Markt+Technik

Christine Peyton

PHP

Der leichte Einstieg

Pocket

PHP

Pearson Education

Markt+Technik

Christine Peyton

PHP

Der leichte Einstieg

Markt+Technik Verlag

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.

10 9 8 7 6 5 4 3 2 1

04 03 02

ISBN 3-8272-6299-2

ein Imprint der Pearson Education Deutschland GmbH,
Martin-Kollar-Straße 10–12, D-81829 München/Germany

Einbandgestaltung: Grafikdesign Heinz H. Rauner, Gmund
Lektorat: Melanie Kasberger, mkasberger@pearson.de
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: reemers publishing services gmbh, Krefeld, www.reemers.de
Druck und Verarbeitung: fgb, freiburger graphische betriebe, www.fgb.de
Printed in Germany

Inhaltsverzeichnis

Vorwort

PHP ist eine Open Source Scripting Language, die – zu Recht – seit einiger Zeit zusehends an Bedeutung gewinnt und immer populärer wird. Wenn Sie Dynamik in Ihre Webseiten-Projekte bringen, Seiten personalisieren und Daten, die auf einer Webseite generiert wurden, per Dateisystem oder Datenbanken sichern möchten, ist PHP eine sehr gute Wahl.

Das Buch

Sie halten ein Buch in den Händen, das PHP erklärt und sich in erster Linie an den Einsteiger wendet und zwar nicht nur den Einsteiger in PHP, sondern an Menschen, für die auch das Programmieren selbst neues Terrain ist (die Projekte der letzten Kapitel gehen allerdings über triviale Aufgaben hinaus, sodass auch Menschen mit grundlegenden Programmierkenntnissen von diesem Buch profitieren können). Deswegen beginnt das Buch ganz langsam mit den ersten Schritten und erläutert die Grundlagen von PHP ausführlich und anhand vieler kleiner Beispiele. Die Grundlagen machen Sie vertraut mit dem Aufbau einer PHP-Datei, mit der Syntax und mit zahlreichen Befehlen, die beim Einsatz von PHP eine Rolle spielen.

In den nächsten Kapiteln werden dann konkrete praxisorientierte Aufgaben vorgestellt und gelöst. So lernen Sie beispielsweise, wie Sie ein Kontaktformular erzeugen, das je nach Dateneingabe unterschiedliche Reaktionen hervorruft, und wie Sie einen Währungsumrechner und einen Counter, der die Aufrufe einer Webseite registriert, programmieren. Auch wie man ein Gästebuch für die Eingabe von Meinungen und Kommentaren erstellt, zeigen wir Ihnen an einem Beispiel.

In den letzten Kapiteln geht es hauptsächlich um die Möglichkeit, Dateien »upzuloaden« und darum, diese Dateien entweder in einer Textdatei oder in einer Datenbank (MySQL) zu verwalten. Auch Ihre kreativen Impulse werden berücksichtigt: ein Kapitel widmet sich speziell den Möglichkeiten, mit PHP Grafiken zu erzeugen.

Aufbau des Buches

Wie schon erwähnt, richtet sich dieses Buch an den Einsteiger. Deswegen legen wir langsam los. Die Beispiele der ersten Kapitel widmen sich einfachen, grundlegenden Aufgaben, die Sie mühelos mitprogrammieren werden können. Die jeweiligen PHP-Codes beschreiben und erklären wir ausführlich und mehr oder minder Zeile für Zeile. Typische Programmiertermini, die zunächst nur Verwirrung stiften, setzen wir dabei nicht voraus.

Nach diesen »Fingerübungen« lösen wir in den Kapiteln weiter hinten Aufgaben, die komplexer sind und erste Programmiererfahrungen voraussetzen (die Sie entweder in den vorangehenden Kapiteln oder anderweitig erworben haben). Aber auch hier erklären wir detailliert den Aufbau der Scripts und die verwendeten Befehle. Dabei verfahren wir so,

dass die einzelnen Scriptteile in den Listings gezeigt und dann »auseinandergenommen« und Stück für Stück erläutert werden.

Ein Buch über eine Programmiersprache kann kein Bilderbuch sein, dennoch gibt es im Buch zahlreiche Abbildungen, die jeweils deutlich machen, welchen Effekt ein Script oder Scriptfragment hat und wie sich die Webseite mit der Entwicklung des gesamten Scripts nach und nach verändert.

Bedenken Sie bitte, dass dies ein Buch über PHP und nicht über HTML ist. Es setzt also voraus, dass Sie mit der Beschreibungssprache HTML einigermaßen vertraut sind. Wir erklären zwar auch die jeweils verwendeten HTML-Codes, aber dies eher kurz und en passant. Sind Sie auch auf diesem Gebiet ein Einsteiger, empfiehlt es sich, dass Sie zumindest ein HTML-Buch zur Hand haben[1] oder das vorzügliche Internetangebot von Stefan Münz als Referenz benutzen, zu finden unter *http://selfhtml.teamone.de*

Tipps zum Lesen und Lernen

Gehören Sie zu dem Kreis der Einsteiger, sollte Sie auf jeden Fall unseren Ratschlag beherzigen, mit der Durchsicht das Buches nicht – vielleicht entgegen Ihre sonstigen Gewohnheiten! – auf den letzten Seiten zu beginnen. Sie brauchen die Kenntnisse, die in *Kap 3: PHP – Die Grundlagen* und in den Anfangskapiteln vermittelt werden, auf jeden Fall, um die Beispiele der nächsten Kapitel nachvollziehen zu können. Am besten ist es tatsächlich, wenn Sie die Aufgaben der Reihe nach durcharbeiten (auch wenn Sie in der Praxis ganz andere Dinge mit Ihren Webprojekten vorhaben), denn der Schwierigkeitsgrad steigt von vorne nach hinten. Sie wären verloren und wahrscheinlich schnell entmutigt, wenn Sie sich beispielsweise gleich an den Datei-Upload aus Kapitel 11 heranwagen würden.

Sie können natürlich die Beispiel-Scripts einfach mechanisch abschreiben und – sofern Sie dies präzise tun – auch so zum Ziel kommen, aber auf diese Weise würden Sie nicht PHP lernen, und Sie wären hilflos, sobald Sie selbsttätig eine Aufgabe bewältigen möchten oder müssen. In einigen wenigen Fällen bauen die Kapitel auch direkt aufeinander auf, d.h. wir greifen in dem einen Kapitel auf ein Script oder Scriptfragment zurück, das wir bereits in einem anderen Kapitel entwickelt haben.

Als Ergänzung zu diesem Buch stehen im World Wide Web eine ganze Reihe von Seiten zur Verfügung, die ebenfalls in PHP einführen, eine Befehlsreferenz enthalten oder beispielsweise auch konkrete Lösungen und komplette Scripts anbieten. Suchen Sie einfach den Begriff »PHP«, z.B. mit der Suchmaschine auf *www.google.de* bzw. *www.google.com*.[2]

1 Zum Beispiel *HTML/XHTML – M+T Pocket* von Günther Born, ISBN 3-8272-6088-4 oder *Jetzt lerne ich HTML* von Harald Taglinger, ISBN 3-8272-5717-4.

2 Oder bedienen Sie sich der ausgezeichneten Literatur, die es zu diesem Thema gibt, z.B. *Jetzt lerne ich PHP und MySQL* von Sven Letzel und Robert Gacki, ISBN 3-8272-6202-X, *MySQL in 21 Tagen* von Mark Maslakowski, ISBN 3-8272-5850-2 u.v.a.m.

Übrigens können Sie die gesamten in dem Buch entwickelten Scripts auch von der Website *www.brain-and-trust.de* downloaden, wenn Ihnen die Tipperei zu viel wird. Vergessen Sie darüber aber nicht, dass Sie PHP erlernen möchten, was nur klappt, wenn Sie tatsächlich so viel selber programmieren, wie es geht!

Damit Sie weiterhin den größtmöglichen Nutzen aus unseren Büchern ziehen können, würden sich Autorin und Verlag über eine Rückmeldung von Ihnen – positive wie negative Kritik – freuen. Sie können dazu das entsprechende Formular auf *www.brain-and-trust.de* nutzen oder – bevorzugte Möglichkeit – eine E-Mail an *christine.peyton@mut.de* schicken.

Viel Spaß beim Scripten wünscht Ihnen zunächst
Christine Peyton

Kapitel 1
Was ist PHP?

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 HMTL-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 *www.php.net*.

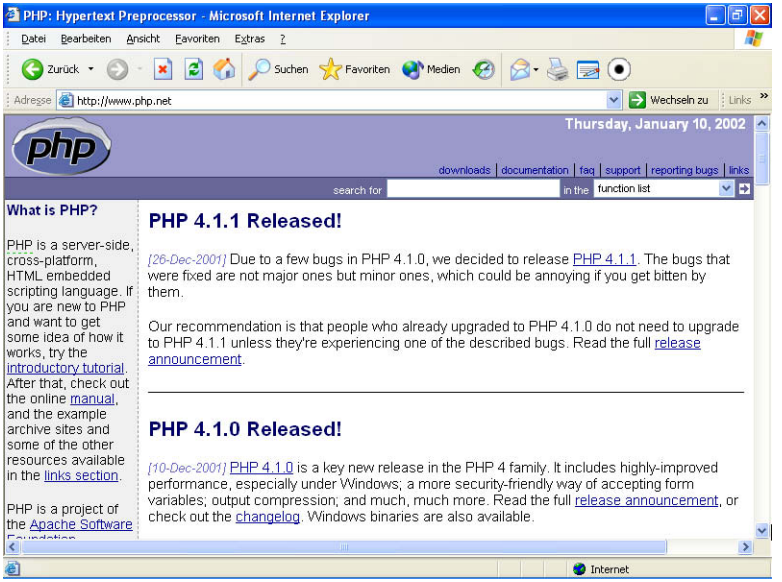

Bild 1.1: Die offizielle PHP-Site

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 Buches 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.

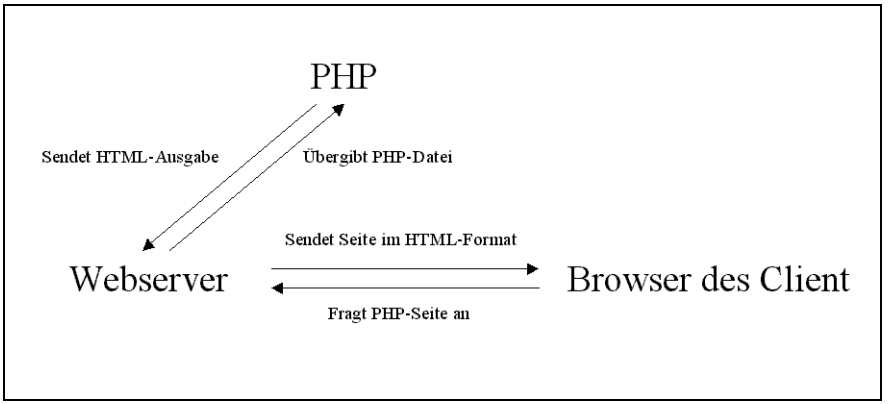

Bild 1.2: Die Anfrage nach einer PHP-Seite

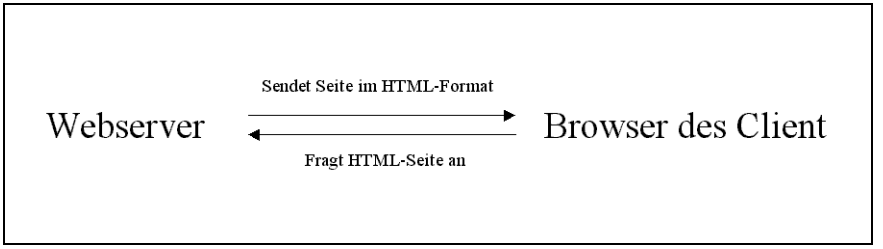

Bild 1.3: Die Anfrage nach einer HTML-Seite

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.

Kapitel 2
Installation

Um Sie auch bei den Schritten zu unterstützen, die im Vorfeld notwendig sind, beschreiben wir in diesem Kapitel die Installation der aktuellen Versionen von *Apache*, *PHP 4* und *MySQL* für Windows XP Professional.

Die Installation auf *Linux* beschreiben wir nicht, da diese Programme mit den neueren Distributionen von *Linux* bei der Installation vorausgewählt werden können, sodass sie auf Ihrem Linux-Rechner bereits vorhanden sein dürften.

Für die Arbeit mit *PHP* brauchen Sie außer PHP einen Web-Server und eine Datenbank, um in Ihren Webseiten auf Informationen aus Datenbanken zugreifen zu können. Web-Server gibt es einige, z.B. den *Internet Information Server* (IIS), *OmniHTTPd* oder *Apache*. Auch als Datenbanken kommen einige in Frage, aber *MySQL* ist wohl am weitesten verbreitet (in dem Kapitel über *MySQL* schreiben wir mehr dazu). Alle drei Programme können Sie als Download kostenfrei aus dem Internet beziehen; finanzielle Überlegungen sind also kein Grund, in letzter Minute einen Rückzieher zu machen! *Apache* bietet sich auch deswegen an, weil fast alle Provider diesen Webserver benutzen und so die Testumgebung dicht an der Realität liegt.

Wenn wir hier also PHP in Kombination mit *Apache* und *MySQL* installieren und verwenden, bleiben wir auf einem ausgetretenen Weg, der sich aber in vieler Hinsicht bewährt hat. Man spricht in der Kurzform auch von *WAMP* (Windows-Apache-MySQL-PHP)!

Hinweis

Wie wir leidvoll erfahren mussten, ändert sich die Installation der einzelnen Programme sowie die benötigten Einträge in die Konfigurationsdateien von Version zu Version. Mitunter unterscheiden sie sich auch je nach Windows-Version. Eine Recherche im Internet hat mehrere leicht unterschiedliche Varianten des nachfolgend beschriebenen Installationsvorgangs erbracht. Unsere Erfahrung und diese Recherche lassen uns vermuten, dass es auch wieder leichte Änderungen bei den nächsten Versionen geben wird, sodass wir nicht hundertprozentig für die Richtigkeit aller hier genannten Einträge in den Konfigurationsdateien garantieren können.

Apache installieren

Voraussetzung

Um *Apache* installieren zu können, benötigen Sie den *MSI* (Microsoft Windows Installer) Version 1.10 oder höher. Um herauszufinden, ob *MSI* bei Ihnen bereits installiert ist, starten Sie die Eingabeaufforderung: START>ALLE PROGRAMME>ZUBEHÖR>EINGABEAUFFORDERUNG und geben Sie ein:

```
MSIEXEC
```

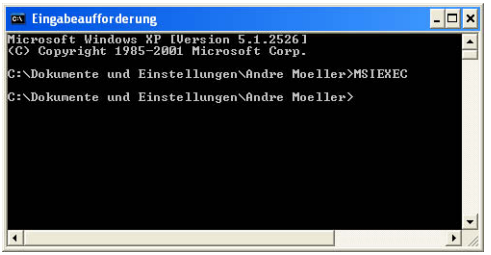

Bild 2.1: So testen Sie, ob MSI installiert ist.

Wenn *MSI* installiert ist, erhalten Sie eine Fehlermeldung, die die Versionsnummer mit anzeigt.

Bild 2.2: Diese Meldung erhalten Sie, wenn MSI bereits installiert ist.

Sollte er nicht installiert sein, finden Sie einen Link zum Download des Installers auf der Seite *http://www.apache.org/dist/httpd/binaries/win32*. Laden Sie ihn herunter und installieren Sie ihn. Dieser Installer ist bei *Windows 2000* und *Windows ME* übrigens inklusive, ebenso bei *Windows XP*. Auch wenn Ihr Rechner mit einem anderen Betriebssystem läuft, ist es nicht unwahrscheinlich, dass die Installation eines anderen Programmes diesen Installer eingerichtet hat.

Apache downloaden

Rufen Sie die Seite *www.apache.org* auf. Klicken Sie hier auf den Link HTTP-SERVER.

Auf der nächsten Seite finden Sie den Bereich DOWNLOAD. Klicken Sie auf einen der Links, die zu den Download-Seiten führen.

Sie erhalten eine Liste der angebotenen Versionen. Wählen Sie hier zunächst den Ordner BINARIES und auf der nächsten Seite dann den Ordner WIN32. Nach den Infos, die Sie zunächst lesen können, suchen Sie die aktuelle Version für x86 Rechner (Intel-kombatibel) mit der Dateiendung *.msi*. Im Beispiel nehmen wir *apache_1.3.22-win32-x86.msi*. Diese Datei laden Sie herunter.

Installation

Nach dem Download starten Sie im Explorer per Doppelklick auf die soeben heruntergeladene Datei den Installationsprozess. Es öffnet sich der Installations-Wizard.

Wandern Sie mit NEXT durch die Dialoge. Stimmen Sie der Lizenzbedingung zu. Lesen Sie die nächste Hinweisseite durch und springen Sie zum nächsten Dialog.

Da wir davon ausgehen, dass Sie den Web-Server nur zu Testzwecken bei sich zu Hause installieren, können Sie in dem Dialog zur Eingabe des Domain-Namens und Server-Namens beide frei vergeben. Am besten Sie verwenden als Domain-Name Ihre IP-Adresse und als Servername *localhost*. Auch wenn der Dialog Ihnen rät, *Apache* als SERVICE zu installieren, sind Sie unserer Meinung nach besser beraten mit der unteren Option, d.h. den Server so zu installieren, dass er manuell gestartet werden muss, da es voraussichtlich reicht, wenn er läuft, während Sie Tests durchführen. Mit der Installation als SERVICE läuft der Web-Server, sowie der Rechner gestartet wird.

