameternamen, die bei einer Funktion definiert werden, sind lokale Variablen der Funktion.

Funktionsaufruf und Rückgabewert

Nachfolgendes Beispiel zeigt einen Komplettzusammenhang zwischen Funktionsaufruf, Funktion, Parametern und Rückgabewert (Return-Wert).

```
$mylink = set_link("http://www.example.org/", "Beispiel-Link");
echo $mylink;
function set_link($uri, $text) {
$str = "<a href=\"$uri\">$text</a>";
return($str);
}
```

Das Beispiel zeigt zunächst, dass ein Funktionsaufruf durchaus vor der Funktion selbst notiert werden darf. Der Grund dafür ist, dass PHP ein Script vor der Ausführung ohnehin erst einmal `on the fly` kompiliert. Dabei werden intern auch alle Funktionen und Variablen registriert.

Im Beispiel wird eine Funktion namens `set_link()` aufgerufen. Ihr werden in den runden Klammern hinter dem Funktionsnamen zwei Zeichenkettenparameter übergeben. Mehrere Parameter werden durch Kommata getrennt. Auch wenn keine Parameter übergeben werden, müssen die runden Klammern notiert werden.

Da die Funktion `set_link()` einen Rückgabewert erzeugt, kann sie wie im Beispiel einer Variablen zugewiesen werden. In der Variablen `$mylink` steht im Beispiel am Ende das, was die Funktion `set_link()` zurückgibt. Funktionen, die einen Wert zurückgeben, können aber auch direkt in Prozessanweisungen oder sogar als Parameter an andere Funktionen übergeben werden. So wäre es im obigen Beispiel sogar kürzer, gleich zu notieren:

```
echo set_link("http://www.html-info.eu/", "Beispiel-Link");
```

Angenommen, es gibt eine zweite Funktion namens `set_link_style()`, die als ersten Parameter einen kompletten HTML-Hyperlink erwartet und als zweiten Parameter den Namen einer CSS-Klasse, die dem Link zugewiesen werden soll. Dann könnten Sie im Beispiel notieren:

```
$mylink = set_link_style(set_link("http://www.html-info.eu/", "Beispiel-Link"),
"extern_link");
```

Als erster Parameter wird beim Aufruf von `set_link_style()` ein Aufruf von `set_link()` übergeben. Der PHP-Interpreter weiß, dass er in diesem Fall zuerst `set_link()` ausführen muss und den davon zurückgegebenen Wert in den Aufruf von `set_link_style()` einsetzen muss.

Default-Werte bei Parametern

Eine PHP-spezifische Besonderheit muss erwähnt werden. Bei Parameterdefinitionen im Funktionskopf können auch Defaultwerte angegeben werden.

```
function line_numbered_file($file, $color="red") {
```

```
$lines = file($file);
$chars = strlen((string) count($lines));
for($i = 1; $i <= count($lines); $i++) {
$line = str_replace("%", "%%", $lines[$i]);
$format = "<span style=\"color:".
$color."\">%".$chars."d</span> ".$line;
$numbered_lines[] = sprintf($format, $i);
}
return($numbered_lines);
}
$file_lines = line_numbered_file("/etc/php.ini");
echo "<pre>";
foreach($file_lines as $line)
echo $line;
echo "</pre>";
```

Die im Beispiel definierte Funktion `line_numbered_file()` versieht die Zeilen einer Textdatei mit führenden Zeilennummern. Dabei werden die Zeilennummern in HTML andersfarbig formatiert. Welche Textfarbe die Zeilennummern erhalten sollen, wird mit dem zweiten Parameter `$color` übergeben. Sollte dieser Parameter beim Funktionsaufruf fehlen, verwendet PHP den Default-Wert `red`. Grund dafür ist die Notation `$color="red"` in der Parameterdefinitionsliste in den runden Klammern der Funktion.

> Normalerweise gibt PHP eine Warnung aus,

wenn ein Funktionsaufruf eine Funktion nicht mit all den Parametern versorgt, die sie erwartet. Wenn jedoch bei der Funktionsdefinition mit der beschriebenen Syntax Defaultwerte für Parameter definiert werden, dann dürfen diese Parameter beim Aufruf auch weggelassen werden.

Auf den Inhalt der Funktion `line_numbered_file()` gehen wir an dieser Stelle nicht näher ein, weil uns das zu weit vom Thema abbringen würde.

Variable Anzahl von Parametern

Eine weitere Möglichkeit besteht darin, bei der Funktionsdefinition überhaupt keine Parameter zu definieren. Dennoch können der Funktion beim Aufruf Parameter übergeben werden, und zwar beliebig viele. PHP stellt einige Funktionen bereit, mit deren Hilfe eine Funktion auf ihr übergebene Parameter zugreifen kann.

```
$a = 17;
$b = 64;
$c = 23;
function sum() {
```

```
$x = 0;
$params = func_get_args();
foreach($params as $param)
$x += $param;
return($x);
}
echo sum($a, $b, $c);
```

Die Funktion `sum()` im Beispiel ist in der Lage, die Summe einer beliebigen Anzahl übergebener Zahlenwerte zu ermitteln und zurückzugeben. Dabei erwartet sie gar keine Parameter. Falls sie ohne Parameter aufgerufen würde, würde sie einfach 0 zurückgeben.

Mit der PHP-Funktion `func_get_args()` kann eine Funktion alle ihr übergebenen Parameter als Array ermitteln. In unserer Beispielfunktion wird der Array in `$params` gespeichert. In einer `foreach`-Schleife arbeitet die Funktion den Array `$params` ab und erhöht `$x` um den jeweiligen Wert, in der Annahme, dass lauter numerische Werte übergeben wurden.

Die `echo`-Ausgabe am Ende des Beispiels gibt 104 aus, weil der Funktion `sum()` beim Aufruf die drei zu Beginn initialisierten Variablen `$a`, `$b` und `$c` übergeben werden.

Zeiger (Referenzen) statt Werte übergeben

In Script-Sprachen wie PHP wird nicht so ausufernd mit so genannten Zeigern (Pointern) gearbeitet wie etwa in C. Dennoch bietet auch PHP die Möglichkeit an, einer Funktion anstelle eines Werts nur die Speicheradresse zu übergeben, an der der Wert beginnt. Sinnvoll ist dies beispielsweise, wenn als Parameter eine 1 Megabyte große Zeichenkette übergeben wird. Anstelle eines Megabyte werden nur ein paar Byte übergeben, eben die Arbeitsspeicheradresse der Variablen mit dem großen Inhalt. Damit PHP weiß, dass ein übergebener Wert nicht wirklich ein Wert ist, sondern nur eine Arbeitsspeicheradresse, muss er kenntlich gemacht werden.

```
$satz = "Ein langer Satz mit vielen Wörtern";
function count_words(&$str) {
return(count(explode(" ",$str)));
}
echo count_words($satz);
```

Die Zeigertechnik wird bei der Parameterdefinition der Funktionsdefinition angewendet. Im Beispiel bewirkt `&$str`, dass die Funktion beim Aufruf keine Kopie des übergebenen Werts erhält, sondern dessen Arbeitsspeicheradresse. Das kaufmännische Und (&) vor dem Parameternamen genügt, um den Parameter als Referenz bzw. Zeiger zu deklarieren.

Beim Aufruf der Funktion muss (im Gegensatz etwa zu C) nichts beachtet werden. Im Beispiel wird der Funktion einfach ein Variablenname übergeben.

Wichtig zu wissen ist jedoch, dass Funktionen, die eine als Referenz übergebene Variable ändern, nicht die Kopie ändern, sondern das Original. In manchen Fällen kann dies sogar explizit gewünscht sein, in anderen dagegen nicht.

In PHP 4.x ist die Möglichkeit objektorientierter Programmierung zwar im Prinzip angelegt, doch fehlen noch entscheidende Features wie Regelungen zur Sichtbarkeit von Variablen. Dennoch lohnt es sich bei größeren Projekten, gleich mit objektorientierter Programmierung zu beginnen.

Grundbegriffe

Eine **Klasse** ist in der objektorientierten Programmierung das, was in der Textverarbeitung eine Dokumentvorlage ist. So, wie in einer Dokumentvorlage Seitenformate, Absatz- und Zeichenformate sowie weitere Vorgaben definiert werden, werden in einer Klasse Datenstrukturen und Funktionen definiert.

Ein **Objekt** ist dann ein konkretes Dokument. So, wie ein Dokument aus einer Dokumentvorlage erstellt werden kann, kann ein Objekt aus einer Klasse erzeugt werden. Andere Bezeichnungen für „Objekt“ sind **Instanz** und **Exemplar**.

Eine **Objekteigenschaft** oder einfach **Eigenschaft** ist eine Variable, die in der Klasse definiert wurde, aus der das Objekt erzeugt wurde.

Eine **Objektmethode** oder einfach **Methode** ist eine Funktion, die in der Klasse definiert wurde, aus der das Objekt erzeugt wurde.

Klassen und Objekte anlegen

Beginnen wir mit einem ganz einfachen Beispiel, um das Prinzip zu demonstrieren:

```
class MiniClass {
var $test = 23;
}
$mini_object = new MiniClass;
echo $mini_object->test;
```

In diesem Beispiel wird eine Klasse mit dem Namen MiniClass definiert. Eine Klassendefinition beginnt mit dem Schlüsselwort `class`. Dahinter folgt, durch Leerzeichen getrennt, der gewünschte Klassenname. Bei der Wahl des Klassennamens sind Sie frei. Klassennamen müssen jedoch innerhalb eines gesamten Script-Kompilats eindeutig sein. Ansonsten gelten die gleichen Regeln wie bei Variablen und Funktionsnamen. Es hat sich jedoch eingebürgert, Klassennamen mit Großbuchstaben beginnen zu lassen.

Der gesamte Inhalt einer Klasse muss in geschweifte Klammern eingeschlossen werden,

genauso wie bei einer Funktion. Innerhalb einer Klasse darf nichts anderes notiert sein als Variablendefinitionen oder Funktionen. In unserem Minimalbeispiel ist lediglich eine einzige Variablendefinition notiert. Variablendefinitionen innerhalb von Klassen werden durch das Schlüsselwort var eingeleitet. Ansonsten gelten die gleichen Regeln wie bei normalen Variablen. Variablen können mit einem Wert initialisiert werden wie im Beispiel. Zwingend erforderlich ist das jedoch nicht. Wenn eine Variable zunächst keinen Wert haben soll, notieren Sie einfach var `$name;` – also direkt hinter dem Variablennamen das Semikolon zum Abschließen der Anweisung.

Wenn Sie Klassenvariablen mit einem Wert initialisieren wollen wie im Beispiel, darf dies nur ein konstanter Wert sein, also etwa eine Zahl oder eine Zeichenkette, nicht aber ein ermittelter Wert, wie er z.B. durch Aufruf von PHP-Funktionen oder anderen, eigenen Funktionen zustande käme. Falls Sie den Initialisierungswert der Variablen doch ermitteln lassen wollen, benötigen Sie eine Konstruktorfunktion. Dazu weiter unten mehr.

In unserem Minimalbeispiel wird unterhalb der Klasse eine Variable namens `$mini_object` definiert. Als Wert wird ihr `new MiniClass` zugewiesen. Mit dem Schlüsselwort new signalisieren Sie, dass Sie aus dieser Variable eine Objektvariable machen möchten. Dahinter geben Sie den Klassennamen an, aus dem ein Objekt erzeugt werden soll. In unserem Beispiel wird `$mini_object` zu einer Objektvariable für die Klasse MiniClass.

Die Variablendefinition

`$mini_object = new MiniClass` ist also nichts anderes als das Erzeugen eines Objekts aus der Klasse MiniClass. Das Objekt ist in der Variablen `$mini_object` gespeichert. Über diese Objektvariable ist der Zugriff auf die Eigenschaften (Variablen) und Methoden (Funktionen) der Klasse möglich.

Der Zugriff erfolgt über den „Zeige-Operator“ lost `->` (Minuszeichen und Größer-als-Zeichen direkt hintereinander notiert). In unserem Minibeispiel wurde in der Klasse MiniClass eine Variable `$test` definiert. Aus Sicht der Objektvariablen `$mini_object` ist das eine Objekteigenschaft. Durch die Syntax `$mini_object->test` können Sie auf den Wert der Eigenschaft zugreifen. Beachten Sie dabei, dass vor dem Namen der Eigenschaft, hier also test, kein $-Zeichen notiert werden darf!

Das Beispiel gibt 23 aus, weil es einfach `$mini_object->test` ausgibt und darin die in der Klassendefinition festgelegte Vorbelegung gespeichert ist. Die Objekteigenschaft kann jedoch wie jede andere Variable auch geändert werden.

Klassen mit Konstruktortfunktion

Eine Klasse darf wie schon erwähnt nichts anderes enthalten als Variablendefinitionen mit var und Funktionen. Wenn innerhalb der Klasse eine Funktion definiert wird, die exakt den gleichen Namen hat wie die Klasse selbst, dann gilt diese Funktion als so genannte **Konstruktorfunktion**. Das Besondere an der Konstruktorfunktion ist, dass sie beim Anlegen eines neuen Objekts mit `$variable = new $Klassenname` automatisch mit ausgeführt wird.

```
class MiniClass {
var $test;
function MiniClass() {
$this->test = ini_get("max_execution_time");
} }
$mini_object = new MiniClass;
echo $mini_object->test;
```

Die Klasse MiniClass enthält nach wie vor eine Variablendefinition, wobei die Variable `$test` jedoch jetzt nicht mehr initialisiert wird. Sie soll zwar initialisiert werden, aber mit einem ermittelten Wert, was bei der Definition mit var innerhalb der Klasse nicht erlaubt ist. Die Initialisierung holen wir deshalb in einer Konstruktorfunktion nach. Diese ist im Beispiel die Funktion `MiniClass()`, weil sie exakt genauso heißt wie die Klasse, innerhalb deren sie definiert wird.