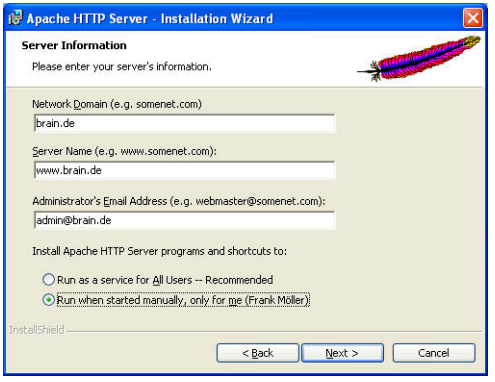

Bild 2.3: Die Server-Informationen angeben

Anschließend wählen Sie den Installationsumfang; entscheiden Sie sich hier für COMPLETE, um auch die Dokumentation mit zu installieren.

Im nächsten Dialog bestimmen Sie den Ordner, in den der *Apache-Web-Server* installiert wird. Klicken Sie auf CHANGE, wenn Sie den vorgeschlagenen Standardordner ändern möchten. Lassen Sie den Installations-Wizard dann die Arbeit abschließen.

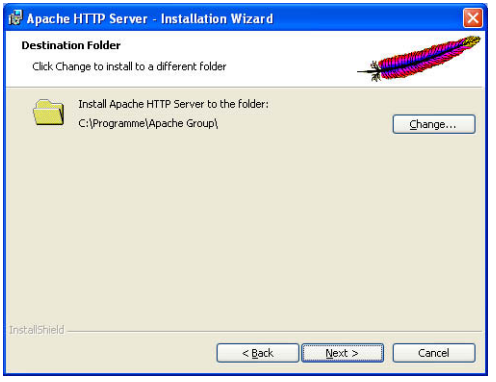

Bild 2.4: Den Ordner für Apache bestimmen

Nach der Installation starten Sie den Web-Server über START>ALLE PROGRAMME>APACHE HTTP SERVER>START APACHE IN CONSOLE. Um nun die Funktionstüchtigkeit des Web-Servers zu testen, rufen Sie Ihren Browser auf und geben *http://localhost* ein. Bei Erfolg erhalten Sie die folgende Meldung (siehe Bild 2.5).

Dokumente speichern

In der Standardeinstellung müssen Sie die Dateien für Ihre Webseiten im Unterordner HTDOCS des Installationsverzeichnisses ablegen. Bis jetzt kann der Web-Server nur HTML-Dateien verarbeiten, es fehlt noch die Installation von *PHP*, die im nächsten Abschnitt vorgenommen und erklärt wird. Die Änderungen des Ordners für die Ablage der Webseiten werden wir bei der Anpassung von *Apache* für den PHP-Einsatz mit durchführen.

PHP installieren

Die Installation von *PHP* ist zwar im Prinzip auch schnell gemacht, aber die dann erforderlichen Anpassungen der Konfiguration sowohl von *PHP* als auch *Apache* erfordern ein bißchen Aufmerksamkeit und sind etwas verzwickt. Nehmen Sie sich also noch ein bisschen Zeit.

Bild 2.5: Glückwunsch, Apache ist installiert!

PHP downloaden

Auch hier organisieren Sie sich zunächst die aktuelle Version. Rufen Sie dazu die Seite *www.php.net* auf und öffnen Sie den Download-Bereich. Den entsprechenden Link finden Sie ziemlich klein oben rechts auf der Seite. Sie sehen auf der Download-Seite unter der Überschrift WINDOWS BINARIES die aktuelle Version. Wählen Sie hier die größere Datei, die auch die Konfiguration als API-Modul für den *Apache* zulässt. Die Installation als Modul hat einige Vorteile, u.a. was die Geschwindigkeit angeht.

Nach dem Download müssen Sie die Zip-Datei entpacken. Legen Sie dafür am besten vorab einen neuen Ordner an, z.B. C:\PHP, da dies beim Entpacken nicht geschieht.

Hinweis

Für Windows 95 brauchen Sie ein DCOM-Update, zu finden auf der Microsoft-Webseite bei den Downloads.

Anpassung der PHP-Konfiguration

Nun geht es weiter mit einigen Anpassungen. Sie müssen verschiedene Dateien hin und her kopieren und Änderungen an der *php.ini*-Datei vornehmen. Keine Angst, wir beschreiben es detailliert.

1. Öffnen Sie im Explorer den PHP-Ordner und kopieren Sie die Datei *php4ts.dll* sowie die Datei *php4ts.lib* in den System-Ordner von Windows: WINNT\SYSTEM32 oder WINDOWS\SYSTEM (je nach Betriebssystem).
2. Dann öffnen Sie den Ordner SAPI und kopieren alle .dll-Dateien aus diesem Ordner bis auf die Datei *php4apache.dll* ebenfalls in den System-Ordner. (Kopieren Sie auch die Datei *php_gd.dll* aus dem Ordner EXTENSIONS in den Systemordner, wenn Sie diese Erweiterung nutzen möchten.)
3. Kopieren Sie auch die Datei *php4apache.dll*, die Sie unter PHP\SAPI finden, da sie in den im Schritt 4 benannten Ordner eingefügt werden soll.
4. Wechseln Sie nun in das Verzeichnis, in das Sie den *Apache Web-Server* installiert haben. In der Standardeinstellung ist dies: C:\PROGRAMME\APACHE GROUP\APACHE. Öffnen Sie hier den Unterordner MODULES und fügen Sie die Kopie der Datei *php4apache.dll* ein.
5. Gehen Sie zurück in das PHP-Verzeichnis und kopieren Sie nun die *php.ini-dist* in das Installationsverzeichnis von Windows. Dies ist normalerweise C:\WINDOWS oder C:\WINNT oder C:\WINNT40. Sie müssen diese Datei dann umbenennen in *php.ini*.

Nun ist die Anpassung der *php.ini* an der Reihe. Dazu öffnen Sie mit einem Doppelklick auf den Dateinamen die *php.ini* im Editor. Als Erstes brauchen Sie die Zeile *extension_dir* (Tipp: Sie gelangen am schnellsten in diese Zeile, wenn Sie über BEARBEITEN>SUCHEN nach dem Begriff suchen lassen). In der Zeile müssen Sie das Verzeichnis angeben, in das Sie die DLLs eben kopiert haben (vermutlich WINDOWS\SYSTEM). Es heißt dann in der Zeile:

```
extension_dir= "C:\Windows\System"
```

Um die GD-Bibliothek mitzustarten, suchen Sie die Zeile *extension=php_gd.dll* und entfernen vor dieser Zeile das Kommentierungszeichen, also den Doppelpunkt.

Den Web-Server anpassen

Es ist noch nicht alles getan! Jetzt müssen Sie noch den Web-Server anpassen. Beenden Sie gegebenenfalls Apache und rufen Sie auf: START>ALLE PROGRAMME>APACHE HTTP SERVER >CONFIGURE APACHE SERVER>EDIT THE APACHE HTTPD.CONF CONFIGURATION FILE. Die entsprechende Datei (*httpd.conf*) wird im Editor geöffnet. Fügen Sie die folgenden zwei bzw. drei Zeilen einfach ein (möglichst in der Nachbarschaft ähnlicher Zeilen):

```
LoadModule php4_module modules/php4apache.dll
Addtype application/x-httpd-php .php

AddModule mod_php4.c
```

(Je nach Version von Apache ist das Hinzufügen dieser Zeile erforderlich oder nicht)

Damit Ihre Dokumente nicht in der untersten Hierarchie des Apache-Ordners herumliegen, legen Sie sich einen neuen Ordner z. B. direkt auf C: an und geben den entsprechenden

Pfad als DocumentRoot in der *httpd.conf* an. In diesen Ordner legen Sie dann alle Ihre Web-Seiten. Wenn der Ordner beispielsweise WWW heißt, sieht das dann so aus:

```
DOCUMENTROOT "C:/www"
```

Damit der Apache-Webserver nicht nur die Index-HTML-Dateien, sondern auch die PHP-Dateien beim Aufruf eines Ordners anzeigt, suchen Sie nach der Zeile `DirectoryIndex`. Hier steht bisher nur *index.html*. Fügen Sie hier *index.php* hinzu. Je nachdem, ob Sie *index.php* vor oder hinter *index.html* geschrieben haben, wird, wenn beide Dateien in einem Ordner vorhanden sind, die eine oder die andere zuerst angezeigt. In der Zeile heißt es nun:

```
DirectoryIndex index.php index.html
```

Vergessen Sie nicht das Speichern der Änderung an der Datei *httpd.conf*.

Nun können Sie die Installation testen:

Starten Sie Apache neu über START>ALLE PROGRAMME>APACHE HTTP SERVER>START APACHE IN CONSOLE. Hier müsste nun in dem Fenster der Console zusätzlich auftauchen: PHP/4.1.1. running. Ist das der Fall, war die Installation und Anpassung erfolgreich und Sie dürfen sich gratulieren. Hat es nicht geklappt, wird ein Error angezeigt. Dann müssten Sie die Anpassungsschritte erneut überprüfen.

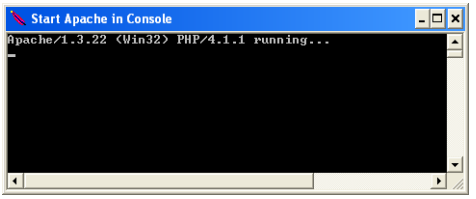

Bild 2.6: Die Erfolgsmeldung!

MySQL installieren

Wenn Sie bei Ihren zukünftigen Projekten auch mit Datenbanken operieren möchten, (was auch hier im Buch gemacht wird), müssen Sie außerdem eine Datenbank installieren. Warum wir hier *MySQL* wählen, erklären wir etwas näher gleich zu Beginn des Kapitels *MySQL – Die Grundlagen*.

MySQL downloaden und installieren

Rufen Sie die Seite *www.mysql.com* auf, dort gibt es den Link DOWNLOADS. Sie finden hier das aktuelle »stable release«, bei uns im Beispiel *MySQL 3.23*. Auf der Download-Seite finden Sie etwas weiter unten auf der Seite die Windows-Downloads als Zip-Dateien.

Anschließend müssen Sie den Server auswählen, von dem Sie *MySQL* beziehen möchten. Naturgemäß nehmen Sie einen Server in Ihrer Nähe. Nachdem Sie die Datei heruntergeladen haben, wird sie entpackt. Klicken Sie dazu im Explorer doppelt auf die entsprechende MySQL-Datei.

In dem Ordner, in den Sie die Datei entpackt haben, finden Sie die Datei *setup.exe*, die Sie mit einem Doppelklick starten. Es öffnet sich ein Installationsfenster, deren Dialoge Sie mit NEXT durchwandern. Auf dieser Wanderung können Sie das Installationsverzeichnis ändern, wir empfehlen aber, den Vorschlag zu übernehmen. Wählen Sie die Installationsvariante CUSTOM und lassen Sie alle Elemente aktiviert.

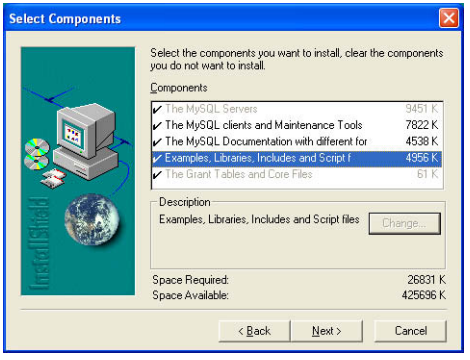

Bild 2.7: Übernehmen Sie die Komponenten.

Die Dateien werden dann kopiert und zu guter Letzt schließen Sie den Vorgang mit FINISH ab.

Bild 2.8: Das Setup von WinMySQLadmin

Gehen Sie in das neu erstellte Verzeichnis; dies ist MYSQL, sofern Sie es bei der Voreinstellung belassen haben. Dort finden Sie im Ordner BIN die Datei *winmysqladmin.exe*. Starten Sie diese Datei per Doppelklick. Geben Sie im ersten Dialog einen frei wählbaren Usernamen und ein Passwort ein und merken Sie sich diese Namen gut. Diese Kombination von Username und Passwort trägt *WinMySQLadmin* als Zugangsdaten für *MySQL* ein. Anschließend finden Sie in der Taskleiste eine kleine Ampel.

Bild 2.9: Die Ampel zum An- und Ausschalten von MySQL

Ist die Ampel grün, läuft der MySQL-Server. Wenn Sie auf die kleine Ampel klicken, öffnet sich das Menü mit den Befehlen zum Schließen des Tools (Ampel) und zum Beenden der MySQL-Datenbank (STOP THE SERVER). Dieses kleine Tool wird automatisch beim Hochfahren mitgestartet, da es sich in AUTOSTART eingetragen hat. Mit der Ampel können Sie den MySQL-Server schnell an- bzw. ausschalten, wenn endlich der Feierabend droht.

Kapitel 3
PHP – Die Grundlagen

Eine neue Programmiersprache zu lernen muss damit beginnen, zunächst die Grundlagen der Sprache zu verstehen. Diese Grundlagen sollen in diesem Kapitel geklärt werden. Sie erfahren zunächst, wie ein PHP-Script aufgebaut wird und wie Sie HTML und PHP kombinieren. Dann werden beispielhaft wichtige und gebräuchliche Funktionen/Befehle vorgestellt und an kleinen Beispielen wird kurz erläutert, wie Sie Befehle benutzen. Ein wichtiger Baustein von PHP (und anderen Programmiersprachen) sind Variablen. Wir klären also, was Variablen sind, welche unterschiedlichen Variablentypen es gibt und wie Sie sie einsetzen.

Entscheidend für dynamische Webseiten ist Flexibilität. Damit ist gemeint, dass Scripts Bedingungen auswerten können und je nach dem Ergebnis eines Vergleiches (true/false) ein anderes Ereignis eintritt. Um diese Möglichkeit von Fallunterscheidungen nutzen zu können, müssen Sie Verzweigungen oder Kontrollstrukturen kennen lernen.

Keine Programmiersprache ohne Schleifen! Mit Schleifen geben Sie an, wie oft ein bestimmtes Programmstück ausgeführt wird – wie Sie sich denken können, ein entscheidendes Element eines Scripts. Wir werden also, jeweils untermauert mit einigen Beispielen, erklären, wie Sie mit der `while`-Anweisung und der `for`-Schleife umgehen.

Im letzten Abschnitt kommen wir zu den regulären Ausdrücken. Leser mit Programmiererfahrung (beispielsweise Perl) wissen, was gemeint ist, als Einsteiger können Sie an dieser Stelle kaum ahnen, worum es sich handelt. Reguläre Ausdrücke gehören eher in die Welt des fortgeschrittenen Programmierens, da sie aber in bestimmten Situationen sehr nützlich sein können, werden wir Sie hier zumindest mit den Grundlagen vertraut machen und ihren Gebrauch an einem Beispiel demonstrieren.

Dieses Beispiel ist – zugegebenermaßen – ausufernder geworden, als ursprünglich beabsichtigt. Wir testen mithilfe eines regulären Ausdrucks die korrekte Eingabe einer E-Mail-Adresse; dabei haben wir versucht, alle erlaubten und unerlaubten Zeichen und die diversen Kombinationsmöglichkeiten abzudecken. Dazu ist – wie sich zeigte – ein Ausdruck erforderlich, der es in sich hat! Wer Spaß hat an ein bisschen Tüftelei, kann versuchen, das Beispiel nachzuvollziehen (am besten mitzuspielen). Als Belohnung winkt ein regulärer Ausdruck, den Sie genauso auf Ihren zukünftigen Webseiten einsetzen könn(t)en, sofern Besucher dort ihre E-Mail-Adresse eingeben. (Ja, wir wissen, es gibt verlockendere Belohnungen.)

Noch ein Hinweis: Dieses Kapitel deckt – wie gesagt – die Grundlagen ab, behandelt aber bei weitem nicht alle Befehle und sonstigen Möglichkeiten von PHP. Dies entspricht dem im Vorwort bereits geschilderten projektorientierten Konzept dieses Buches, das versucht, Ihnen PHP an praxisnahen Beispielen unter dem Einsatz von Befehlen, die tatsächlich gebraucht werden, nahe zu bringen. Falls Sie hier dennoch eine Erklärung schmerzlich

vermissen: Werfen Sie einen Blick in die Befehlsreferenz, dort finden Sie unter Umständen das Gesuchte.

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.