> Innerhalb der Konstruktorfunktion wird die Variable

`$test` initialisiert, und zwar zum Spaß mit einem Aufruf der PHP-Funktion `ini_set()`. Diese holt Werte aus der **php.ini**. In unserem Fall ermittelt sie, wie lange das Script maximal laufen darf, bevor es gewaltsam beendet wird.

Wichtig zu kennen ist die Syntax: Wenn innerhalb einer Klasse auf Klassenvariablen zugegriffen werden soll, also auf Variablen, die außerhalb von Funktionen mit var definiert wurden, dann muss das mit `$this->variablenname` geschehen.

Das Beispiel-Script gibt am Ende den Wert aus der **php.ini** aus, der dort bei der Direktive **max_execution_time** zugewiesen ist.

Konstruktorfunktion mit Parametern

Eine Konstruktorfunktion, so haben wir schon gelernt, ist nicht zwingend erforderlich. Sie ist beispielsweise dann nötig, wenn Initialisierungswerte von Klassenvariablen durch Funktionsaufrufe oder Ähnliches ermittelt werden sollen. Ein anderer wichtiger Grund, eine Konstruktorfunktion zu verwenden, wäre der Wunsch, Parameter an eine Klasse zu übergeben.

```
class RGBColor {
var $R = 0;
var $G = 0;
var $B = 0;
function RGBColor($r, $g, $b) {
if($r >= 0 and $r <= 255)
$this->R = $r;
```

```
if($g >= 0 and $g <= 255)
$this->G = $g;
if($b >= 0 and $b <= 255)
$this->B = $b;
}
}
$my_blue = new RGBColor(40,80,220);
echo $my_blue->B;
```

In diesem Beispiel wird eine Klasse namens RGBColor definiert, in der ein RGB-Wert gespeichert werden kann. Sie enthält neben den drei Klassenvariablen $R, $G und $B für jeden Farbanteil eine gleichnamige Funktion RGBColor, die dadurch ihre Konstruktorfunktion ist. Diese Funktion erwartet drei Parameter (für die drei Farbanteile).

Beim Erzeugen des Objekts wird die Klasse dann wie eine Funktion aufgerufen. Nur durch das Schlüsselwort new davor ist noch erkennbar, dass eine Klasse und keine Funktion aufgerufen wird.

Die Konstruktorfunktion bekommt die beim Klassenaufruf übergebenen Parameter übergeben. In unserem Beispiel prüft sie, ob sich die Werte im erlaubten Bereich zwischen 0 und 255 befinden. Wenn ja, weist sie den Klassenvariablen die übergebenen Werte zu.

Abgeleitete und erweiternde Klassen

Angenommen, Sie haben folgende Klasse:

```
class Person {
var $given_name;
var $family_name;
var $birthday_dd;
var $birthday_mm;
var $birthday_yyyy;
}
```

Sie haben möchten nun außerdem eine Klasse für Kontaktdaten anlegen. Dann bietet Ihnen das Konzept der objektorientierten Programmierung an, die existierende Klasse Person zu kopieren und zu erweitern.

```
class Contact extends Person {
var $street;
var $zip_code;
var $location;
var $phone_number;
var $mobile_number;
```

```
var $mail_address;
}
```

Bei der Definition einer Klasse können Sie hinter dem Klassennamen das Schlüsselwort extends und dahinter den Namen einer anderen existierenden Klasse angeben. Dadurch binden Sie alle dort definierten Klassenvariablen (Eigenschaften) und Funktionen (Methoden) in die neu anzulegende Klasse mit ein. Die neue Klasse kann dann zusätzliche Variablen und Funktionen definieren.

Im Beispiel wird eine Klasse `Contact` definiert, welche die Klasse Person erweitert. Innerhalb von `Contact` kann nun z.B. innerhalb einer Methode auf `$this->given_name` zugegriffen werden, da alle Definitionen aus `$Person` in `$Contact` übernommen wurden. Ebenso verhält es sich bei Objekten, die aus `Contact` erzeugt werden.

```
$julia = new Contact;
$julia->given_name = "Julia";
echo $julia->given_name;
```

Wichtig zu wissen ist noch, wie es sich bei abgeleiteten Klassen mit Konstruktorfunktionen verhält. Hat die abgeleitete Klasse eine eigene Konstruktorfunktion, so wird diese beim Erzeugen eines Objekts aufgerufen, die der Elternklasse dagegen nicht. Gibt es in der abgeleiteten Klasse dagegen keine Konstruktorfunktion, so wird, falls die Elternklasse eine Konstruktorfunktion besitzt, diese aufgerufen. Vor allem Letzteres müssen Sie als Programmierer wissen, denn wenn die Konstruktorfunktion der Elternklasse Parameter erwartet, müssen Sie die abgeleitete Klasse beim Erzeugen eines Objekts mit den Parametern aufrufen, die von der Konstruktorfunktion der Elternklasse erwartet werden.

Direktaufruf von Methoden

Mithilfe des `Operators ::` ist es möglich, Methoden aus Klassen direkt aufzurufen, ohne ein Objekt der Klasse zu erzeugen.

```
class Member {
var $name = "Max Mustermann";
function show_name() {
echo "bla"; }
}
Member::show_name();
```

Mit der Syntax `Klassenname::Methodenname()` kann eine Methode aus einer Klasse direkt ausgeführt werden. Allerdings mit einer Einschränkung: Die Methode darf nicht mit `$this->Variablenname` auf Klassenvariablen Bezug nehmen, da die Klassenvariablen ohne Objektinstanz nicht existieren. Das obige Beispiel gibt `bla` aus. Würde in `show_name()` jedoch stehen: `echo $this->name`, dann würde beim Aufruf über `Member::show_name()` gar nichts ausgegeben, da in diesem Aufrufkontext keine

Klassenvariable $name existiert.

Praxisbeispiel: Template-Klasse

In einem kleinen Praxisbeispiel greifen wir auf das Beispiel aus Codeerstellung und Codeverteilung zurück. Dort haben wir bereits gesehen, wie sich PHP-Code bei größeren Projekten sinnvoll auf mehrere Scripts verteilen lässt. Dies wollen wir nun auch in Zusammenhang mit Klassen und Objekten demonstrieren.

Das Beispiel besteht aus folgenden PHP-Scripts:

- **linkweb.php**: das zentrale Script, das auch im Browser aufgerufen wird
- **class.template.php**: Script mit der Klasse „Template“ für HTML-Templates
- **class.page.php**: Script mit Klasse „Page“ zum Zusammenkleben einer Seite aus Template und Inhalt
- **class.error.php**: Script mit Klasse „Error“ zur Ausgabe von Fehlermeldungen

Die übrigen Dateien für Templates, Inhaltsdateien und CSS-Datei bleiben wie gehabt.

Zunächst der Quelltext des zentralen Scripts:

```
<?php
include_once("class.template.php");
include_once("class.page.php");
$main_template = new Template;
$main_page = new Page;
$content_files = array();
$content_files['home'] = "0001.txt";
$content_files['impressum'] = "0007.txt";
$content_files['themen'] = "0003.txt";
$content_files['branchen'] = "0009.txt";
$content_files['auskunft'] = "0005.txt";
$content_files['geo'] = "0002.txt";
$content_files['wissen'] = "0004.txt";
$content_files['literatur'] = "0008.txt";
$content_files['lexika'] = "0006.txt";
$content_files['glossare'] = "0010.txt";
$content_files['wikis'] = "0014.txt";
$content_files['ftp'] = "0011.txt";
$content_files['wais'] = "0015.txt";
$content_files['newsgroups'] = "0012.txt";
$content_files['foren'] = "0013.txt";
if(isset($_GET['page']))
$get_page = $_GET['page'];
else
$get_page = 'home';
```

```
$main_template->template_file = "linkweb.tpl";
$main_page->page_content = $main_template->get_template();
$main_page->set_var('content',
file_get_contents($content_files[$get_page]));
$main_page->set_var('title', $main_page->get_title());
$main_page->show_page();
exit();
?>
```

Dieses Script enthält selber keine Klasse, sondern regelt nur den Gesamtablauf. Die Definition des Arrays `$content_files` haben wir der Einfachheit halber mit in das zentrale Script übernommen – bei größeren Projekten empfiehlt sich aber, wie früher erwähnt, ein eigenes PHP-Script für globale Variablendefinitionen.

> Die eigentliche Arbeit des Gesamt-Scripts wird jedoch von Script-Dateien erledigt,

die im Wesentlichen aus Klassen bestehen. Im Zentralscript werden oben die Scripts **class.template.php** und **class.page.php** eingebunden. Die Dateinamen sind natürlich frei wählbar. Es ist jedoch guter Stil, PHP-Scripts, die eigentlich nur eine Klasse enthalten, im Dateinamen entsprechend zu bezeichnen.

Unterhalb der `include`-Anweisungen sind denn auch gleich die Definitionen der benötigten Objektvariablen notiert. In `$main_template` wird ein Objekt der Klasse Template gespeichert und in `$main_page` ein Objekt der Klasse Page. Der Script-Ablauf im Zentral-Script besteht wie schon im früheren Beispiel darin, zunächst den übergebenen GET-Parameter page abzufragen, um zu ermitteln, welche Seite angezeigt werden soll.

Dann wird mit den beiden erzeugten Objekten gearbeitet. Zunächst wird der Eigenschaft `template_file` des Template-Objekts die gewünschte Template-Datei zugewiesen. Über die Methode `get_template()` wird die Template-Datei eingelesen. Der Rückgabewert der Methode ist die eingelesene Datei. Dieser wird gleich der Eigenschaft `page_content` des Page-Objekts zugewiesen. So lesen wir in einer einzigen Anweisung das HTML-Template ein und kopieren es in den zu erstellenden Seiteninhalt.

> Nun müssen die aus dem Template kopierten Platzhalter des Typs

`[%name%]` durch seitenabhängige Inhalte ersetzt werden. Dies besorgt die Methode `set_var()` des Page-Objekts. Ihr werden zwei Parameter übergeben: der Platzhaltername, bei `[%name%]` also `'name'`, und der gewünschte Wert, durch den der Platzhalter ersetzt werden soll. Das Zentral-Script muss die Platzhalter `[%content%]` und `[%title%]` ersetzen. Bei `[%content%]` setzt es seitenabhängig die passende Datei aus dem Array `$content_files` ein und bei `[%title%]` kann es den Inhalt durch Aufruf der Methode `get_title` des Page-Objekts ermitteln. Die Methode `get_title` sucht im bislang vorhandenen HTML-Code der Seite nach einer `h1`-Überschrift, ermittelt deren Inhalt und gibt ihn zurück.

Am Ende muss das Zentral-Script nur noch die Methode `show_page()` des Page-Objekts aufrufen, um seine Arbeit abzuschließen und die gewünschte Seite auszugeben.

Das gesamte Script-Ensemble wird natürlich erst verständlich durch die Quelltexte der Dateien mit den Klassen. Zunächst die Template-Klasse:

```
<?php
#----------------------------------------
# Includes:
include_once("class.error.php");
$template_error = new Error;
#----------------------------------------
# Klasse Template:
class Template {
var $template_file;
var $template_content;
function get_template() {
global $template_error;
if(empty($this->template_file))
$template_error->handle_error
('Keine Template-Datei angegegeben!');
if(! file_exists($this->template_file))
$template_error->handle_error
('Template-Datei nicht gefunden!');
$this->template_content =
file_get_contents($this->template_file);
return($this->template_content);
} }
?>
```

Die Script-Datei zeigt den typischen Aufbau einer Script-Datei, die lediglich eine Klasse enthält. Im globalen Bereich vor der Klassendefinition werden andere Scripts eingebunden, die mehrfach benötigt werden. In unserem Fall wird **class.error.php** benötigt. Da dies auch eine Klasse ist, muss natürlich auch ein Objekt daraus erzeugt werden, das in der Objektvariablen `$template_error` gespeichert wird.

Umgekehrt werden wir in **class.error.php** sehen, dass dort ein Template zur Ausgabe der Fehlerseite benötigt wird und dass dazu ein Template-Objekt in der Variablen `$error_template` erzeugt wird. Eine solche Stringenz sowohl bei Dateinamen als auch bei Variablennamen ist bei größeren Projekten dringend zu empfehlen, da ansonsten leicht Probleme mit doppelt vorkommenden Variablen vorkommen können und die Gesamtübersicht einfach leichter verloren geht.

Die Klasse Template wird durch class Template { … } definiert. Der Inhalt besteht aus zwei Klassenvariablen (Objekteigenschaften) namens `$template_file` und `$template_content`. In der einen muss ein Template-Objekt den Pfadnamen der gewünschten Template-Datei ablegen und in der

anderen wird nach Aufruf der einzigen Klassenfunktion (Objektmethode) `get_template()` der Inhalt der Template-Datei gespeichert.

Eine Konstruktorfunktion wird in dieser Klasse nicht benötigt. Als Nächstes der Quelltext der Error-Klasse:

```
<?php
#----------------------------------------
# Includes:
include_once("class.template.php");
include_once("class.page.php");
$error_template = new Template;
$error_page = new Page;
#----------------------------------------
# Klasse Error:
class Error {
var $error_template = "error.tpl";
function handle_error($message) {
global $error_template;
$error_template->template_file = $this->error_template;
$error_page->page_content =
$error_template->get_template();
$error_page->set_var('message', $message);
$error_page->show_page();
exit();
} }
?>
```

Das Script mit der Error-Klasse benötigt die Klassen Template und Page, bindet dazu die nötigen Script-Dateien ein und erzeugt die erforderlichen Objekte in den Objektvariablen `$error_template` und `$error_page`.