Hinweis

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>
```

Listing 3.1: Das Grundgerüst für ein Script mit HTML-Elementen und PHP-Code

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. Sie werden weiter hinten in den Kapiteln, in denen konkrete Aufgaben gelöst werden, Beispiele (Cookies setzen, Header-Informationen angeben) finden, in denen so vorgegangen wird.

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 Buch 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 folgen-

den 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.)

Sie können in manchen Fällen auch zwei separate Dokumente erstellen, ein HTML-Dokument und ein PHP-Dokument. In dem HTML-Dokument wird dann auf die PHP-Datei, die die Auswertung übernimmt, verwiesen. Diese Methode hat allerdings viele Nachteile bei der Verarbeitung. Da diese Verfahrensweise das Zusammenspiel von PHP und HTML bzw. die Übergabe von Inhalten (beispielsweise eines Formularfeldes) an die PHP-Scriptdatei sehr anschaulich macht, werden wir für die Lösung der ersten konkreten Aufgabe, die in *Kapitel 4, Ein Kontaktformular erstellen*, beschrieben wird, zunächst zwei Dokumente erstellen anstatt einer einzigen PHP-Datei, die den PHP-Programmcode und HTML kombiniert.

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>
<?php
echo "hier steht ein neuer PHP-Code";
?>
</body></html>
```

Listing 3.2: Ein Script, in dem PHP und HTML gemischt werden

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 Browser gibt dann folgende Seite aus:

Bild 3.1: Die Ausgabe im Browser – ein fett formatierter Satz (<b>) und eine horizontale Linie <hr>

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!
```

Hinweis

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. Aus drucktechnischen Gründen können wir selbst diese Empfehlung hier nicht konsequent einhalten, die meisten abgebildeten Scripts sind optisch also nicht vorbildlich!

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. Bild 3.2 gibt einen Ausschnitt der Tabelle wieder.

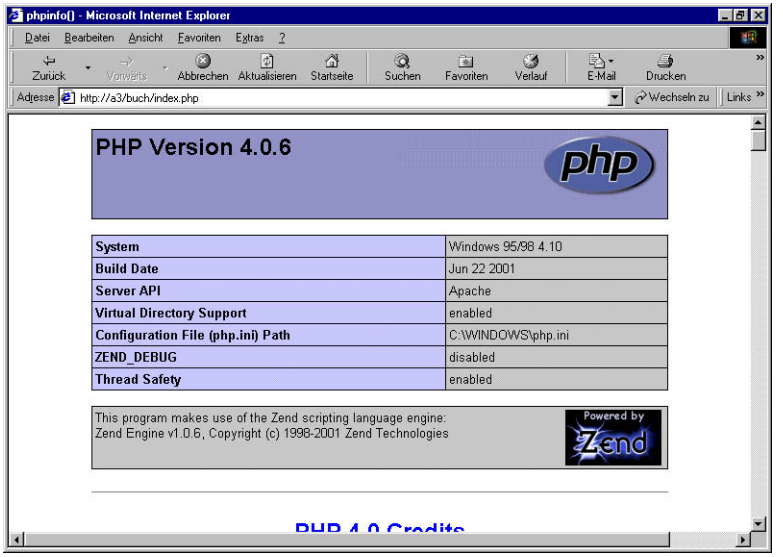

Bild 3.2: Mit phpinfo() können Sie sich Informationen über die spezifische PHP-Installation ausgeben lassen.

Sonderzeichen

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";
```

Dies zeigt das Bild 3.3. Sie sehen die Eingabe im Editor und die Ausgabe im Browser.

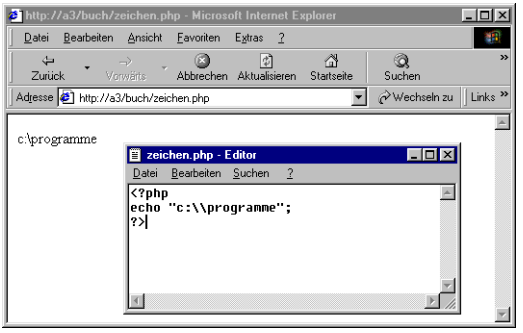

Bild 3.3: PHP erwartet einen zweiten Backslash.

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
\\	Backslash
\$	Dollarsymbol
\"	doppelte Anführungszeichen

Tabelle 3.1: Ein Überblick über die Zeichen, die zur Ausgabe spezieller Zeichen verwendet werden

Kommentare

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!

Hinweis

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.

Variablen

Ein entscheidender Baustein in PHP-Scripts sind Variablen, mit denen Sie im Prinzip ständig arbeiten. Sie deklarieren (so heißt es professionell) Variablen im Script. Man könnte Variablen beschreiben als Behälter für Daten. Sie speichern temporär (während der Lauf-

zeit des Scripts) die Daten bzw. Werte, die ihnen zugewiesen werden. (Technisch gesehen verhält es sich noch ein bisschen anders: Die Namen von Variablen verweisen auf einen bestimmten Speicherplatz im Rechner, an dem der Inhalt der Variable gespeichert ist.)

Eine bereits definierte Variable kann mit anderen Daten kombiniert und unter Beibehaltung des Ursprungswerts ergänzt und erweitert werden. Variablen sind also von Natur aus sehr flexibel oder eben: variabel! Sobald die Variable deklariert wurde, ist sie »einsatzfähig«.

Syntax und Wertezuweisung

Der Name einer Variablen ist frei wählbar. Die Schreibkonvention erlaubt Buchstaben, Zahlen und Unterstriche. Der Name darf keine Leerstellen oder nicht-alphanumerischen Zeichen enthalten.

Ein gültiger Variablenname beginnt mit einem Buchstaben oder einem Unterstrich. Entscheidend ist: Allen Variablennamen ist ein Dollarzeichen vorangestellt. Der Wert wird nach einem Gleichheitszeichen (Zuweisungsoperator) zugewiesen und das Semikolon markiert das Ende. Werte, die aus Text bzw. Zeichenketten (eine genauere Erklärung zu Zeichenketten finden Sie im Abschnitt *Datentypen*) bestehen, werden in Anführungszeichen gesetzt. Wenn Sie Zahlen zuweisen, dürfen keine Anführungszeichen verwendet werden, tun Sie es doch, werden Zahlen wie Text behandelt, sodass man nicht mit ihnen rechnen könnte (so die Theorie, aber PHP ist »programmiererfreundlich« und ändert den Variablentyp bei Berechnungen wenn möglich in das benötigte Format). Wenn mit Zahlen nicht gerechnet werden soll, dann werden sie wie Strings in Anführungszeichen gesetzt.

Ein paar Beispiele für Variablennamen und die Wertezuweisung:

```
$vorname = "Daniel";
```

Dieser Variablen wurde der Wert *Daniel* zugewiesen. Schreiben Sie im Script:

```
echo $vorname;
```

wird im Browser ausgegeben: `Daniel`.

Weitere Beispiele für gültige Variablennamen:

```
$nummer1 = 100;
$nummer_1 = 100.00;
$_Text1 = "Last Minute";
$jahr_alt1 = "2001";
```

Im Regelfall werden Variablen jeweils mit dem neuen, in der Reihenfolge des Scripts zuletzt zugewiesenen Wert überschrieben (changed on the fly). Passen Sie auf:

```
$beispiel = "hier dreht es sich um Variablen";
$beispiel = "wir behandeln Datentypen";
```

Geben Sie an dieser Stelle mit `echo` den Wert der Variablen `$beispiel` aus, wird der letzte Satz gedruckt bzw. angezeigt werden. Sie sehen dies in Bild 3.4.

Bild 3.4: Variablen werden überschrieben.

Variablennamen unterscheiden zwischen Groß- und Kleinschreibung (sie sind *case-sensitiv*). `$Nummer` und `$nummer` wären also nicht die gleiche Variable. Erfahrungsgemäß empfiehlt es sich, grundsätzlich alles kleinzuschreiben, dann müssen Sie sich nicht daran erinnern, ob Sie irgendeinen Teil des Variablennamens groß- oder kleingeschrieben haben, bzw. das Script mühselig danach durchforsten. Ein Name wie `$gartenHaus_nr3` wäre nicht besonders klug gewählt, denn mit Sicherheit wissen Sie irgendwann nicht mehr, dass Sie diese merkwürdige Groß- und Kleinschreibung verwendet haben. Außerdem ist es ratsam, mehr oder minder aussagekräftige Namen zu verwenden, die Ihnen auch später noch ihre Bedeutung verraten. Hüten Sie sich also vor kryptischen Abkürzungen.

Bei der Ausgabe des Inhalts einer Variablen mit dem Befehl `echo` schreiben Sie den Variablennamen nicht in Anführungszeichen (sie würden aber auch nicht stören). Wenn Sie aber einen Text durch mehrere Variablen ausgeben lassen, werden Anführungszeichen benötigt:

```
$stadt = "Berlin";
$code = 10666;
echo $stadt;
echo "$code $stadt";
```

Bild 3.5 zeigt den Code und die Ausgabe.

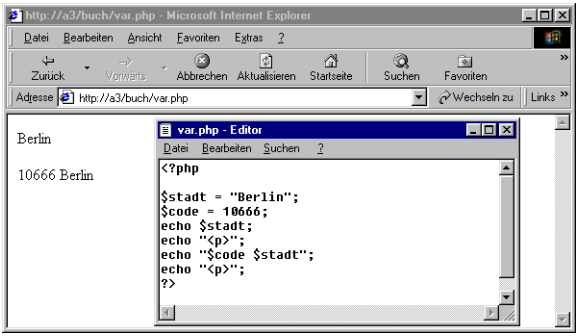

Bild 3.5: Die Inhalte von Variablen ausgeben

Wertezuweisung durch Variablen

Variablen können ihren Inhalt auch von anderen Variablen erhalten. Das sieht ganz ähnlich aus, aber die Anführungszeichen entfallen:

```
$testvar = $testvariable1;
```

Beachten Sie bitte: Wenn eine Variable ihren Inhalt von einer anderen Variablen erhalten hat, behält sie diesen Inhalt, auch wenn der Inhalt der Ursprungsvariablen geändert wird. Um dies noch einmal deutlich zu machen, schauen Sie sich das unten stehende kleine Scriptfragment an:

```
$variable1 = "Hans";
$variable2 = $Variable1;
//ausgegeben wird mit
echo $variable2;
Hans
//Verändern der Variablen $variable1
$variable1 = "Daniel";
//ausgegeben wird dennoch mit
echo $variable2;
Hans
```

Allerdings bietet PHP4 nun auch eine Wertezuweisung durch Referenzierung. In dem Fall zeigen beide Variablen auf die gleiche Speicherstelle mit dem Effekt, dass Änderungen der neuen Variablen auch die Ursprungsvariable ändern und umgekehrt. Für die Zuweisung per Referenz wird der Ausgangsvariablen ein `&` vorangestellt.

```
$tag = "Montag";
$tagneu = &$tag;
$tag = "Dienstag";
echo $tagneu;
```

```
//Ausgabe ist:
Dienstag
```

Bild 3.6 zeigt noch einmal beides zusammen, den Code und die Ausgabe.

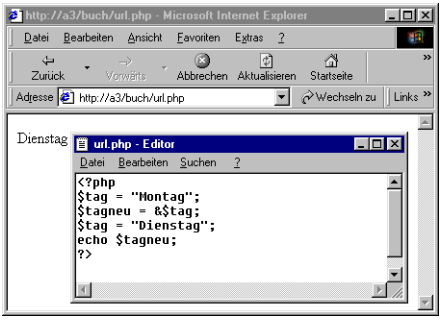

Bild 3.6: Zuweisung von Variablen per Referenzierung

Variablen erweitern/verketten

Variablen lassen sich erweitern/verlängern. Dies funktioniert durch den Verkettungsoperator, dargestellt durch einen Punkt. Bei dieser Methode verwenden Sie den- bzw. dieselben Variablennamen und bei der Zuweisung des nächsten Wertes einen Punkt vor dem Gleichheitszeichen.

```
$name = "der Vorname ist ";
$name .= "Daniel!;
```

In der Variablen steht nun *der Vorname ist Daniel!*. Geben Sie die Variable mit `echo $name;` aus, lesen Sie im Browser:

Bild 3.7: Die Ausgabe im Browser

Hätten wir den Punkt weggelassen, wäre der erste Wert der Variablen mit dem zweiten Wert (Daniel) überschrieben worden. Die Erweiterung einer Variablen funktioniert nicht nur einmal, sondern quasi so oft Sie möchten. Ergänzen Sie das Script um:

```
$name .= " Er ist 17";
```

wird bei der Ausgabe der Variablen `$name` auch dieser Satz zu lesen sein. Achten Sie bei Erweiterungen auf Leerstellen (die natürlich innerhalb der Anführungszeichen stehen müssen), ansonsten klebt der Text beieinander. In Bild 3.8 sehen Sie den Code und die Ausgabe.

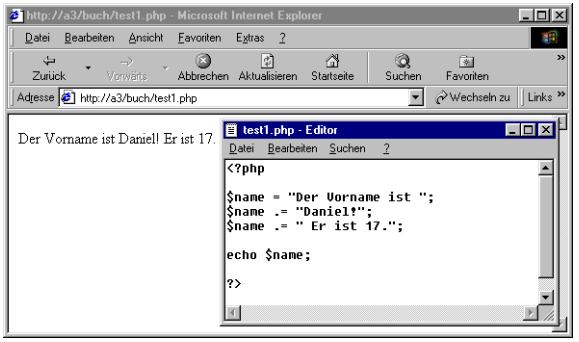

Bild 3.8: Variablen können erweitert werden.

Verketten

Etwas anders funktioniert die Verkettung. Dabei wird der Punkt verwendet und – technisch gesprochen – eine Zeichenkette ausgegeben, die aus dem rechten und linken Operand zusammengesetzt ist. Unabhängig vom Datentyp werden bei dem Einsatz des Punktes als Verkettungsoperator die Operanden als Zeichenkette behandelt. Die Syntax sieht folgendermaßen aus:

```
$nr1 = "aha,";
$nr2 =$nr1. " eine Verkettung";
```

Achten Sie hierbei auf die Leerstelle vor dem Wort *eine*, damit auch bei der Ausgabe dort eine Leerstelle ist, sonst hieße es: *aha, eine Verkettung*.

Bild 3.9 zeigt den Code und die Ausgabe im Browser.

Eine derartige Verkettung kann weiterentwickelt werden. Die Crux sind zum Teil die Leerstellen, achten Sie gut darauf, dass sie (als Zeichenkette) bei einer Verkettung auch in Anführungszeichen gesetzt werden müssen. Werfen Sie einen Blick auf Bild 3.10. Durch eine weitere Verkettung wird der Variablen `$nr4` ihr Wert zugewiesen.

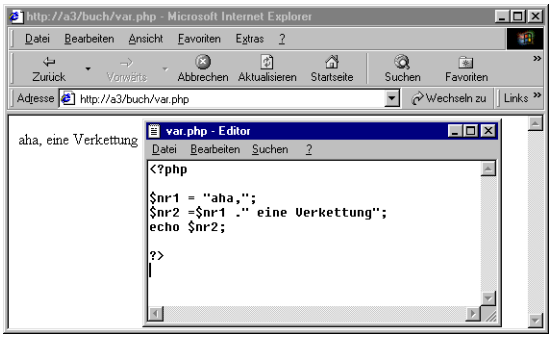

Bild 3.9: Variablen und Text können verkettet werden.

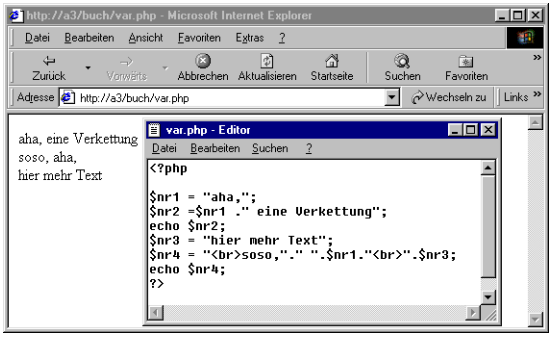

Bild 3.10: Noch mehr Elemente in der Verkettung

Hinweis

Statt die Werte von Variablen im Script festzulegen, können Sie auch übergeben werden, beispielsweise durch User-Eingaben in ein Formular. In dem Fall wird mit eindeutigen Namen gearbeitet, d.h. die Formularfelder erhalten einen eindeutigen Namen und dieser Name wird bei der Auswertung als Variablenname verwendet.

Die Existenz von Variablen prüfen

In nicht wenigen Situationen ist es wichtig zu prüfen, ob eine Variable mit einer leeren Zeichenkette gefüllt ist (oder mit 0) oder überhaupt schon zugewiesen wurde. Es gibt eine Funktion, die es Ihnen ermöglicht, Variablen auf ihre Existenz hin zu prüfen. Diese Funktion lautet:

```
isset($variable)
```

Also etwa: ist gesetzt. In der Klammer wird die Variable erwartet, deren Existenz Sie überprüfen möchten. Die Überprüfung gibt *wahr* zurück, wenn die Variable existiert. Umgekehrt kann man eine Variable auch wieder ungültig machen, was – wie Sie später in den Beispielen dieses Buches sehen werden – mitunter notwendig ist. Die entsprechende Funktion lautet

```
unset($variable)
```

Ob einer Variablen ein Wert zugewiesen wurde, lässt sich ebenfalls überprüfen:

```
empty($variable)
```

Hier erhalten Sie als Ergebnis der Prüfung *wahr*, wenn der Inhalt der Variablen `0` ist oder eine leere Zeichenkette.