Die Error-Klasse selbst besteht aus einer Klassenvariable für den Pfadnamen der Error-Template-Datei sowie aus der Funktion `handle_error()`. Deren Aufgabe ist eine ähnliche wie die des Zentral-Scripts. Sie muss die Error-Template-Datei einlesen und in die Objekteigenschaft `page_content` des erzeugten Page-Objekts kopieren. Dann muss sie noch mit der `set_var`-Methode des Page-Objekts den Platzhalter `[%message%]` durch die Fehlermeldung ersetzen, die beim Aufruf von `handle_error()` übergeben werden muss, und zuletzt kann sie die Fehlerseite ausgeben und den Scriptlauf beenden.

Nun noch der Quelltext der Page-Klasse:

```
<?php
#----------------------------------------
# Klasse Page:
```

```
class Page {
var $page_content;
var $standard_title = "untitled";
function set_var($name, $value) {
if(empty($this->page_content))
return;
else {
$this->page_content =
str_replace("[%$name%]",
$value, $this->page_content);
return($this->page_content);
} }
function show_page() {
if(empty($this->page_content))
return; else
echo $this->page_content;
}
function get_title() {
if(empty($this->page_content))
return;
else {
$pattern = "/<h1>(.*)<\/h1>/";
preg_match($pattern, $this->page_content, $matches);
if(isset($matches[1]))
return($matches[1]);
else
return($this->standard_title);
}
}
}
?>
```

Die Page-Klasse benötigt keine anderen Scripts. Die Klasse selbst besteht aus den Klassenvariablen `$page_content` und `$standard_title`. In `$page_content` wird der komplette HTML-Code einer auszugebenden Seite gespeichert. In diese Variable muss zunächst der durch Einlesen des Templates gewonnene Inhalt kopiert werden, bevor die Methoden des Page-Objekts sinnvoll anwendbar sind. Die Initialisierung von `$standard_title` soll nur einen Defaultwert für den Titel vorgeben, falls die Funktion `get_title()` keine `h1`-Überschrift ermitteln kann, aus der sie den Titel gewinnen kann.

Die Funktionen der Page-Klasse sind bereits aus den Aufrufen als Objektmethoden bekannt: `set_var()` erlaubt es, Platzhalter des Schemas `[%name%]` durch Inhalte zu ersetzen. Die Methode aktualisiert dabei die Klassenvariable `$page_content`, gibt deren Wert aber zusätzlich auch an eine aufrufende Anweisung zurück. `show_page()` gibt eine fertig erzeugte Seite aus und `get_title()` sucht nach einer `h1`-Überschrift und gibt deren Inhalt zurück.

Ohne die Anwendung von Kontrollstrukturen kommen Sie beim Programmieren nicht aus. Hinter der Bezeichnung Kontrollstrukturen verbergen sich Bedingungen und Schleifen. Mit einer Bedingung ist gemeint, dass man **PHP** anweist, etwas zu vergleichen und dann in Abhängigkeit von dem Resultat des Vergleichs bzw. der Prüfung die eine oder andere Aktion durchzuführen. Erst mit dem Einsatz dieser Möglichkeit kreieren Sie wirklich dynamische Seiten, denn es erfolgen unterschiedliche Reaktionen je nach vorausgegangenen „Ereignissen“.

PHP kennt zwei Bedingungen: die `if`-Anweisung und die `switch`-Anweisung.

Die `if`-Anweisung

Diese Anweisung benutzen Sie, wenn Sie im Klartext sagen möchten: Wenn eine Bedingung, die mit Hilfe eines Ausdrucks formuliert wird, erfüllt ist, also `true` (wahr) ergibt, dann soll der angegebene Befehlsblock ausgeführt werden. Ist der Ausdruck `false` (falsch), wird dieser Programmblock ignoriert und die Programmausführung mit dem nachfolgenden Befehl fortgesetzt. Eine `if`-Anweisung wird folgendermaßen geschrieben:

```
if(Bedingung) {was soll geschehen;}
```

In der runden Klammer steht der logische Ausdruck, in der geschweiften Klammer, auch als Block bezeichnet, der Dann-Teil der Anweisung bzw. Anweisungen. Jede Anweisung muss mit Semikolon abgeschlossen werden, aber Sie können im Prinzip beliebig viele Anweisungen in den Dann-Teil schreiben. Am Ende folgt die geschlossene geschweifte Klammer. Das folgende Listing zeigt einen Ausschnitt eines Scripts mit einer `if`-Anweisung.

```
<?php
if($kontakt=="e-mail")
{
echo "geben Sie eine e-mail-Adresse ein";
}
?>
```

Hier wurde (z.B. durch eine Option eines Auswahlfeldes in einem Formular) eine Kontaktoption (e-mail) übergeben, die in der Variablen $kontakt gespeichert ist.

Grundsätzlich kann die `if`-Bedingung weiter verschachtelt werden, d.h. jeder Block kann erneut einen `if`-Befehl enthalten. Dies führt aber – wie Sie sich denken können – zu ziemlich unübersichtlichen Codes und ist deshalb nicht unbedingt empfehlenswert. In der Regel gibt es Alternativen.

Vergleichsoperatoren und logische Operatoren

Für den richtigen Gebrauch der `if`-Anweisung müssen Sie noch einiges wissen. Wenn Sie als Bedingung

einfach einen zuvor definierten Variablennamen verwenden, sagen Sie damit quasi so etwas wie: Wenn die Variable existiert und wenn sie einen Wert hat, der nicht Null ist, ergibt die Prüfung `true`. Anders bei einem Vergleich. Wenn Sie eine `if`-Anweisung dazu einsetzen, um zu prüfen, ob sie mit einem spezifischen Wert übereinstimmt, dann brauchen Sie einen Vergleichsoperator. In diesem Fall ist dies nicht einfach das Gleichheitszeichen (mit dem Sie einer Variablen einen Wert zuweisen), sondern es sind zwei Gleichheitszeichen hintereinander ==. Es ist wichtig, sich daran zu erinnern:

```
if($anrede == "Frau")
```

müsste es also heißen, wenn eine Anweisung ausgeführt werden soll, sofern als Anrede Frau übergeben und der Wert in der Variablen $anrede gespeichert ist. Ist dies der Fall, ergibt die Prüfung der Bedingung `true` (wahr).

Doppelachtung! Wenn Sie in einer `if`-Bedingung die doppelten Gleichheitszeichen vergessen, gibt PHP keine Fehlermeldung aus, sondern nimmt die mit dem einfachen Gleichheitszeichen angewiesene Zuweisung vor. Bei der Zuweisung ändert sich der Wert der Variablen.

Nach `if($anrede = "Frau")` steht in $anrede der Wert „Frau“.

Diese Fehler sind schwer zu finden und zeigen keine typischen Auswirkungen auf das **Script**, sodass man nicht sagen kann: Immer, wenn das und das passiert, haben Sie ein doppeltes Gleichheitszeichen in der `if`-Bedingung vergessen. Der einzige Hinweis, den wir Ihnen geben können, ist der folgende:

Gibt das Script keinen Fehler aus und scheint der Code eigentlich richtig zu sein, aber die Ausgabe des Scripts ergibt partout nicht das, was Sie sich gedacht haben, dann kontrollieren Sie die doppelten Gleichheitszeichen.