Dynamische Variablen

Unter dynamischen Variablen versteht man die Möglichkeit, Variablennamen in Variablen zu speichern. Hier müssen Sie ein bisschen »um die Ecke« denken. Es verhält sich folgendermaßen: Die dynamische Variable nimmt den Wert der zuvor definierten gleichnamigen »normalen« Variablen als ihren Namen. Mit zwei vorangestellten Dollarzeichen kann man einer dynamischen Variablen einen Wert zuweisen. In einem Script kann das z.B. so aussehen:

```
$vari = "Vorname";
$$vari = "Daniel";
echo $Vorname;
```

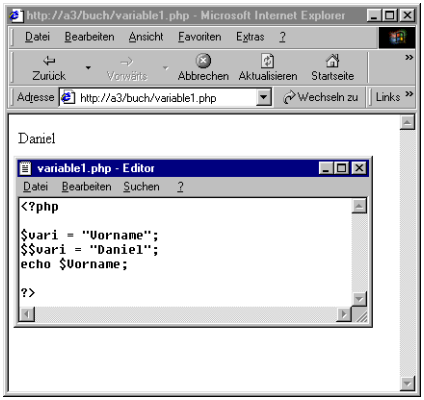

Bild 3.11: Die Wirkungsweise dynamischer Variablen

Datentypen

Wie Sie oben an den paar Beispielen bereits gesehen haben, können Sie Variablen Werte zuweisen, die unterschiedlichen Datentypen zuzuordnen sind. PHP erkennt den Datentyp bei der Zuweisung automatisch und behandelt den Wert entsprechend.

Als Grobunterscheidung kann man die Datentypen zunächst unterteilen in:

- Zeichenketten, auch als String bezeichnet
- Zahlen
- Arrays

Genau genommen gibt es auch noch die Typen Boolean (kennt nur *true* oder *false*) und Object.

Strings oder Zeichenketten

In dem obigen Beispiel

```
$vorname = "Daniel";
```

enthält die Variable eine Zeichenkette oder einen – dies ist die andere Bezeichnung für Zeichenketten – String. Es handelt sich also um eine Variable vom Typ String. Von einem String spricht man bei Eingaben, die aus Buchstaben oder aus einer Kombination von Buchstaben, Zahlen (mit denen keine Rechenoperation durchgeführt wird), Symbolen und Leerstellen bestehen und sich zwischen einfachen oder doppelten Anführungszeichen befinden. Strings wären beispielsweise:

```
"Hallo Welt"
"Hallo, $vorname"
"1999"
"Haus 59"
```

Im Zusammenhang mit dem Befehl `echo` spielt es übrigens eine Rolle, ob Sie für die Ausgabe doppelte oder einfache Anführungszeichen gebrauchen. Der Unterschied ist der, dass bei doppelten Anführungszeichen die Variablen mit ausgewertet werden. Würden Sie statt dessen `echo 'Hallo, $vorname';` schreiben, also einfache Anführungszeichen benutzen, würde `'Hallo, $vorname'`ausgegeben werden. Das ist meistens nicht Sinn der Sache.

Zeichenketten lassen sich in vieler Hinsicht manipulieren und bearbeiten. Ein paar Funktionen wurden bereits weiter oben erwähnt, im Lauf der nächsten Kapitel werden Sie weitere kennen lernen.

Zahlen

Bei dem Datentyp Zahl müssen Sie unterscheiden zwischen Integer oder Ganzzahl (eine Zahl ohne Nachkommastellen) und Double oder Fließkommazahl.

Als Integer können beispielsweise gelten:

- 10
- -10

Beispielwerte für den Typ Double wären:

- 10.50
- -10.50

Werte inkrementieren

Im Verlauf der nächsten Kapitel werden wir demonstrieren, wie Sie Zahlen manipulieren und ihnen z. B. ein bestimmtes Format zuweisen können. An dieser Stelle beschreiben wir noch, wie Sie den Wert einer Variablen jeweils um eins hochzählen, ein Vorgang, der als Inkrementieren bezeichnet wird und in einem Script sehr häufig vorkommt. Denkbar ist das folgende Vorgehen:

```
$zahl = 0;
$zahl = $zahl + 1;
```

Gefälliger ist die Kurzform, die auch meistens verwendet wird. Sie können einfach zwei Pluszeichen an den Wert hängen:

```
$zahl++
```

Wenn Sie diese Aktion in einem Script probieren möchten, lassen Sie die Zahlen jeweils ausgeben:

1. Öffnen Sie gegebenenfalls Ihren Editor und schreiben Sie das PHP-Startzeichen `<?php`.
2. Setzen Sie die Variable `$zahl=0;`.
3. Lassen Sie mit dem Befehl `echo` den Wert ausgeben.
4. Für bessere Lesbarkeit weisen Sie mit `echo "<br>";` einen Zeilenumbruch an.
5. Nun lassen Sie den eben gesetzten Wert `0` jeweils um eins hochzählen: `$zahl++;`
6. Beenden Sie PHP, speichern Sie das kleine Script als PHP-Datei und testen Sie es im Browser.

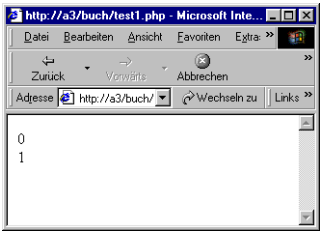

Bild 3.12: Zahlen mit ++ inkrementieren

Um das Gegenteil zu erreichen, also um zu dekrementieren, schreiben Sie statt der zwei Pluszeichen einfach zwei Minuszeichen, beispielsweise:

```
$zahlneu=10;
$zahlneu--;
```

Arithmetische Operatoren

Im Zusammenhang mit dem Datentyp *Zahlen* werden Sie in einem Script häufig arithmetische Operatoren verwenden. Dies sind auch in PHP die üblichen Operatoren, die Sie für grundlegende Berechnungen einsetzen:

Operatoren	Rechenart
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Tabelle 3.2: Die Zeichen für arithmetische Operatoren

Sie könnten also beispielsweise folgende Variable setzen:

```
$kosten = 12 * 50;
```

Wenn Sie dann schreiben würden:

```
echo "Sie zahlen im Jahr Euro $kosten";
```

hieße es auf der Webseite:

Sie zahlen im Jahr Euro 600.

Bild 3.13 zeigt beides, das Script im Editor und die Ausgabe im Browser.

Eine geschicktere Variante wäre es, vorher andere Variablen zu deklarieren und dann die Berechnung mit Hilfe dieser Variablen erfolgen zu lassen:

```
$monat = 12;
$beitrag = 50;
$kosten = $monat * $beitrag;
echo "Sie zahlen im Jahr Euro"." ".$kosten;
```

Der Vorteil: Ihr Script ist einfach flexibler. Ändert sich beispielsweise der Beitrag, brauchen Sie lediglich den Wert der einen Variablen zu ändern.

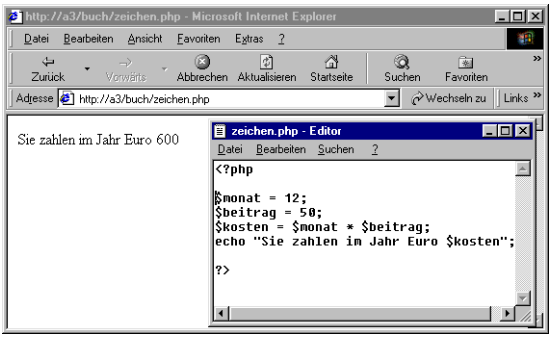

Bild 3.13: Für Berechnungen setzen Sie die üblichen Operatoren ein.

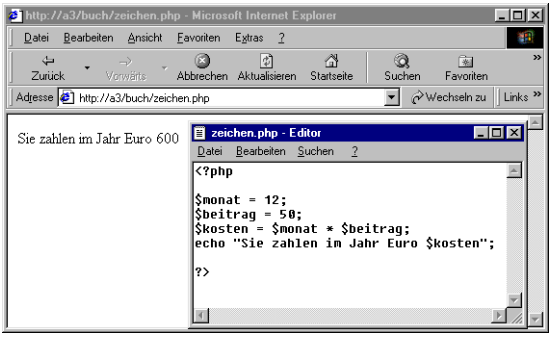

Bild 3.14: Eine einfache Berechnung – definieren Sie Variablen, um ein Script flexibel zu halten.

Präzedenz

PHP rechnet übrigens auf »normale« Art und Weise, d.h. folgt den üblichen mathematischen Regeln. Eine dieser Regeln besagt bekanntlich: Punktrechnung geht vor Strichrechnung. Diese Regel müssen Sie beachten, wenn Sie Variablen Werte zuweisen und dabei mehrere Operatoren benutzen. Unvermeidbar ist dann in bestimmten Fällen der Gebrauch von Klammern, die – wie Sie ja wissen – zuerst gerechnet werden, also eine so genannte Präzedenz erzwingen. Sie müssen also beispielsweise, je nachdem, wie die Berechnung aussehen soll, schreiben: `$kosten = (20 - 8)/4` oder aber `$kosten = 20 - (8/4)`.

Datentypen feststellen und festlegen

Um sicher zu wissen, welcher Datentyp intern von PHP genutzt wird, können Sie eine Funktion benutzen, die den jeweiligen Typ zurückgibt. Die Funktion lautet: `gettype()`.

In die Klammer schreiben Sie die Variable, deren Typ Sie sich anzeigen lassen möchten:

```
<?php
$datentyp=10;
echo gettype($datentyp);
echo "<br>";
$datentyp="Hallo";
echo gettype($datentyp);
?>
```

Listing 3.3: Ein kleines Script zur Rückgabe von Datentypen im Browser

Der Browser zeigt damit untereinander (wegen
) die Datentypen Integer und String.

Das Gegenstück zu `gettype()` ist die Funktion `settype()`. Damit kann der Typ einer Variablen geändert werden, was mitunter notwendig ist. Die Klammer erwartet als Argumente die zu ändernde Variable und – durch Komma getrennt – den gewünschten Datentyp. Angenommen, es gibt die Variable `$zahl = 4711` vom Datentyp Zahl. Um daraus einen String zu machen, würden Sie schreiben:

```
settype($zahl, string);
```

Mit der Funktion `gettype()` (und `echo`) können Sie überprüfen, ob die Konvertierung geklappt hat.

Arrays

Arrays sind Sonderformen von Variablen, denn in Arrays können beliebig viele Werte gespeichert werden. Sie stellen gewissermaßen eine Sammlung von Werten in einer Variablen dar. Arrays können Strings und Zahlen enthalten (oder andere Arrays). Die Namen von Arrays werden ebenfalls mit einem Dollarzeichen gebildet, gefolgt von einer eckigen Klammer. Ansonsten unterliegen die Namen der gleichen Schreibkonvention wie andere Variablennamen.

Um Arrays zu verstehen, wird häufig der Vergleich mit einer Tabelle herangezogen. Werfen Sie einen Blick auf die folgende kleine Tabelle:

Index oder Schlüssel	Wert
1	Hosen
2	Röcke
3	Kleider
4	Jacken
5	Pullover

Tabelle 3.3: Arrays sind strukturiert wie Tabellen. Im Beispiel werden Zahlen als Schlüssel verwendet.

Zu einem Array könnte man die Daten folgendermaßen zusammenfassen: Sie legen einen Array-Namen fest, beispielsweise `$kleidung`. Der Indexwert wird einer eckigen Klammer übergeben und dann weisen Sie der Array-Liste die Werte zu:

```
$kleidung[1] = "Hosen";
$kleidung[2] = "Röcke";
$kleidung[3] = "Kleider";
$kleidung[4] = "Jacken";
$kleidung[5] = "Pullover";
```

Listing 3.4: So erstellen Sie ein Array.

Eine andere Schreibweise – diese ist eher die klassische – zum Erstellen einer Array-Liste funktioniert auch. Die Zuweisung der Werte erfolgt über die Funktion `array()`:

```
$kleidung = array (1=>"Hosen", 2=>"Röcke", etc);
```

Dies erspart zwar die Wiederholung des Wortes *array*, ist aber nach unserem Verständnis weniger übersichtlich. Achten Sie auf die Syntax: Sie schreiben den Schlüssel für den Wert und dann durch einen Pfeil getrennt den Wert selbst.

Wenn Sie, nachdem Sie ein Array erstellt haben, den `echo`-Befehl verwenden und lediglich schreiben:

```
echo $kleidung;
```

ist das Ergebnis nicht besonders spannend (Bild 3.15), denn Sie erhalten lediglich das Wort *Array* als Rückgabewert. Immerhin wissen Sie dadurch, dass das Erstellen des Arrays geklappt hat. Um die vollständige Array-Liste angezeigt zu bekommen, müssen Sie das Array durchlaufen. Dies wird weiter unten besprochen. Ansonsten können Sie auf einzelne Elemente eines Arrays zugreifen. Dies geschieht, indem Sie den entsprechenden Indexwert ansprechen, also beispielsweise schreiben:

```
echo $kleidung[1];
```

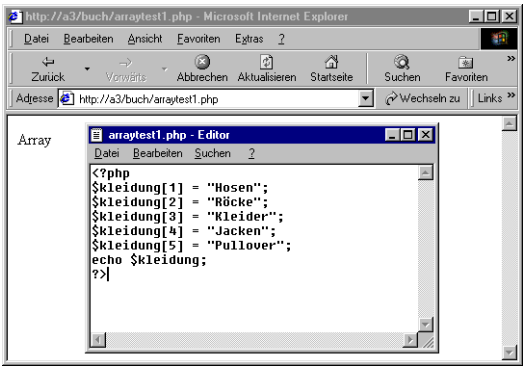

Bild 3.15: Eine Array-Liste erstellen – die Ausgabe mit echo im Browser

Sie müssen den Elementen keine Indexwerte zuweisen. PHP4 sorgt auch selbst für die Indizierung, wobei eine Besonderheit zu beachten ist: Diese Indizierung beginnt bei Null und nicht bei Eins. Im Array

```
$kleidung[] = "Hosen";
$kleidung[] = "Röcke";
$kleidung[] = "Kleider";
```

wäre das nullte Element *Hosen*. Sie müssten sich also auf `$kleidung[0]` beziehen, wenn PHP auf das Element mit dem Wert *Hosen* zugreifen soll. Schreiben Sie ein Script wie in Bild 3.16, wird im Browser ausgegeben: *wir haben schöne Kleider*.

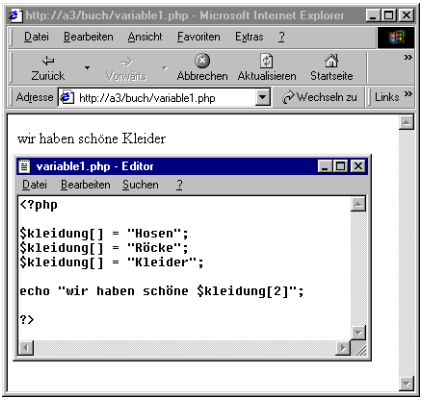

Bild 3.16: Auf Elemente eines Arrays zugreifen

Wenn Sie das Array über `array()` festlegen, schreiben Sie – sofern Sie PHP die Indizierung überlassen – einfach:

```
$kleidung = array("Hosen", "Röcke", "Kleider");
```

Wollen Sie auf *Kleider* zugreifen, brauchen Sie den Indexwert [2].

Assoziative Arrays

Der Indexwert muss keine Zahl sein, sondern kann auch eine Zeichenkette sein. Man spricht dann von assoziativen Arrays. Es kann auch vorkommen, dass Sie mitunter den Begriff Hash-Arrays lesen. So könnten Sie auch folgenden Code schreiben:

```
$kleidung[lieferant1] = "Hosen";
$kleidung[lieferant2] = "Röcke";
```

Dabei müssen Sie wieder daran denken, dass PHP auch bei den Schlüsseln, die Sie als String in die eckige Klammer schreiben, sensibel auf Groß- und Kleinschreibung reagiert. Die Worte (bzw. Strings) müssen nicht in Anführungszeichen gesetzt werden, wenn der Schlüssel nur ein Wort enthält. Die andere Schreibweise ist:

```
$kleidung = array (lieferant1 =>"Hosen", lieferant2 =>"Röcke",
lieferant2=> "Kleider");
```

Arrays ergänzen

Arrays lassen sich auch mühelos ergänzen. Sie hängen der vorhandenen Liste einfach die gewünschten Elemente an. Wenn Sie die Elemente nicht spezifizieren, wird jedes Element mit der nächsten logischen Zahl indiziert. Ein neuer Wert in der Array-Liste `$kleidung`, beispielsweise `$kleidung[] = "Socken";`, wäre automatisch das dritte Element.

Der Vorteil von Arrays liegt neben der einfachen Eingabe (d.h. ein Variablenname statt ganz viele für jeden unterschiedlichen Wert) in der bequemen Bearbeitung der einzelnen Elemente. Dies werden wir an relativ einfachen Beispielen in den Kapiteln weiter hinten demonstrieren.

Der Umgang mit Arrays – Beispiele

Probieren Sie am besten ein einfaches Script mit einem Array aus.

Öffnen Sie Ihren Editor und beginnen Sie eine neue Datei. Nach dem üblichen HTML-Header erstellen Sie eine Array-Liste. Sie können dazu Listing 3.5 benutzen. Beachten Sie die beiden Zeilen zwischen den Zeichen /* und */. Es handelt sich um einen Kommentar, der nicht ausgeführt wird. Er zeigt die alternativen Schreibweisen, das Array zu erstellen.

```
<html>
<head>
<title>comment</title>'
```

```
</head> <body>
<?php
$Kurse = array("Montag"=>"Weben", "Dienstag"=>"Stricken", "Mittwoch"=>"Tanzen",
"Donnerstag"=>"Malen");
/*
$Kurse[Montag]="Weben";
$Kurse[Dienstag]="Stricken";
*/
?>
</body> </html>
```

Listing 3.5: Zeigt ein Script mit einer Array-Liste, erstellt mit der Funktion array().

Um einen spezifischen Wert eines Arrays direkt anzusprechen, müssen Sie sich, wie oben bereits erwähnt, auf den Schlüssel beziehen. Schreiben Sie beispielsweise die folgende Zeile in das Script:

```
echo "Neu ist unser Kurs in"." ".$Kurse[Montag]."<p>";
```

Wir wählen hier übrigens die »saubere« Schreibweise mit Verkettung (die Punkte). Wir wollen Ihnen aber nicht verschweigen, dass auch die einfachere Zeile:

```
echo "Neu ist unser Kurs in $Kurse[Montag] <p>";
```

funktionieren würde. Es gibt aber auch Fälle, in denen diese Schreibweise nicht das Gelbe vom Ei ist, und deswegen ist es eigentlich ratsam, sich an die Syntax mit Verkettung zu gewöhnen.

Speichern Sie das Dokument als PHP-Datei und testen Sie es in Ihrem Browser.

Arrays auswerten

PHP bietet eine Reihe von Möglichkeiten, eine Array-Liste auszuwerten. Wie Sie auf eines der Elemente zugreifen, haben wir bereits besprochen. Sie können aber beispielsweise auch die Anzahl der Elemente in einem Array feststellen. Das geht recht einfach. Der Befehl, den Sie dafür einsetzen, ist die Funktion `count()`.

Schreiben Sie unter die letzte Zeile des Arrays:

```
$allekurse = count($Kurse);
```

Damit legen Sie eine neue Variable fest (`$allekurse`), die die Anzahl der Elemente ausgeben wird.

Dann folgt der `echo`-Befehl. Sie benutzen also für die Ausgabe die eben definierte Variable, denn dieser wurde der Wert der gezählten Elemente zugewiesen:

```
echo $allekurse;
```

Betrachten Sie das Ergebnis im Browser. Sie müssten die Anzahl der Kurse angezeigt bekommen.

Die Existenz von Elementen prüfen

Mitunter kommt es vor, dass Sie prüfen möchten, ob ein bestimmtes Element in einem Array vorkommt. Statt das Array durchlaufen zu müssen, gibt es nun (seit PHP4) die Funktion `in_array()`. Als Argumente schreiben Sie das zu suchende Element in die Klammer und das zu durchsuchende Array. Sie verwenden dabei einen einfachen `if`-Befehl. Das Ganze könnte – bezogen auf unser obiges Beispiel-Array `$kleidung` – so aussehen wie in Bild 3.17:

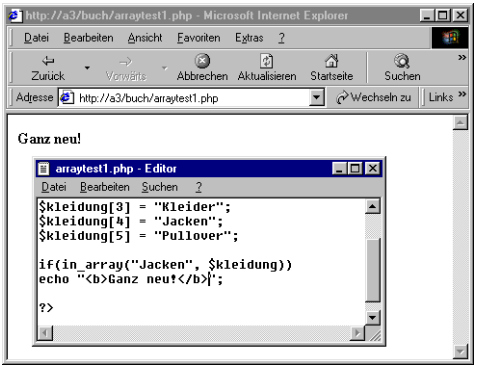

Bild 3.17: Mit in_array prüfen Sie, ob ein Element existiert.

Ein neues Array aus zwei Arrays bilden

Neu in PHP4 ist die Möglichkeit, aus zwei Arrays ein neues Array zu bilden. Dies ist mit der Funktion `merge()` zu erreichen. *Merge* heißt übrigens so viel wie »verschmelzen«, die Bezeichnung passt also recht gut. Nehmen Sie an, Sie haben die folgende numerisch indizierte Array-Liste in einem Script:

```
$reisen[] = "Holland";
$reisen[] = "Schweiz";
$reisen[] = "Frankreich";
```

Außerdem gibt es eine zweite Liste:

```
$reisen1[] = "Schweden";
$reisen1[] = "Daenemark";
$reisen1[] = "Finnland";
```

Aus diesen beiden Arrays können Sie eine neue Array-Liste bilden. Dazu geben Sie einen beliebigen Array-Namen ein, als Argumente schreiben Sie den Namen des ersten Arrays und den Namen des zweiten Arrays in die Klammer:

```
$reisenneu = array_merge($reisen, $reisen1);
```

Um zu sehen, ob das funktioniert hat, lassen Sie die Elemente zählen und den Rückgabewert ausgeben. Ergänzen Sie also das Script um die folgenden Zeilen:

```
$allereisen = count($reisenneu);
echo $allereisen." Reisen im Angebot <br>";
```

Der Rückgabewert müsste in dem Fall *6* ergeben, ergänzt durch die Worte *Reisen im Angebot*. Achten Sie wieder auf die Leerstelle vor *Reisen*.

Arrays sortieren

Da Arrays ja Listen mit mehreren Elementen sind, eignen sie sich auch hervorragend zum Sortieren. Bestehen die Elemente aus Strings, kann man alphabetisch sortieren, bestehen sie aus Zahlen, numerisch. Überdies lassen sich entweder die Elemente (Werte) oder die Schlüssel in eine Reihenfolge bringen.

Die Funktion zum Sortieren der Elemente lautet `sort()` oder `rsort()`.

Die zweite Funktion erzeugt eine Sortierung von Z nach A.

Um die Werte der Array-Liste `$reisenneu` (also die Länder) zu alphabetisieren, ergänzen Sie das Script um die folgende Zeile:

```
sort($reisenneu);
```

Das nützt natürlich noch wenig, so lange Sie das Ergebnis nicht im Browser betrachten können. Folglich müssen Sie auf alle Elemente der Liste zugreifen, indem Sie die Liste durchlaufen.

Eine Array-Liste durchlaufen

Eine Möglichkeit zum Durchlaufen eines Arrays bietet der Befehl `foreach`, mit dem jeder Wert des Arrays vorübergehend einer Variablen zugeordnet wird. Die Anweisung `foreach` ist eine so genannte Schleife (die wir im nächsten Abschnitt ausführlicher erklären). Die allgemeine Syntax ist die folgende:

```
foreach(arrayname as $temp)
{
Anweisung;
}
```

In der Variablen `$temp` wird jedes Element temporär gespeichert. Der Name der Variablen wird durch den Befehl `as` angegeben. Damit die Namen untereinander erscheinen, wird ein `<br>` (achten Sie auf die Anführungszeichen) in die Zeile geschrieben.

Daher schreiben Sie in das Script:

```
foreach($reisenneu as $ziel)
{
echo $ziel."<br>";
}
```

Speichern Sie das Script ab und betrachten Sie das Ergebnis im Browser. Sie müssten nun eine alphabetisierte Liste der Länder erhalten. Das komplette Script dazu sieht aus wie in Bild 3.19.

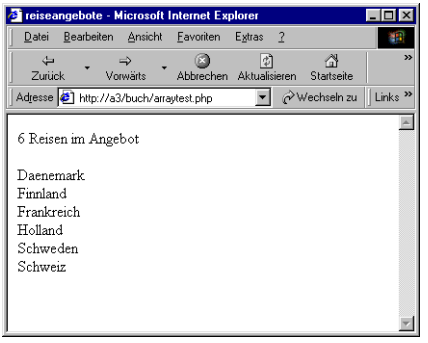

Bild 3.18: Eine alphabetisierte Liste wird ausgegeben.

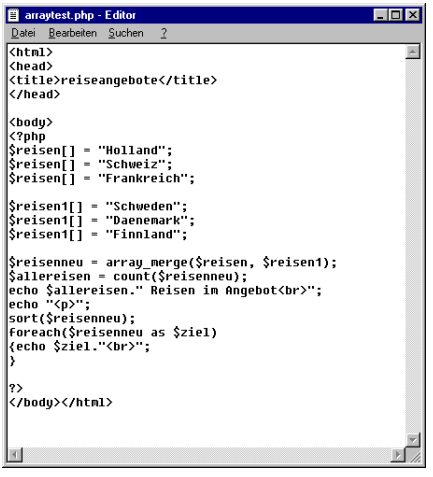

Bild 3.19: Das Script zum Verbinden von zwei Array-Listen, zum Sortieren der Elemente, zum Zählen der Elemente und zur Ausgabe der Liste.

Assoziative Arrays durchlaufen

Betrachten wir auch noch die Möglichkeit, assoziative Arrays zu durchlaufen, also die, in denen die Schlüssel aus Zeichenketten bestehen. Nehmen wir eine neue Array-Liste an:

```
$reisen = array(Norden=>"Finnland", Westen=>"Irland", Osten=>"Polen");
```

Um die Schlüssel und Werte anzuzeigen, schreiben Sie:

```
foreach($reisen as $key=>$wert)
{
echo "$key = $wert<br>";
}
```

Die Variable `$key` enthält temporär jeden Schlüssel des Arrays und die Variablen `$wert` jeden Wert. Mit `foreach` wird das Array Element für Element durchlaufen.

Als Ergebnis erhalten Sie eine Ausgabe, die in Bild 3.20 zu sehen ist.

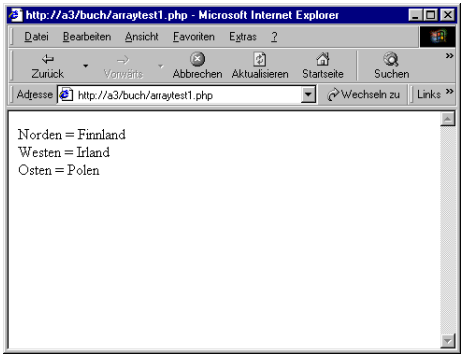

Bild 3.20: Die Ausgabe der Elemente eines assoziativen Arrays im Browser

Varianten der sort-Funktionen

Mit der Funktion `sort()` werden die Elemente zusammen mit den Schlüsseln neu sortiert. Sollen trotz der Neusortierung die Schlüssel beibehalten werden, würden Sie stattdessen die Funktion `asort()` benutzen. Um nach den Indexwerten zu sortieren und zwar so, dass Sie die entsprechenden Werte behalten, setzen Sie die Funktion `ksort()` ein. Die Funktion `krsort()` sortiert auf Basis der Schlüssel von hinten nach vorn. Erwähnen wir zu guter Letzt die Funktion `shuffle()`. Damit werden die Elemente einer Array-Liste nach einem Zufallsprinzip in eine neue Reihenfolge gebracht.

Mehrdimensionale Arrays

Auch wenn mehrdimensionale Arrays nicht unbedingt zu den (notwendigsten!) Grundlagen gehören, wollen wir sie kurz ansprechen. Bei mehrdimensionalen Arrays nutzen Sie den `array`-Befehl innerhalb eines übergeordneten Arrays. Dadurch können Sie ein Array erstellen, das sogar noch mehr Informationen enthält als ein »normales« Array. Der Trick ist folgender: an Stelle von Strings oder Zahlen dienen andere Arrays als Werte. Wenn Sie also beispielsweise zwei Arrays haben mit den Namen `$inlandreisen` und `$auslandreisen`, könnte die Syntax für das mehrdimensionale Array so aussehen wie in der dritten Zeile:

```
$inlandreisen=array("Hamburg", "Franfurt", "Berlin");
$auslandreisen=array("Finnland", "Island", "Schweden");
$liste1=array("inland"=>$inlandreisen, "ausland"=>$auslandreisen);
```

Um ein Element eines mehrdimensionalen Arrays anzusprechen, müssen Sie (sofern Sie selbst keine Schlüssel gesetzt haben) die Reihe mitzählen. Die Schreibweise ist folgende:

```
$liste1["inland"][1]
```

Dies wäre also in unserem Beispiel Frankfurt, das zweite Element im Ursprungs-Array `$inlandreisen`.

Vordefinierte Funktionen

PHP kennt eine Fülle von vordefinierten Funktionen (im Gegensatz zu selbst definierten, die wir an dieser Stelle noch nicht erklären möchten). Durch Funktionen werden bestimmte Anweisungen, die PHP ausführen soll, übersichtlich zu Blöcken zusammengefasst.

Auch der eben eingesetzte Befehl `echo` ist eine Art Funktion, er bildet aber hinsichtlich der Schreibweise eine Ausnahme. Funktionen folgen normalerweise der Syntax:

```
Funktionsbezeichnung()
```

Sie brauchen also eine Klammer; diese Klammern nehmen (im Regelfall) ein oder mehrere Argumente oder Parameter auf. Sind es mehrere, werden die Argumente, also die Daten, die der Funktion übergeben werden, durch Kommata abgetrennt. Nur bei dem Befehl `echo` reichen Anführungszeichen, die Klammer ist optional.

Da Funktionen für die unterschiedlichsten Zwecke eingesetzt werden, werden sie systematisch in Kategorien eingeteilt. Dies erleichtert den Umgang mit ihnen, da es vor allem dem Einsteiger unmöglich ist, alle Funktionen auf Anhieb parat zu haben. Sind Sie also auf der Suche nach einer Funktion, die eine bestimmte Aufgabe durchführen soll, können Sie in Referenzbüchern oder Manuals in der entsprechenden Kategorie nachschlagen und müssen nicht eine unsortierte Liste von Hunderten von Funktionen durchsehen.

Mit Funktionen aus folgenden Kategorien werden Sie zunächst (vermutlich) am häufigsten konfrontiert:

- Dateisystem-Funktionen
- Datums- und Zeitfunktionen
- Mail-Funktionen
- Mathematische Funktionen
- MySQL-Funktionen
- String-Funktionen

Es gibt eine Menge weiterer Kategorien; wir greifen hier – wie gesagt – die heraus, die besonders gängige und häufig eingesetzte Funktionen enthalten. In den folgenden Abschnitten stellen wir kurz aus den erwähnten Kategorien einige Funktionen vor.

Zeichenketten(String)-Funktionen

Eine String-Funktion (genau genommen ist dies keine Funktion, sondern ein Sprachkonstrukt) haben Sie bereits kennen gelernt:

```
echo
```

Mit `echo` werden alle Zeichenketten ausgegeben (Sie erfahren mehr über Zeichenketten in Abschnitt *Datentypen*). Statt `echo` können Sie wahlweise auch

```
print
```

benutzen. Das ist reine Geschmackssache.

Der Befehl `trim()` entfernt überflüssige Zeichen am Anfang und Ende einer Zeichenkette (Achtung, nicht in der Mitte!). Dieser Befehl wird häufig eingesetzt, wenn die Daten, die von PHP verarbeitet wurden, von einem Formular übertragen wurden. Es passiert Usern – bewusst oder unbewusst – recht häufig, dass bei der Eingabe von Daten überflüssige Leerstellen hinzugefügt werden. Als weitsichtiger Programmierer sollten Sie im PHP-Code dafür sorgen, dass diese Leerstellen entfernt werden. So kann der Code aussehen, wenn es beispielsweise ein Formularfeld `$nachname` gibt, das die Quelle der Daten ist:

```
$nachname =trim($nachname);
```

Eine Variante ist:

```
ltrim()
```

mit der die Leerstellen am Anfang einer Zeichenkette entfernt werden.

```
strlen()
```

gibt die Länge einer Zeichenkette aus. Schreiben Sie zum Beispiel:

```
<?php
$testtext = "was haben wir denn da?"
$length = strlen($testtext);
echo $length;
?>
```

Bild 3.21 zeigt den Code und die Ausgabe im Browser.

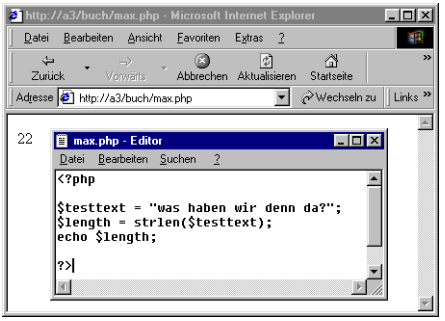

Bild 3.21: Mit strlen() wird die Länge eines Strings ausgegeben

Mail-Funktionen

`mail()`

Mit dieser Funktion können Sie eine E-Mail im Text- oder HTML-Format an eine oder auch mehrere Empfänger verschicken. In der Klammer werden als Argumente erwartet: Empfänger, der Betreff und die Message.

```
mail("an@t-online.de", "testmail", "HalliHallo");
```

Um beim Verschicken einer Mail einen vernünftigen Absender zu erhalten, ist es empfehlenswert, auch einen optionalen Parameter, mit dem zusätzliche Header-Informationen angegeben werden, zu verwenden.

Hinweis

Den Header einer E-Mail-Nachricht sehen Sie in einem normalen E-Mail-Programm nicht. Mit diesem Header werden Informationen über die E-Mail übermittelt. Der Header enthält z.B. Informationen über den Weg, den die Nachricht durch das Internet genommen hat. Einige Informationen des Headers werden von dem E-Mail-Programm ausgelesen und Ihnen angezeigt. Dies sind u.a. der Absender und die Priorität. Die Header-Informationen setzen sich aus einem Schlüsselwort und dem Wert dieses Schlüssels zusammen. So ist im obigen Beispiel `from` das Schlüsselwort und die E-Mail-Adresse der Wert.

Sie ergänzen die Funktion durch einen Absender, indem Sie in die Klammer `"from: xyz@yxz.de"` schreiben.

Datum- und Zeitfunktionen

`date()`

Diese Funktion gibt eine Zeitangabe formatiert zurück. Je nach der Zeitangabe, die Sie wünschen, setzen Sie unterschiedliche Platzhalter ein, z.B. `d` für Tag (in der Schreibweise `01` etc.), `M` für eine dreistellige Monatsangabe, `Y` für eine vierstellige Jahreszahl: `date("d M Y");` gibt also das aktuelle Datum in folgender Schreibweise zurück: `tt mmm yyyy`.