Andere Vergleichsoperatoren sind Ihnen wahrscheinlich bekannt. So können Sie in einer `if`-Anweisung die mathematischen Zeichen für Größer als und Kleiner als benutzen, also > und < bzw. >= und <=, beispielsweise so:

```
if($kosten < 50) {echo "Zahlen Sie bitte per Scheck";}
```

Einen Überblick über die Operatoren bietet die Tabelle.

Symbol	Bedeutung
==	Gleichheit
!=	Ungleichheit

Symbol	Bedeutung
>	Größer als
<	Kleiner als
<=	Kleiner als oder gleich
>=	Größer als oder gleich

Vermutlich kennen Sie noch aus der Schule auch die speziellen (logischen) Operatoren, mit denen Ausdrücke verknüpft werden. Dies sind:

- AND
- OR
- XOR

Sie können diese Operatoren bei `if`-Bedingungen einsetzen. Wenn Sie beispielsweise $variable1 AND $variable2 schreiben, bedeutet dies, dass nur wahr zurückgegeben wird, wenn beide Variablen wahr sind. Bei OR reicht es, wenn eine der Variablen wahr ist, um als Ergebnis wahr auszugeben. Mit XOR ergibt die Prüfung falsch, wenn beide Variablen gleich sind.

Verwendung von `if`-else

Neben der reinen `if`-Anweisung kommt sehr häufig die nächste logische Bedingung ins Spiel: die `if`-Anweisung, erweitert durch `else` (sonst). Mit der `else`-Klausel wird ein Code ausgeführt, wenn die Prüfung der Bedingung `false` ergeben hat. Die Syntax ist folgende:

```
if (Bedingung) {Anweisung1;}
else {Anweisung2;}
```

Das würde dann in einem Script beispielsweise so aussehen:

```
if($kosten < 50) {echo "Zahlen Sie bitte per Scheck";}
else {echo"geben Sie Ihr Konto an";}
```

Neben der `else`-Klausel kennt PHP auch die `elseif`-Klausel. Damit können mehrere Ausdrücke (Bedingungen) geprüft werden, indem in dem `else`-Zweig noch ein weiterer `if`-Befehl eingebaut wird. Sie verwenden die `if`-Anweisung mit `elseif`-Klausel folgendermaßen:

```
if (Bedingung1) {Anweisung1;}
elseif (Bedingung2) {Anweisung2;}
else {Anweisung3;}
```

Den `else`-Teil können Sie auch weglassen, wenn es keine Anweisung für den Fall gibt, dass die Prüfung der Bedingungen nicht `true` zurückgegeben hat.

- 1. Öffnen Sie gegebenenfalls den Editor und beginnen Sie ein neues Script mit `<?php`.
- 2. In der nächsten Zeile verwenden Sie die Funktion `date()`, die je nach Formatierungsanweisung, die in der Klammer steht, ein Datum in formatierter Form zurückgibt. W ist das Symbol für die numerische Darstellung des Wochentags. Schreiben Sie in die nächsten Zeilen:

```
$tag=date("W");
if($tag==0 OR $tag==7)
```

- 3. Dann beginnt der Anweisungsblock. Sie schreiben:

```
{echo "endlich Wochenende";
}
```

- 4. Nun beginnt die `elseif`-Klausel:

```
elseif ($tag==5)
{echo "fast geschafft";
}
```

- 5. Als Letztes schreiben Sie die `else`-Anweisung, danach schließen Sie den PHP-Teil.

```
else {echo "Arbeiten";
}
?>
```

- 6. Im Ganzen sieht das Script folgendermaßen aus:

```
<?php
$tag=date("w");
if($tag==0 OR $tag==7)
{echo "endlich Wochenende";
} elseif ($tag==5)
```

```
{echo "fast geschafft";
}
else {echo "Arbeiten!";
}
?>
```

Die switch-Anweisung

In vielen Fällen erweist es sich als sinnvoll, statt der if-Anweisung die Alternative switch zu benutzen. Entgegen der if-elseif-Auwertung prüft die switch-Bedingung immer nur eine Bedingung bzw. einen Ausdruck (meistens den Wert einer Variablen) und führt je nach Resultat des Vergleichs die angegebenen Befehle aus. Dabei wird nicht entweder true oder false zurückgegeben, sondern mit mehreren Fallunterscheidungen gearbeitet:

ergibt der Vergleich den Fall 1, dann Anweisung 1;

ergibt der Vergleich den Fall 2, dann Anweisung 2;

ergibt der Vergleich den Fall 3, dann Anweisung 3;

Sie verwenden die switch-Anweisung im Script folgendermaßen:

```
switch ($Variable)
{
case "wert1:
Anweisung1;
break;
case "wert2:
Anweisung2;
break;
case "wert3":
Anweisung3;
break;
default:
Anweisung4;
break;
}
```

PHP beginnt die Auswertung am Anfang; sobald PHP den Fall (case) findet, der mit dem Wert korrespondiert, wird die Anweisung ausgeführt, und zwar tapfer so lange, bis das Ende der switch-Anweisung erreicht ist oder bis es auf break stößt. Deswegen also das break. Aus Gründen der Konsistenz schreiben wir es auch nach der default-Anweisung, wobei die default-Anweisung allerdings optional ist.

In Währungen umrechnen, in dem beispielhaft ein Euroumrechner programmiert wird, verwenden wir die `switch`-Bedingung, sodass Sie dort sehen können, wie dieser Befehl konkret eingesetzt wird.

In PHP stehen folgende Kontrollstrukturen zur Verfügung:

- Bedingte Ausführung von Anweisungen und Verzweigungen mit `if`, `elseif` und `else`.
- Fallunterscheidungen mit `switch/case`.
- Abarbeitungsschleifen mit `for` und `foreach`.
- Kopfschleifen mit `while` und Fußschleifen mit `do ... while`.

Bedingte Ausführung und Verzweigung mit if, elseif und else

Eine oder mehrere Anweisungen können abhängig davon ausgeführt werden, ob eine Bedingung erfüllt ist.

```
if($_GET['page'] == 'home') {
$page = file_get_contents('home.dat');
replace_variables('home', $page);
echo $page;
}
```

Der Block bedingt auszuführender Anweisungen wird durch das Schlüsselwort `if` eingeleitet. In Klammern wird dahinter eine Bedingung formuliert, die wahr oder falsch sein kann. Typisch dafür ist die Verwendung von **Vergleichsoperatoren**, da diese ein Ergebnis vom Typ „wahr“ oder „falsch“ liefern. Auch die Verknüpfung von Ausdrücken mit logischen Operatoren kommt häufig vor, da diese ebenfalls Wahr-Falsch-Werte liefern.

Daneben liefern aber auch viele PHP-Funktionen `true` oder `false` als Ergebnis zurück.

```
$name = "Rainer Zufall";
if(isset($name))
echo $name;
```

Die **PHP-Funktion** `isset()` ermittelt, ob eine Variable existiert oder nicht. Wenn ja, wird `true` zurückgegeben, wenn nein, dann `false`. Innerhalb der Klammern von `if()` als Bedingung notiert, bewirkt der Funktionsaufruf, dass die bedingt auszuführenden Anweisungen dann ausgeführt werden, wenn `isset()` den Wert `true` zurückliefert.

Wenn mehrere Bedingungen abhängig von `if` ausgeführt werden sollen, müssen diese als Block in geschweifte Klammern eingeschlossen werden, wie im ersten der obigen Beispiele. Wenn nur eine Anweisung bedingt ausgeführt werden soll, können die geschweiften Klammern entfallen.

Mit `else` kann ein Alternativzweig für die Fälle definiert werden, in denen die `if`-Bedingung nicht erfüllt

ist.

```
if(isset($name))
echo $name;
else
echo "Hier könnte Ihr Name stehen!";
```

Auch bei else gilt: Sollen mehr als eine Anweisung abhängig davon ausgeführt werden, müssen diese in geschweifte Klammern eingeschlossen werden.

Für einfache Fallunterscheidungen können Sie auch noch ein oder mehrere Zweige des Typs `elseif` zwischen `if` und `else` setzen.

```
if($_GET['page'] == 'home') {
$page = file_get_contents('home.dat');
replace_variables('home', $page);
}
elseif($_GET['page'] == 'after_login') {
check_login();
$page = file_get_contents('logged_in.dat');
replace_variables('logged_in', $page);
} else
$page = file_get_contents('login.dat');
echo $page;
```

Bei `elseif` muss ebenso wie bei `if` eine Bedingung formuliert werden, und zwar eine andere als bei `if`. Wenn allerdings wie im Beispiel sowohl bei der `if`-Bedingung als auch bei einer oder mehreren `elseif`-Bedingungen immer wieder abgefragt wird, ob eine bestimmte Variable einen bestimmten Wert hat, dann bieten sich auch Fallunterscheidungen an, wie nachfolgend beschrieben.

Fallunterscheidung mit switch/case

Nicht selten besteht das „Herzstück“ einer größeren PHP-Anwendung letztlich aus einer mehr oder weniger umfangreichen `switch/case`-Konstruktion. Denn wenn ein Script beispielsweise als Container dient, um via GET-Parameter bestimmte Seiten auszugeben, aber bei jeder Seite andere Aufgaben erfüllen muss, dann ist das ein typischer Fall für eine Fallunterscheidung. Um also beim Beispiel zu `if-elseif-else` zu bleiben – dies könnte auch so programmiert werden:

```
switch($_GET['page']) {
case 'home':
$page = file_get_contents('home.dat');
replace_variables('home', $page);
```

```
break;
case 'after_login':
check_login();
$page = file_get_contents('logged_in.dat');
replace_variables('logged_in', $page);
break;
default:
$page = file_get_contents('login.dat');
break;
}
echo $page;
```

Das Konstrukt wird eingeleitet mit switch(`$Variablenname`). Für jeden Wert, der unterschieden werden soll, wird ein `case` Wert: notiert. Zeichenkettenwerte werden dabei wie üblich in einfache oder doppelte hohe Anführungszeichen gesetzt, Zahlenwerte nicht. Im Anschluss daran können beliebig viele Anweisungen notiert werden, die ausgeführt werden, wenn die Fallunterscheidungsvariable diesen Wert hat. Dabei sind keine geschweiften Klammern nötig. Jedoch muss der gesamte `switch`-Block in geschweifte Klammern gesetzt werden.

Am Ende sollte noch an Stelle eines `case ...:` ein `default:` notiert werden. Dies entspricht einem `else`-Zweig und ermöglicht es, alle Fälle zu behandeln, die durch die explizit unterschiedenen Fälle nicht abgedeckt werden.

Von großer Bedeutung ist bei Fallunterscheidungen mit `switch/case` auch die Anweisung `break;`. Wird sie nicht notiert, dann werden alle nachfolgenden Fälle ebenfalls ausgeführt. In manchen Fällen kann das sogar gewünscht sein, meistens jedoch nicht. Wenn also von dem gesamten `switch`-Konstrukt maximal ein Fall ausgeführt wird, dann muss am Ende jedes Falls ein `break;` notiert werden.

Abarbeitungsschleifen mit for und foreach

Den typischen Aufbau einer `for`-Schleife kennen Sie bereits aus JavaSript. Das nachfolgende Beispiel zeigt, wie `for`-Schleifen zur Erstellung einer HTML-Tabelle genutzt werden können:

```
$html = '<table border="1">';
for($i = 1; $i <= 10; $i++) {
$html .= '<tr>';
for($j = 1; $j <= 10; $j++)
$html .= '<td>'.($i * $j).'</td>';
$html .= '</tr>';
}
$html .= '</table>';
echo $html;
```

Das Beispiel erzeugt mittels einer verschachtelten `for`-Schleife eine HTML-Tabelle mit dem kleinen Einmaleins. Dabei wird die Variable `$html` in einer äußeren und dann noch in einer inneren `for`-Schleife sukzessive um entsprechende Inhalte erweitert. Am Ende genügt es, die Variable auszugeben.

Eine `for`-Schleife beginnt mit dem Schlüsselwort `for`. In den Klammern dahinter stehen typischerweise drei Anweisungen. Die erste initialisiert eine Variable mit einem Wert, die dritte verändert den Wert der Variablen und die mittlere formuliert eine Schleifendurchlaufbedingung. Die erste und die dritte Anweisung müssen so geartet sein, dass durch das Verändern der Variablen mit der dritten Anweisung irgendwann ein Wert erreicht ist, der nicht mehr die Durchlaufbedingung erfüllt. Dann ist die `for`-Schleife beendet.

> Es hat sich unter vielen Programmierern eingebürgert,

die Zählvariable einer `for`-Schleife `$i` zu benennen. Sollte wie im Beispiel eine innere `for`-Schleife benötigt werden, die eine andere Variable benutzen muss, kann der nächste Buchstabe im Alphabet verwendet werden.

Im Beispiel ist for(`$i = 1; $i <= 10; $i++`) also so zu lesen:

`$i` wird auf 1 gesetzt. Ist `$i` kleiner oder gleich 10? Ja! Also führe den Schleifenkörper aus. Zähle `$i` außerdem hoch. `$i` wird dadurch 2. Ist `$i` kleiner oder gleich 10? Ja! Also führe den Schleifenkörper aus. Zähle außerdem hoch usw. Irgendwann hat `$i` den Wert 11 und die Schleifendurchlaufbedingung ist nicht mehr erfüllt.

Den Schleifenkörper bilden diejenigen Anweisungen, die bei jedem Schleifendurchgang ausgeführt werden sollen. Bei mehr als einer Anweisung müssen die Anweisungen in geschweifte Klammern eingeschlossen werden.