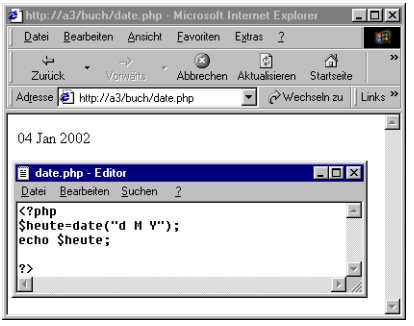

Bild 3.22: Die Verwendung der Datumsfunktion und die Ausgabe im Browser

Einen Überblick über die Platzhalter finden Sie in der Tabelle 3.4:

Platzhalter	Ausgabe
g	Stunden im Format: 1 – 12
G	Stunden im Format: 0 – 23
h	Stunde im Format: 01 – 12
H	Stunde im Format: 00 – 23
i	Minuten im Format: 00 – 59
j	Tag des Monats
d	Tag der Woche mit 2 Stellen und führenden Nullen
D	Tag der Woche in dreistelliger Abkürzung (Mon)
j	Tag des Monats ohne führende Null

Tabelle 3.4: Ein Überblick über die Platzhalter für die Datumsformate

Platzhalter	Ausgabe
W	Numerische Darstellung des Wochentags (0 = Sonntag)
m	Monat im Format: 01 – 12
M	Monat als dreistellige Abkürzung
n	Monat ohne führende Null
Y	Vierstellige Jahreszahl
y	Zweistellige Jahreszahl
a	am oder pm

Tabelle 3.4: Ein Überblick über die Platzhalter für die Datumsformate (Forts.)

`checkdate()`

Damit können Sie ein Datum auf Gültigkeit überprüfen lassen. Das Ergebnis der Prüfung kann `true` (wahr) oder `false` (falsch) sein. In die Klammer schreiben Sie numerisch den Monat, den Tag und das Jahr:

```
$checkdatum =checkdate(12, 24, 2001);
```

`time()`

Diese Funktion gibt die aktuelle Zeit des Servers zurück. Bei `time()` bleibt die Klammer leer. Wenn Sie `echo time()` schreiben, erhalten Sie den aktuellen Unix-Zeitstempel zurück. Dieser enthält die Anzahl der Sekunden seit Beginn der Unix-Epoche, also seit dem 01.01.1970, 00.00.00 Uhr. Ihr Browser gibt also eine ziemlich krause Zahl aus.

Mathematische Funktionen

`abs()`

Diese Funktion gibt von einer bestimmten Zahl den Absolutwert zurück. Ist die Zahl ohnehin ein Absolutwert, passiert nichts.

`max()`

Mit `max` wird der numerisch größte Eingabewert zurückgegeben. Es müssen mindestens zwei Parameter übergeben werden. Werfen Sie einen Blick auf das Bild 3.23.

`min()`

Mit `min` wird der numerisch kleinste Eingabewert zurückgegeben. Es müssen mindestens zwei Parameter übergeben werden.

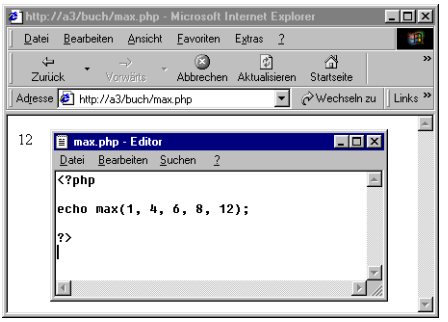

Bild 3.23: Die Funktion max() gibt den größten Wert zurück.

Dateisystem-Funktionen

`fopen()`

Diese Funktion öffnet eine Datei oder einen URL. Wenn die Datei noch nicht existiert, wird sie mit `fopen` erzeugt. In der Klammer steht der Name der Datei und der Modus (Dateizeiger).

`fgets()`

Mit dieser Funktion lesen Sie eine Zeile einer Datei aus, und zwar von der Position des Dateizeigers.

`fputs()`

Mit `fputs` schreibt man Daten an die Position des Dateizeigers. Sie können stattdessen auch `fwrite` benutzen.

`filesize()`

Diese Funktion liefert die Größe einer Datei.

MySQL-Funktionen

`mysql_connect()`

Diese Funktion setzen Sie ein, um eine Verbindung zum Datenbank-Server herzustellen. Als Argumente setzen Sie in die Klammer *hostname*, *Benutzername* und *Kennwort*.

`mysql_create_db()`

Diese Funktion erzeugt (bei entsprechender Berechtigung) eine neue Datenbank auf dem Server.

`mysql_close()`

Diese Funktion schließt eine Verbindung zum Datenbank-Server.

`mysql_db_query()`

Diese Funktion sendet ein SQL-Statement an die Datenbank, die dieses abarbeitet.

> **Hinweis**
>
> Die Funktionen des Dateisystems und die MySQL-Funktionen werden an dieser Stelle nur kurz erwähnt, aber nicht weiter erläutert, da sie im konkreten Kontext wesentlich verständlicher werden.

Kontrollstrukturen

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";
}
?>
```

Listing 3.6: Ein Scriptfragment mit einer if-Bedingung

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).

Achtung

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 3.5:

Symbol	Bedeutung
==	Gleichheit
!=	Ungleichheit
>	Größer als
<	Kleiner als
<=	Größer als oder gleich
<=	Kleiner als oder gleich

Tabelle 3.5: Vergleichsoperatoren

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!";
}
?>
```

Listing 3.7: Das Script mit einer elseif-Klausel: Je nach Wochentag erscheint ein anderer Text.

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 *Kapitel 5*, in dem beispielhaft ein Euroumrechner programmiert wird, verwenden wir die switch-Bedingung, sodass Sie dort sehen können, wie dieser Befehl konkret eingesetzt wird.

Schleifen

Schleifen sind ein Herzstück der Programmsteuerung. Mit ihnen wird angegeben, wie oft ein Programmstück ausgeführt bzw. wiederholt werden soll. Die erste Schleife, die Sie in PHP verwenden – die `while`-Anweisung – bewirkt, dass eine Anweisung wiederholt wird, so lange die Bedingung erfüllt ist, der Ausdruck also `true` zurückgibt. Der Ausdruck wird vor jedem Schleifendurchlauf geprüft. Die Struktur der while-Schleife ist vergleichbar mit der if-Anweisung.

Die Verwendung der while-Schleife

Die while-Schleife wird so lange ausgeführt, wie eine bestimmte Bedingung zutrifft. In der Regel arbeitet man bei der while-Schleife mit einem internen Wert, der zunächst in eine Variable geschrieben wird. Das schafft die Bedingung für die Schleife. Generell wird die while-Schleife folgendermaßen geschrieben:

```
while (Bedingung) {Anweisung;}
```

Probieren Sie die while-Anweisung anhand eines einfachen Scripts, indem Sie ein neues Dokument beginnen und die üblichen HTML-Tags eingeben.

Fügen Sie in den Body-Container die PHP-Tags `<?php` und `?>` ein.

Schreiben Sie dann zwischen die PHP-Tags:

```
$i=1
while($i<10)
{
echo ($i * $i). "<br>";
$i++;
}
```

Damit setzen Sie die Variable `$i` und weisen ihr den Wert 1 zu. Dann sagen Sie in der while-Schleife: So lange `$i` kleiner als 10 ist, soll der Wert mit sich selbst multipliziert werden; mit `echo` werden die Ergebnisse der Anweisung ausgegeben. Entscheidend ist die Anweisung `$i++`. Damit wird der Wert jeweils um eins hochgezählt oder inkrementiert. Der Prozess setzt sich fort, bis `$i` größer als 10 ist. An diesem Punkt endet die Schleife.

Speichern Sie das Dokument als PHP-Datei und rufen Sie es im Browser auf.

Die Verwendung der for-Anweisung

Während die while-Anweisung durchlaufen wird, bis der Ausdruck `false` ergibt, wird die for-Anweisung eingesetzt, um eine Anweisung so oft auszuführen, wie man es im Code angibt. Sie mögen die for-Anweisung von der Syntax her etwas komplizierter finden, dafür ist sie aber in gewisser Weise in der Formulierung der Bedingung einfacher als die while-Anweisung und deswegen auch wieder leichter zu handhaben. Sie werden feststellen, dass

in manchen Fällen die while-Anweisung besser passt und in manchen Fällen die for-Anweisung. Die generelle Syntax ist die folgende:

```
for (Variablenzuweisung/Schleifenbedingung1; Ausdruck2;Ausdruck3)
{Anweisung;
}
```

Sie sehen, die Ausdrücke in den Klammern werden durch Semikolon getrennt. Der erste Ausdruck initiiert in der Regel die Variable, der zweite Ausdruck prüft, ob die Anweisungen ausgeführt werden oder nicht, der dritte Ausdruck wird immer dann ausgeführt, wenn die Prüfung der Bedingung `true` ergeben hat. In geschweiften Klammern folgt dann wie üblich der Block mit der/den Anweisungen.

Wir haben oben bereits über Arrays gesprochen. Eine von vielen Möglichkeiten, eine for-Anweisung zu benutzen, ist beispielsweise die Anweisung, alle Elemente eines Arrays ausgeben zu lassen. Probieren Sie die for-Anweisung in diesem Zusammenhang:

1. Öffnen Sie in Ihrem Texteditor ein neues Dokument.
2. Geben Sie die Standard-HTML-Tags sowie die PHP-Tags ein.
3. Erstellen Sie ein beliebiges Array oder orientieren Sie sich am Listing 3.8. Hier heißt die Array-Liste `$fotos`.
4. Nun schreiben Sie die `for`-Schleife:

```
for ($n = 0; $n<count($fotos); $n++)
{
echo "$fotos[$n]";
}
?>
```

Insgesamt sieht das Script folgendermaßen aus:

```
<html><head>
<title>Formulart-test</title>
</head><body>
<?PHP
$fotos[]="Sommer";
$fotos[]="Herbst";
$fotos[]="Winter";
for ($n=0; $n<count($fotos); $n++)
{
echo "$fotos[$n]<br>";
}
?>
</body></html>
```

Listing 3.8: Das Script mit einer for-Schleife, um auf die Elemente eines Arrays zuzugreifen

Speichern Sie das Script und öffnen Sie es im Browser. Das Ergebnis müsste dem Bild 3.24 entsprechen.

Bild 3.24: Die Ausgabe der Array-Elemente im Browser

Erläuterung der for-Schleife

Um das Verständnis für for-Schleifen etwas zu vertiefen, werden wir das Script aus Listing 3.8 noch einmal genauer betrachten. Zunächst wird das Array `$fotos` erstellt. Anschließend wird die Schleife durchlaufen.

Mit `for ($n=0; $n<count($fotos); $n++)` sagen Sie PHP Folgendes: Setze die Variable `$n=0` für den ersten Durchlauf der Schleife. Prüfe bei jeden Schleifendurchlauf die Bedingung `$n<count($fotos)` – also ob `$n` kleiner ist als die Anzahl der Elemente im Array `$fotos`. Im Beispiel sind dies 3 Elemente. Mit `$n++` wird `$n` nach jedem Schleifendurchlauf um eins erhöht. Beachten Sie, dass diese Argumente der for-Schleife mit Semikolon getrennt werden. Wenn die formulierte Bedingung erfüllt ist (wahr), wird der Anweisungsblock durchlaufen: `{echo "$fotos[$n]<br>";}`.

Beim ersten Aufrufen der Schleife wird `$n==` gesetzt, sodass die Bedingung erfüllt ist (`0<3`). Der Anweisungsblock wird ausgeführt – also `{echo "$fotos[0]<br>";}` – und somit das nullte Element des Arrays `$fotos` ausgegeben. Nachdem der Anweisungsblock abgearbeitet wurde, wird mit `$n++ $n` um eins erhöht – `$n` ist dann also 1. Anschließend wird die Bedingung erneut geprüft. Sie ist wieder erfüllt, da `1<3`, somit wird der Anweisungsblock erneut ausgeführt: `{echo "$fotos[1]<br>";}`. Wieder wird mit `$n++ $n` um eins erhöht und anschließend die Bedingung getestet. Dieser Vorgang wiederholt sich so lange, bis die Bedingung nicht mehr erfüllt ist – also `$n` größer oder gleich `count($fotos)`ist. In diesem Fall werden die Befehle ausgeführt, die Sie im Script nach der Schleife angegeben haben.

Achten Sie darauf, die Schleifenbedingung so zu formulieren, dass sie auch irgendwann nicht mehr erfüllt wird. Eine for-Schleife in der folgenden Form würde unendlich oft durchlaufen werden:

```
for($n=0; $n>0; $n++).
```

Sie können den Schleifenzähler auch innerhalb des Anweisungsblocks der Schleife verändern. Das folgende Scriptfragment gibt nur jedes zweite Element eines Arrays aus, da der Zähler in der Schleife auch noch einmal erhöht wird.

```
for ($n=0; $n<count($fotos); $n++)
{
echo "$fotos[$n]<br>";
$n++;
}
```

Nicht nur den Schleifenzähler können Sie während des Anweisungsblocks verändern, sondern in Grenzen auch die Bedingung. Wenn Sie eine Schleife haben, die 10-mal durchlaufen werden soll, aber es einen Ausnahmefall gibt, bei dem die Schleife 30-Mal durchlaufen werden soll, können Sie diese Bedingung im Anweisungsblock einfügen. Die Schleife könnte dann so aussehen:

```
$max=10;
for ($n=0; $n<$max; $n++)
{
echo "$n <br>";
if($n==8){$max=30;}
}
```

Dieses kleine Beispiel ist nicht unbedingt sinnvoll, da `$n` immer den Wert 8 annimmt und damit die Bedingung der if-Anweisung erfüllt ist. Aber es geht hier darum, dass Sie sehen, dass auf die Bedingung auch während des Abarbeitens der Schleife Einfluss genommen werden kann.

Reguläre Ausdrücke

Wie eingangs bereits gesagt, sind reguläre Ausdrücke sehr nützlich und zeitsparend (vor allem im Zusammenhang mit Zeichenketten). Sie haben allerdings die merkwürdige Eigenart, im Prinzip leicht zu verstehen, aber in der Praxis des Programmierens etwas bis ziemlich verzwickt zu sein. Wir können in dieser Einführung nur die Spitze des Eisberges abdecken und Ihnen den Rat mit auf den Weg geben, etwas tiefer in die Einsatzmöglichkeiten der regulären Ausdrücke einzusteigen, sofern Sie zukünftig mit PHP programmieren möchten oder müssen.

Sie können sich reguläre Ausdrücke als ein System sich entsprechender Muster vorstellen. Im Prinzip leisten die Funktionen für reguläre Ausdrücke Folgendes: Man kann mit ihrer Hilfe vorher definierte Muster mit Text-(Teil-)Zeichenketten abgleichen, um entweder Übereinstimmungen zu finden oder eben nicht. Sofern gewünscht, können Sie Muster oder Zeichenketten bei gefundenen Übereinstimmungen zwischen dem Muster und der (Teil-)Zeichenkette dann auch ersetzen lassen. Das Verfahren ist folgendermaßen: Man schreibt zunächst das Muster, dann verwendet man eine der Funktionen und wendet das Muster auf einen Text (bzw. eine Zeichenkette) an.

Im Wesentlichen kennt PHP zwei Funktionen zum Vergleichen von Mustern und Funktionen und zum Ersetzen von übereinstimmendem Text durch einen anderen Text.

Wir beginnen damit, ein Muster zu definieren und eine Übereinstimmung zu suchen.

Muster definieren

Bevor Sie die Funktionen der regulären Ausdrücke verwenden können, müssen Sie Muster definieren. Die einzusetzende Funktion benutzt dieses Muster dann für den Vergleich. Um Muster zu definieren, gebrauchen Sie bestimmte Symbole/Zeichen, die gruppiert (und u.U. zu Klassen zusammengesetzt) werden. Die Kombination von Symbolen, Gruppen (und Klassen) bildet das Muster.

Symbole/Zeichen

Der gebräuchlichste Typ bei der Definition von Mustern ist das einfache, wortwörtlich zu nehmende Zeichen. Wenn Sie beispielsweise als Muster *a* verwenden, wird auch nur mit dem Buchstaben *a* eine Übereinstimmung gefunden. Diese Definition reicht in manchen Situationen aus, aber leistungsstärker wird der Einsatz von regulären Ausdrücken, wenn Sie spezielle Symbole verwenden, die jeweils eine eigene Bedeutung jenseits ihrer »wortwörtlichen« haben. Der Unterschied zwischen den einfachen Zeichen (also *a* stimmt nur mit *a* überein) und den speziellen Symbolen (auch als Metazeichen bezeichnet) ist der, dass Metazeichen weitreichendere Übereinstimmungen finden – z.B. entspricht der Punkt (.), der ein solches Zeichen ist, jedem einzelnen Buchstaben (». « gleich a, b, c, d, etc.) – bzw. nach bestimmten Wiederholungen eines Zeichens suchen.

Die abgebildete Tabelle listet diese Zeichen und ihre Bedeutung auf (die Symbole mit der geschweiften Klammer werden auch als Grenzen bezeichnet, sie legen damit fest, wie oft ein Zeichen oder Zeichenbereich gesucht wird).