Neben der klassischen `for`-Schleife bietet PHP auch eine `foreach`-Schleife an. Sie bietet sich speziell an, um alle Elemente eines Arrays zu durchlaufen (**Traversion**) und zwar dann, wenn man innerhalb des Schleifenkörpers keinen Zugriff auf eine Zählervariable wie `$i` benötigt. Nachfolgendes Beispiel zeigt einen Ausschnitt aus einer HTML-Datei:

```
<h1>Städte</h1>
<h2>Aachen</h2>
<p>bla</p>
<p>bla</p>
<p>bla</p>
<p>bla</p>
<h2>Augsburg</h2>
<p>bla</p>
<p>bla</p>
<p>bla</p>
```

```
<h2>Bamberg</h2>
<p>bla</p>
<p>bla</p>
<h2>Berlin</h2>
<p>bla</p>
```

Wenn die Aufgabe gestellt ist, aus diesen Zeilen die in h2-Überschriften notierten Städtenamen zu extrahieren und in einer Aufzählungsliste auszugeben, so bietet sich eine `foreach`-Schleife an:

```
$lines = file("staedte.html");
echo "<ul>\n";
foreach($lines as $line) {
if(preg_match("/<h2>(.*)<\/h2>/", $line, $matches))
echo "<li>{$matches[1]}</li>\n";
}
echo "</ul>";
```

Zunächst wird mit der PHP-Funktion `file()` die HTML-Datei eingelesen – in unserem Beispiel heißt sie **staedte.html**. Gespeichert wird das Ergebnis in `$lines`. Da `file()` eine Datei zeilenweise einliest, ist `$lines` am Ende ein Array, wobei in jedem Element eine Dateizeile steht.

In der `foreach`-Schleife werden alle Zeilen der Reihe nach abgearbeitet. Bei jeder Zeile wird gefragt, ob auf sie das Muster „`<h2>Irgendwas</h2>`“ passt. Wenn ja, steht hinterher in der mit übergebenen Variablen `$matches`, aus der die Suchfunktion `preg_match()` einen Array macht, im zweiten Element (also in `$matches[1]`) der im regulären Ausdruck geklammerte Inhalt, also das, was zwischen <h2> und </h2> steht. Passt eine Zeile auf das Suchmuster, dann wird ein Listenpunkt der Aufzählung ausgegeben.

Der `foreach`-Schleifenkopf hat folgenden typischen Aufbau:

```
foreach($Arrayvariable as $Elementvariable)
```

Damit wird `$Arrayvariable` so oft durchlaufen, wie es Elemente im Array gibt. Innerhalb des Schleifenkörpers kann auf das jeweils aktuelle Element mit `$Elementvariable` zugegriffen werden.

Bei assoziativen Arrays ist auch folgende Variante möglich:

```
foreach($Arrayvariable as $Name => $Wert)
```

Damit kann dann innerhalb des Schleifenkörpers sowohl auf den Schlüsselnamen als auch auf den zugehörigen Wert eines Array-Elements zugegriffen werden.

```
foreach($_POST as $name => $value)
echo "<b>$name:</b> $value</b><br>";
```

Das Beispiel gibt alle im superglobalen Array `$_POST` gespeicherten Elemente mit Feldnamen und Wert aus.

Kopfschleifen mit while und Fußschleifen mit do ... while

While-Schleifen benötigen im Schleifenkopf einen Ausdruck, der wahr oder falsch sein kann, genauso wie `if`-Bedingungen. Die Schleife wird dann so oft durchlaufen, wie der Ausdruck im Schleifenkopf wahr ist. Innerhalb des Schleifenkörpers muss dafür gesorgt werden, dass sich Zustände ändern, damit die Durchlaufbedingung im Schleifenkopf irgendwann nicht mehr erfüllt ist und die Schleife verlassen wird. Ansonsten entsteht eine Endlosschleife.

Schleifen mit `while` eignen sich dann, wenn nicht ermittelbar ist, wie oft die Schleife durchlaufen wird.

```
$now = time();
$then = $now + 5;
$count = 0;
while($now < $then) {
$file = fopen ("http://www.google.com/", "r");
$count += 1;
$now = time();
}
echo "$count mal Google geladen innerhalb 5 Sekunden";
```

Mit der PHP-Funktion `time()` wird ein Unix-Zeitstempel erzeugt, und zwar die aktuell vergangenen Sekunden seit dem 1. Januar 1970. Der Wert wird in `$now` gespeichert. In `$then` wird ein um 5 höherer Wert gespeichert, in dem zu `$now` 5 hinzuaddiert wird.

Unsere `while`-Schleife lassen wir nun 5 Sekunden lang laufen. Erreicht wird das, indem die Schleifendurchlaufbedingung mit `$now < $then` formuliert wird, also „solange der Wert von `$now` kleiner ist als der von `$then`„. Innerhalb des Schleifenkörpers wird in der letzten Anweisung jedes Mal wieder `$now = time()` ausgeführt, d.h., jedes Mal wird `$now` wieder mit einem neuen Unix-Zeitstempel aktualisiert. Dadurch ist `$now` natürlich nach irgendeiner Anzahl von Schleifendurchläufen gleich oder größer `$then`, was zum Abbruch der Schleife führt.

Innerhalb der Schleife wird die Startseite von

google.com geladen. Auch hier begegnet uns übrigens wieder die „Leichtigkeit“ von PHP: mit der gleichen Funktion (`fopen()`), die zum Öffnen lokaler Dateien dient, lassen sich auch beliebige Webseiten

einlesen. Der Programmierer muss keine umständlichen Socketverbindungen erstellen, HTTP-GET-Kommandos erzeugen oder Sonstiges. Das alles erledigt PHP für ihn.

Da das Laden übers Internet natürlich einen Moment lang dauert, wird unsere `while`-Schleife innerhalb von 5 Sekunden nicht allzu oft durchlaufen.

Wenn die Schleifendurchlaufbedingung schon gleich beim ersten Prüfen nicht erfüllt ist, wird der Schleifenkörper kein einziges Mal ausgeführt. Da es jedoch Fälle gibt, in denen er wenigstens einmal ausgeführt werden soll, egal ob die Schleifendurchlaufbedingung zutrifft oder nicht, gibt es auch fußgesteuerte Schleifen mit `do ... while:`

```
$Zahl = 10;
do {
echo $Zahl;
$Zahl += 1; }
while($Zahl < 10);
```

In diesem Beispiel würde die Schleife, wenn die Durchlaufbedingung für `while` oberhalb des Schleifenkörpers notiert wäre, kein einziges Mal ausgeführt, weil `$Zahl` schon höher ist als die Schleifendurchlaufbedingung erlaubt. Durch das Konstrukt über `do ... while` wird der Schleifenkörper jedoch zuerst ausgeführt, ehe die Durchlaufbedingung geprüft wird. Die Prüfung entscheidet dann über den nächsten Schleifendurchlauf.