Zeichen	Übereinstimmung
.	jeder beliebige Buchstabe
^a	stimmt überein mit einem String, der mit dem Zeichen beginnt, das nach dem »Dach« steht
a$	stimmt überein mit einem String, der mit dem Zeichen endet, das vor dem Dollarzeichen steht
a+	stimmt überein mit einem String, der mehrfach, mindestens aber einmal auftritt
a?	stimmt überein mit einem String, der höchstens einmal auftritt
a*	stimmt überein mit einem String, der beliebig häufig auftreten kann – auch keinmal
\n	stimmt mit einer neuen Zeile überein
a{2}	stimmt überein mit einem String, der zweimal auftritt

Tabelle 3.6: Metazeichen der regulären Ausdrücke

Zeichen	Übereinstimmung
a{1,}	stimmt überein mit einem String, der mindestens einmal auftritt
a{1,3}	stimmt überein mit einem Sting, der mindestens einmal und höchstens dreimal auftritt
[0-9]	stimmt überein mit irgendeiner Ziffer zwischen 0 und 9

Tabelle 3.6: Metazeichen der regulären Ausdrücke (Forts.)

Lassen Sie uns dies anhand einiger konkreter Beispiele verdeutlichen.

Das Zeichen *f+* findet seine Entsprechung mit allen Zeichenbereichen, die mindestens ein f (oder aber mehr) enthalten, beispielsweise: *f+*, als Übereinstimmung gefunden werden fisch, affe, schifffahrt, ffff etc.

Das Zeichen *b{2,3}* passt zu einer Zeichenkette, die b mindestens zweimal, höchstens dreimal enthält. Es werden also gefunden: ebbe oder blubbbad, aber nicht bbbb oder abc.

Die Kombination des Punktes . (steht dafür, dass ein beliebiger Buchstabe vorausgesetzt wird) mit dem Zeichen + ist ein Muster für einen beliebigen Buchstaben, und davon muss es mindestens einen geben. Insofern wird .+ als Zeichen häufig eingesetzt, denn es entspricht als Muster (mindestens) einem Buchstaben. Findet sich ein Match, würde die Prüfung also `true` zurückgeben.

Gruppen

Eine Gruppe besteht aus den Zeichen, wenn sie zusammengefasst in Klammern geschrieben werden. Eine einfache Gruppe wäre *(abc)*, wobei eine Gruppierung hier überflüssig ist, da *abc* das gleiche Resultat erzeugt: Es passt zur Zeichenkette abc. Interessanter ist die Gruppierung in Kombination mit der geschweiften Klammer. Schauen Sie noch einmal in die obige Tabelle: *a{2}* passt zu aa, d.h. der Buchstabe a tritt zweimal auf. Wenn Sie *(abc){2}* schreiben, findet sich bei abcabc eine Übereinstimmung. Deutlicher wird der Unterschied bei folgendem Beispiel eines gruppierten Musters:

Das Muster *schiff+* findet eine Übereinstimmung, wenn schiff gefolgt wird von (mindestens) einem weiteren f. *(schiff)+* hingegen entspricht nur einer Zeichenkette, die schiff und (mindestens) noch einmal schiff enthält. Alles klar?

Funktionen der regulären Ausdrücke

Zwei vordefinierte Funktionen suchen Übereinstimmungen mit festgelegten Mustern. Die eine ist `ereg()` und die andere `eregi()`. Der Unterschied zwischen beiden ist ganz einfach: `Ereg` unterscheidet zwischen Groß- und Kleinschreibung und `eregi` nicht.

Beide Funktionen geben `true` zurück, wenn die Übereinstimmung zwischen Muster und Zeichenkette gefunden wird, und ansonsten `false`. Ein Einsatz der Funktion(en) kann folgendermaßen aussehen:

```
ereg("definiertes Muster", "zu vergleichende zeichenkette");
```

Geschickter ist es, das Muster einer Variablen zuzuweisen:

```
$muster = "definiertes Muster";
$string = "zu vergleichende Zeichenkette";
ereg($muster, $string);
```

Vergleichen und Ersetzen

Während die Funktionen `ereg()` und `eregi()` beispielsweise sehr wirksam einsetzbar sind, um die Gültigkeit bestimmter Eingaben (z.B. in einem Formularfeld) zu testen, ist es in anderen Fällen wünschenswert, Eingaben zu verwandeln. Dann kommen die Funktionen

`ereg_replace()` oder `eregi_replace()`

ins Spiel.

Wie eingangs bereits erwähnt, demonstrieren wir den konkreten Einsatz regulärer Ausdrücke erst im letzten Kapitel des Buches, da wir Sie an dieser Stelle damit vermutlich überfordern würden. Im Kapitel *Tipps und Tricks* finden Sie dann zwei Beispiele für die Verwendung regulärer Ausdrücke.

Kapitel 4
Ein Kontaktformular

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.

Diese Kapitel 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>
```

Listing 4.1: Das Grundgerüst der HTML-Datei

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 in Listing 4.2 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>
```

Listing 4.2: Das HTML-Script für ein einfaches Formular mit drei Textfeldern und einem Auswahlfeld

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`. Über die Unterschiede zwischen beiden Methoden lesen Sie im nächsten Abschnitt. 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. Der Browser müsste eine Seite ausgeben, die mehr oder minder aussieht wie in Bild 4.1.

Bild 4.1: Das Kontaktformular im Browser

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.

Hinweis

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;
```

Bild 4.2 zeigt das kleine ›Script‹ und das Ergebnis im Browser.

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.

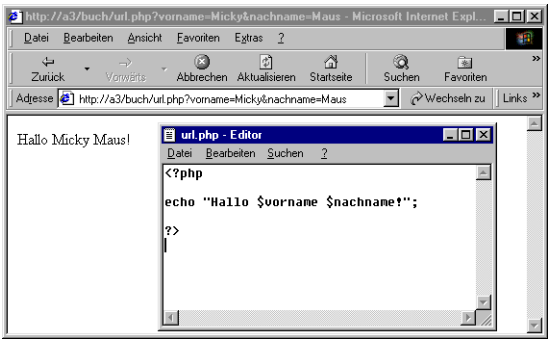

Bild 4.2: Die Inhalte wurden per URL übermittelt. Achten Sie auf die Adresszeile!

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 das Listing 4.3. 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.

Hinweis

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. Listing 4.3 zeigt den fertigen Programm-Code.

```
<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>
```

Listing 4.3: Das Script für eine einfache Feedback-Seite mit einer if-elseif-else-Verzweigung

Hinweis

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 (Bild 4.3). 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 4.4 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.

Bild 4.3: Das Formular mit ausgefüllten Feldern

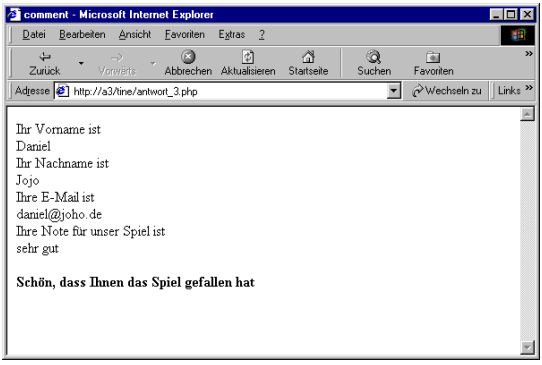

Bild 4.4: Die Formulardaten werden im Browser ausgegeben

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.

Hinweis

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, also wie in Listing 4.3.
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 Listing 4.3, 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 Listing 4.4, 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>
```

Listing 4.4: Das PHP-Script, das die Formulardaten verarbeitet und eine Fehlermeldung ausgibt, wenn die Felder nicht vollständig ausgefüllt werden

Das Script testen

Testen Sie nun das Script, indem Sie es in Ihrem Browser öffnen. Sie erhalten, ersichtlich in Bild 4.5, 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.

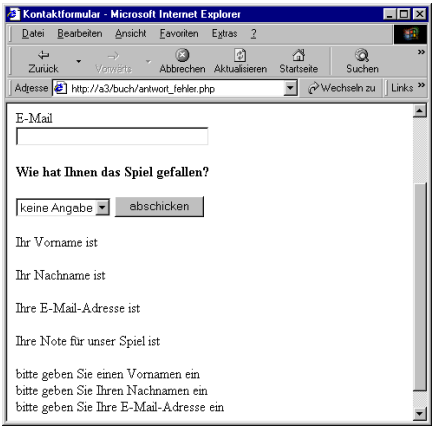

Bild 4.5: So soll die Seite beim Aufrufen eigentlich nicht aussehen!

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.

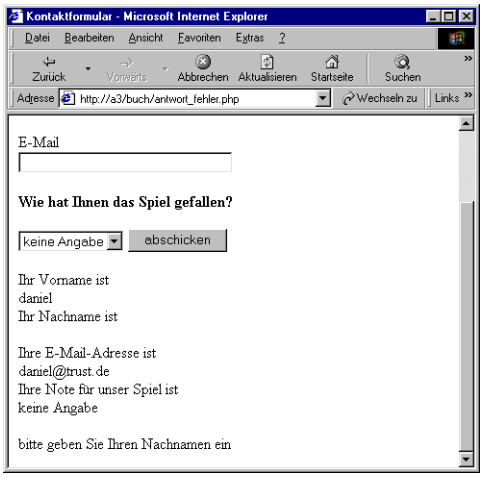

Bild 4.6: Soweit hat es geklappt, die Meldung erscheint.

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. Listing 4.5. zeigt das modifizierte Script:

```
<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>
```

Listing 4.5: Das Script, ergänzt durch die Variable $sent bzw. die Negation der Variablen, sodass das Formular nur beim ersten Aufrufen der Seite angezeigt wird.

Hinweis

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);
}
```

Listing 4.6: Scriptfragment mit Fehlermeldung und Löschen der Variablen $sent

Hinweis


```
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);
```

Hinweis

```
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 Sie erstens ein Buch über PHP in den Händen halten und weil zweitens 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.
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.

Tabelle 4.1: Die Argumente der Funktion mail

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);
}
```

Listing 4.7: Das Scriptfragment, das die Nachricht im gewünschten Format »zusammenbaut«, den Header erzeugt und anschließend die Mail auf den Weg schickt

Hinweis

`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.

Hinweis

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 (Listing 4.8) 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>
```

Listing 4.8: Das komplette Script zum Versenden einer Mail mit eingebauter Fehlermeldung

Kapitel 5
Währungen umrechnen

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!

Dieses Kapitel 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 5.1. 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.

Bild 5.1: Das Formular für einen Euroumrechner

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 5.2.

Bild 5.2: Ein Formular zur Umrechnung von Beträgen

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 Kapitel über die PHP-Grundlagen 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 Listing 5.1. 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>
```

Listing 5.1: Das Script für einen einfachen Umrechner einer Währung in Euro

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 5.3 sehen Sie das Ergebnis.

Bild 5.3: Das Ergebnis der Berechnung wird angezeigt.

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.

Listing 5.2 zeigt das ergänzte Script bis zum Schließen des PHP-Tags.

```
<?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>";
}
}
?>
```

Listing 5.2: Der endgültige PHP-Teil des Scripts mit Fehlermeldung

Bild 5.4: Der Besucher erhält eine Meldung, wenn er keinen Betrag eingegeben hat!

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 5.4 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.

▶ Beachten Sie bitte: Was Arrays sind, haben wir in *Kapitel 3, PHP – die Grundlagen* bereits erklärt. Blättern Sie bitte notfalls zurück; 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.

Listing 5.2 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>";

?>
```

Listing 5.3: Das Script für einen Währungsumrechner

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.

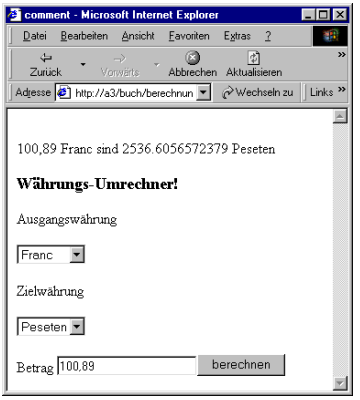

Bild 5.5: Das Ergebnis wird angezeigt und die Optionen und der Betrag sind weiterhin zu sehen.

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 5.6 zeigt das Script und die Ausgabe:

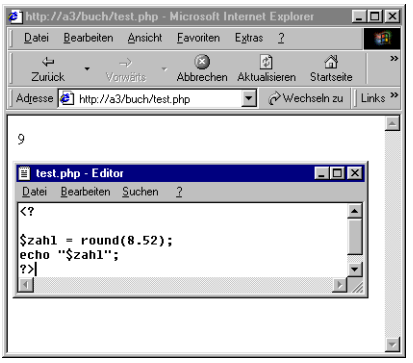

Bild 5.6: Zahlen auf- oder abrunden mit round()

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.

Hinweis

```
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.

Hinweis

```
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>";
```

Listing 5.4: Die Zahlen werden mit round() ab- bzw. aufgerundet und mit number_format() formatiert.

Testen Sie nun den Währungsumrechner erneut. Sofern die Anweisung geklappt hat, würde ein Ergebnis nun etwa so dargestellt werden:

Bild 5.7: Der Browser zeigt das Ergebnis in formatierter Form.

Weitere Vereinfachungen

Hinweis

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.

```
next()
```

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.

Das gesamte Script, zu sehen in Listing 5.4, hat dann folgende Form:

```
<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>";

?>
```

Listing 5.5: Das komplette Script für einen Währungsumrechner

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 5.8 zu sehen, die Lire auftauchen:

Bild 5.8: Die Auswahlfelder wurden ergänzt durch eine weitere Währung.

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 `$rate` immer auf eine »Basiswährung« beziehen müssen. Haben Sie das Script so angepasst, wie es im letzten Abschnitt beschrieben wurde, reicht es, das Array `$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. Der Umgang mit Dateien wird schwerpunktmäßig in Kapitel 8 erklärt, in dem es um die Programmierung eines Gästebuchs und das Speichern der Daten in eine TXT-Datei geht.)

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. Der Dateiupload wird im Kapitel 11 beschrieben.

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.

Kapitel 6
Wechselnde Textausgaben

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 6.1:

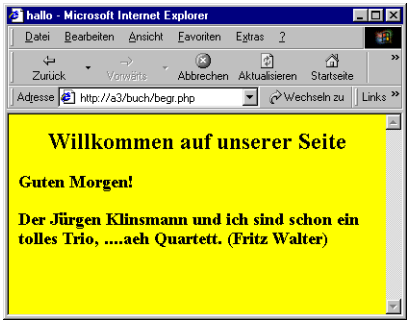

Bild 6.1: In diesem Kapitel wird eine Seite mit einer wechselnden Begrüßung und einer »Fußballerweisheit« erstellt.

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 *Kapitel 3, PHP – die Grundlagen*).

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.

Hinweis

```
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 (Listing 6.1):

```
<?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>
```

Listing 6.1: Das Script für die Seite mit wechselndem Text

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);
```

Hinweis

```
int mt_rand(int min [, int max])
```

Die Funktion gibt eine Zufallszahl zwischen min und max zurück.

Hinweis

```
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];
```

Hinweis

```
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 6.2 oder Bild 6.3. 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.

Bild 6.2: Die User werden begrüßt und mit einem Spruch beglückt.

Bild 6.3: Erscheint nach der Aktualisierung ein anderer Spruch, hat die Programmierung geklappt!

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>
```

Listing 6.2: Der Programmcode zum Auslesen einer Textdatei

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*).

Kapitel 7
Ein Gästebuch programmieren

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.

Tabelle 7.1: Die Modi von fopen()

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.

Hinweis

```
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);
```

Hinweis

```
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.

Hinweis

`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;
```

Hinweis

```
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 7.1 zeigt das kleine Script und die Ausgabe im Browser.

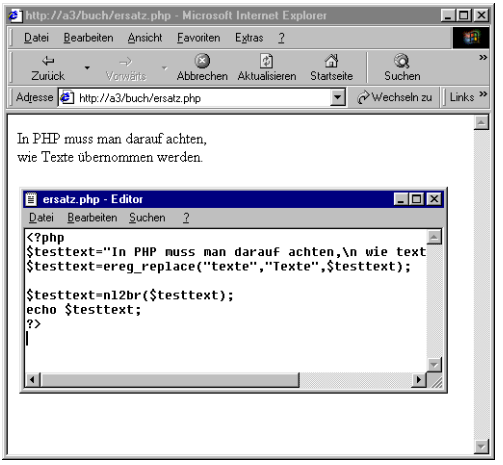

Bild 7.1: Der Text wurde mit ereg_replace und nl2br bearbeitet.

Hinweis

```
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 7.2.

Bild 7.2: Ein Formular mit einem Textbereich

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

Listing 7.1 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>
```

Listing 7.1: Das Script für das Gästebuch, u.a. mit Fehlermeldungen, Befehlen zum Bearbeiten der Texteingaben und für das Anlegen der Textdatei

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);
```

Hinweis

```
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.

Hinweis

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);
```

Hinweis

```
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.

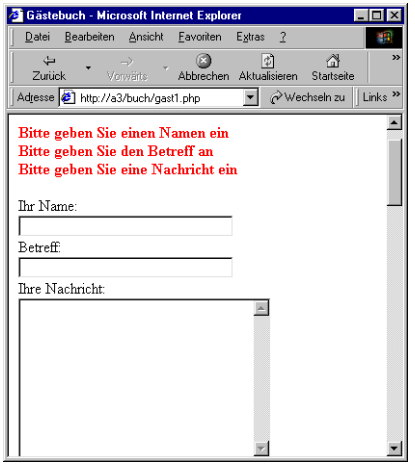

Bild 7.3: Mit der eingebauten Fehlermeldung erscheinen im Browser entsprechende Meldungen.

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.

Bild 7.4: Ein Datensatz in der Textdatei

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; ?>
```

Bild 7.5: Die Eingaben verschwinden nach dem Abschicken nicht aus den Feldern.

Hinweis

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 7.6.

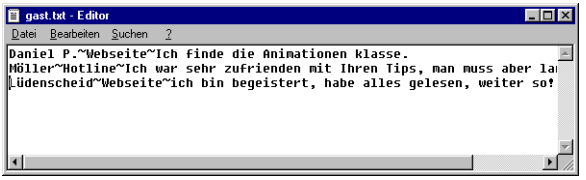

Bild 7.6: Die Einträge werden in einer Text-Datei gespeichert.

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');
```

Hinweis

```
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.

Beachten Sie bitte: 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.

Hinweis

```
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; ?>
```

In Listing 7.2 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>";
}
```

Listing 7.2: Der Teil des Scripts, der für das Auslesen der Datei sorgt

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 7.7 sehen Sie, wie die Seite in etwa aussehen müsste.

Bild 7.7: Die Einträge sind im Browser einsehbar.

Hinweis

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 in diesem Buch 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.

Kapitel 8
Einen Counter programmieren

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.

Ein Blick auf Listing 8.1 hilft Ihnen mit dem Counter auf die Sprünge.

Hinweis

```
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.

Das folgende Listing (8.1) 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;

?>
```

Listing 8.1: Das Script für den einfachen Counter ohne Reloadsperre

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()
```

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);
```

Hinweis

```
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 8.1 entspricht.

Klicken Sie dann ein paar Mal auf AKTUALISIEREN. Wenn es geklappt hat, wird die angezeigte Zahl hochgezählt. Der Browser zeigt, wie in Bild 8.2 zu sehen, den Erfolg.

Bild 8.1: Der Counter mit ein wenig zusätzlichem Text und der TXT-Datei im Texteditor

Bild 8.2: Der Counterstand nach einigen Aktualisierungen

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.

> **Hinweis**
>
> 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.

> **Hinweis**
>
> `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 (Listing 8.2.) für den Counter mit Cookie. Sie finden hier die Zeile:

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

Damit wird das Cookie gesetzt.

Das vollständige Script für den Counter mit implementiertem Cookie wird aus dem Listing 8.2 ersichtlich:

```
<?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'>";
}

?>
```

Listing 8.2: Das Script für einen Counter mit Cookie

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.

Hinweis

`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;
```

Hinweis

(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++)
{
```

Hinweis

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'>";
}
```

Bild 8.3 zeigt die Ausgabe im Browser.

Bild 8.3: Der Counter mit drei GIFs und im Vordergrund die TXT-Datei mit dem Zählerstand

Nach vielen Besuchern können Sie sich über eine Anzeige freuen, die aussieht wie in Bild 8.4.

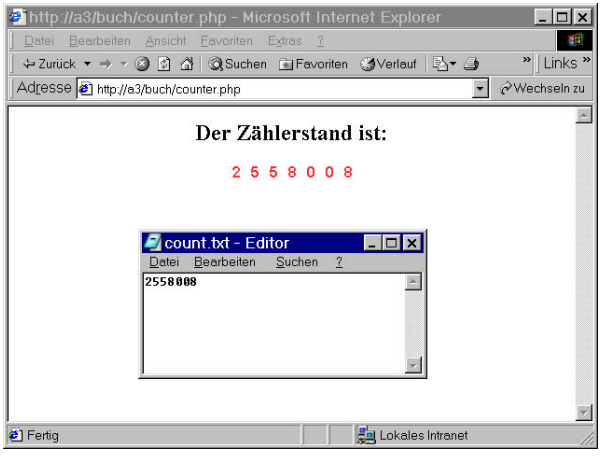

Bild 8.4: Der Counter mit einem hohen Zählerstand

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!

Bild 8.5 zeigt, wie die Ausgabe nach erfolgreicher Programmierung aussehen soll.

Das erweiterte Script

Das Script, das ein Cookie setzt, die Ziffern als GIFs anzeigt und für die feste Stellenanzahl sorgt, sehen Sie in Listing 8.3.

```
<?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

?>
```

Listing 8.3: Das erweiterte Script, das zusätzlich für eine feste Stellenzahl sorgt

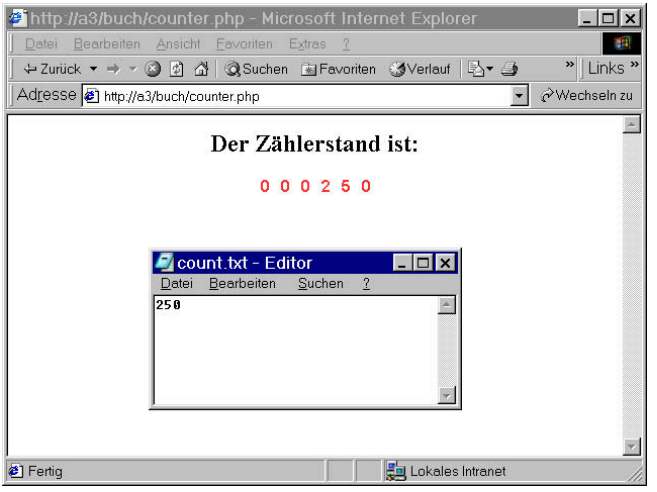

Bild 8.5: Der Counter mit einer festen Anzahl an Stellen

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 8.6 zeigt den Zählerstand in der Textdatei und im Browser.

Bild 8.6: Wenn der Zähler mehr als 6 Stellen hat, werden die letzten Ziffern abgeschnitten.

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. Listing 8.4 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

?>
```

Listing 8.4: Der erweiterte Scriptteil: So werden auch mehr als 6 Stellen angezeigt.

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 8.7.

Hinweis

```
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.

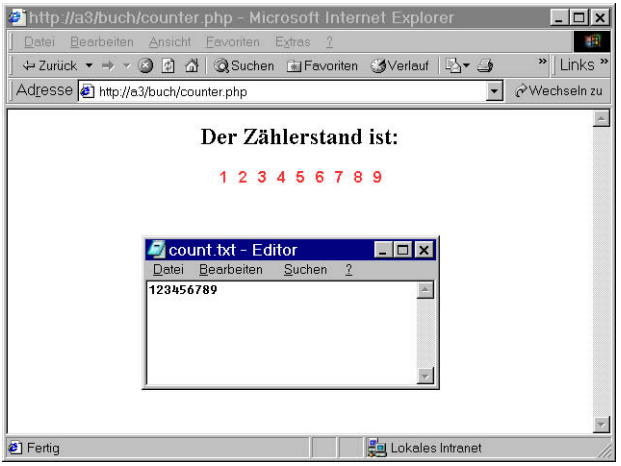

Bild 8.7: Der Counter mit mehr als 6 Stellen

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.

Kapitel 9
MySQL – Grundlagen

Grundsätzlich geht es bei Web-Projekten immer wieder darum, Daten extrahieren und speichern zu können.

Wir können unterscheiden zwischen zwei Methoden der Datensicherung. Die eine Methode wurde im Kapitel über die Programmierung eines Gästebuchs angewendet und erläutert. Die zweite Methode ist die Verwendung von Datenbanken, die gegenüber der Arbeit mit Dateien und Verzeichnissen Vorteile besitzt, aber den Nachteil mit sich bringt, dass man sich mit Datenbanken auskennen muss. Als Anfänger hat man häufig den Eindruck, als hätten Datenbanken so ihre Tücken, doch schon bei kleineren Projekten zeigt sich in der Regel, dass Datenbanken relativ einfach zu handhaben sind und bequeme Lösungen anbieten, die bei einer Speicherung der Daten in Textdateien umfangreiche Tricks und Kniffe voraussetzen würden.

Unter Datenbanken versteht man – vereinfacht gesagt – zunächst einmal eine Sammlung von Tabellen, in denen Informationen gespeichert sind. Das Internet käme ohne Datenbanken nicht aus. Ob es sich um E-Commerce-Seiten handelt, Befragungen, Portale etc. etc., überall werden Daten generiert, die gesichert werden müssen.

PHP unterstützt eine ganze Reihe von Datenbank-Servern, also die Software, die Datenbanken steuert und verarbeitet. Ein herausragende Rolle spielt *Oracle*, ein besonders leistungsfähiges, jedoch teures DBMS (Datenbank Management System). Folglich benutzen wir hier als Beispiel-DBMS nicht *Oracle*, sondern *MySQL*, das zwar weniger leistungsfähig, aber auch effektiv ist, gut mit PHP zusammenarbeitet, und im Internet für zahlreiche Plattformen als Download zur Verfügung steht.

MySQL hat sich bei den meisten Providern als Basis-Datenbank durchgesetzt. Der große Vorteil von *MySQL* ist insbesondere seine Geschwindigkeit. Hingegen werden einige Funktionen, die für umfangreiche Datenbank-Projekte hilfreich sind, nicht unterstützt. Dies sind jedoch Funktionen, die Sie als Einsteiger in die Datenbank-Welt kaum vermissen werden.

Sollten Sie sie dennoch für ein Projekt benötigen, müssen Sie nicht auf teure Datenbanken wie *Oracle* zurückgreifen, sondern könnten z.B. auch *PostgreSQL* nutzen. Einige Provider bieten die Unterstützung von *PostgreSQL* an. Der Befehlssatz in PHP, mit dem Sie *MySQL* bzw. *PostgreSQL*-Datenbanken ansprechen, ist fast identisch, sodass sich Kenntnisse und Programme schnell austauschen lassen.

Die oben genannten Datenbanken werden entwickelt und abgefragt mit SQL (Structured Query Language), einer Sprache, die relativ benutzerfreundlich ist, aber nicht allzu viele Kommandos kennt. Dies macht die Entwicklung komplexerer Anweisung mitunter etwas umständlich.

Wir können in einem PHP-Buch nicht in erschöpfendem Umfang *MySQL* bzw. SQL erklären. Verstehen Sie die einführenden Worte deshalb bitte nicht als regelrechten Einstieg in SQL. Wir möchten an dieser Stelle lediglich dafür sorgen, dass Sie genug lernen, um ein Grundverständnis zu entwickeln und ein paar relativ einfache Beispiele mitspielen können.

Nachdem Sie *MySQL* installiert haben, können Sie in PHP-Scripten auf die Datenbank zugreifen. Für diesen Zugriff benötigen Sie eine Verbindung zum Datenbankserver (auf die Installation von *MySQL* wurde im Kapitel 3 eingegangen).

Die Verbindung zum Datenbank-Server

Der erste Arbeitsschritt besteht darin, dass Sie eine Verbindung mit dem Datenbank-Server, also in diesem Fall zu *MySQL*, herstellen müssen. Diese Verbindung ist dann sozusagen das Tor, durch das Sie Zugang zu den weiteren Befehlen erhalten.

Die Verbindung erfolgt über eine Funktion

```
mysql_connect()
```

Die Klammer kann bis zu drei Argumente aufnehmen; meistens handelt es sich um den Host-Namen, den (vorgegebenen) Benutzernamen, und das (vorgegebene) Passwort. Der Host-Name ist in aller Regel *localhost*, sofern die Datenbank auf dem gleichen Rechner läuft wie der Webserver. Sollte dies nicht der Fall sein, hat Ihr Provider Ihnen den Namen des Servers in irgendeiner Form mitgeteilt. Mit dem Benutzernamen und dem Passwort wird angegeben, welche Rechte Sie bezüglich der Datenbank haben.

Hinweis

```
int mysql_connect([string hostname[:port][:/path/to/socket][, string
Benutzername][, string Kennwort]]])
```

Die Funktion stellt eine Verbindung zum Datenbankserver her. Alle Parameter sind im Prinzip optional, aber in der Regel werden die entsprechenden Zugangsdaten für die Datenbank (die Sie vom Provider erhalten haben) in die Parameter geschrieben. Bleiben die Parameter leer, gilt: hostname, Name des Benutzers, dem der Server gehört, leeres Kennwort.

Beim Verbinden setzen Sie normalerweise eine Variable für die Verbindung (Verbindungsbezeichner), um damit dann weiter zu arbeiten. Konkret sieht der Code für die Verbindung dann so aus:

```
$link = mysql_connect("localhost", "benutzername", "passwort");
```

Zugriffsrechte

Direkt nach der Installation sind keine Benutzernamen und Kennwörter in *MySQL* festgelegt. Jeder hat also Zugriff auf alle Datenbanken/Tabellen des Servers. Dies mag für Sie zu Hause zum Testen hinreichen, ein professioneller Provider wird diesen Zustand nicht dulden können.

Daher legen die Provider in der Regel für jeden Account eine Datenbank an. Der Name dieser Datenbank und die dazugehörige Benutzernamen/Passwort-Kombination werden Ihnen mitgeteilt und Sie haben dann nur auf die Datenbank Ihres Accounts Zugriff.

In dieser Datenbank sind Sie allerdings König! Sie können Tabellen anlegen, löschen, die Tabelle mit Datensätzen füttern etc. Die Begrenzung der *MySQl*-Datenbanken werden Sie im Regelfall nicht erreichen. Sie müssen sich also nicht, wie etwa in Access, mit Tabellen mit einer maximalen Anzahl von Feldern (255) herumärgern. Auch Leistungseinbrüche wie man Sie von Access bei gleichzeitigen Zugriff bereits weniger User kennt, wird Ihnen *MySQl* ersparen. Allerdings müssen Sie sich bei den meisten Providern den *MySQL*-Datenbankserver mit vielen Kunden teilen, sodass es in Spitzenzeiten zu Engpässen kommen kann. In diesem Fall hilft es nur, Ihren Provider auf das Problem aufmerksam zu machen, und im Wiederholungsfall über einen Wechsel nachzudenken!

Eine Datenbank anlegen

Für das erste Beispiel in diesem Kapitel soll eine neue Datenbank angelegt werden, die z. B. Informationen speichert, die aus einem Formular einer Webseite generiert werden.

Beginnen Sie ein neues Dokument in Ihrem Editor und geben Sie den üblichen HTML-Header ein.

1. Öffnen Sie den PHP-Teil, indem Sie `<?php` eingeben.
2. Setzen Sie die Variablen für die Datenbank. Die Definition von Variablen ist nicht zwingend, hat aber wieder einmal den Vorteil, dass Sie das Script flexibel halten. Sie geben (beispielsweise) ein:

```
$benutzer= "Ihren_Benutzername";
$passwort= "Ihr_Passwort";
$dbname= "Testdatenbank";
```

3. Danach verbinden Sie sich mit dem Datenbankserver, indem Sie schreiben

```
$link = mysql_connect("localhost", $benutzer, $passwort);
```

4. Sofern die eingegebenen Daten korrekt sind, erfolgt nun die Verbindung. Ansonsten erhalten Sie an dieser Stelle im Script eine Fehlermeldung.
5. Die Funktion zum Erstellen einer Datenbank lautet: `mysql_create_db();` Um eine neue Datenbank anzulegen und eine entsprechende Meldung im Browser zu erhalten, können Sie eine if-Anweisung konstruieren:

Hinweis

```
int mysql_create_db(string Datenbankname [, int Verbindungskennung])
```

Die Funktion legt eine Datenbank an. Sie müssen das Recht haben, eine Datenbank anlegen zu dürfen.

```
if(myql_create_db($dbname, $link))
{
echo "Die Datenbank $dbname wurde erstellt<br>;
} else { echo "keine Datenbank erstellt<br>";
}
```

6. Sie können nun – dies ist optional – die Verbindung schließen. Die Verbindung würde auch automatisch mit dem Ende des PHP-Scripts beendet werden, aber zu einer sauberen Programmierung gehört die Verwendung der Funktion `mysql_close()`. Sie geben also zu guter Letzt ein:

   ```
   mysql_close($link);
   ?>
   </body> </html>
   ```

7. Speichern Sie das neue Script als PHP-Datei und testen Sie es im Browser.

Hinweis

Wie oben beschrieben, gestatten die meisten Provider nicht das Anlegen von Datenbanken. Das obige Beispiel können Sie also wahrscheinlich nur in Ihrer heimischen Testumgebung erfolgreich mitspielen.

Eine Tabelle anlegen

Eine Datenbank besteht aus Tabellen, diese wiederum aus Zeilen und Spalten. Das nächste Beispiel zeigt, wie Sie eine Tabelle anlegen, in der Daten gespeichert werden können.

Eine neue Tabelle zu erstellen erfordert das Schreiben und Benutzen einer SLQ-Abfrage. Die Funktion dafür lautet generell:

```
mysql_db_query();
```

Als Argumente setzen Sie den Datenbanknamen, eine Zeichenkette (SQL-String), die die Abfrage enthält, und den Verbindungsbezeichner ein.

Hinweis


```
int mysql_db_query(string Datenbank, string Anfrage [, int
Verbindungskennung])
```

Die Funktion sendet eine Abfrage an die Datenbank.

Die *SQL-Abfrage* (auch SQL-Statement oder SQL-String genannt) ist in PHP ein normaler Text, den Sie in einer Variablen speichern können. *PHP* kann mit diesem Text nichts anfangen. Die SQL-Abfrage sind Befehle/Anweisungen für die Datenbank. Mit `mysql_db_query()` wird dieser Text mit SQL-Befehlen an die Datenbank übergeben.

Im Gegensatz zu PHP kann die Datenbank mit diesen Befehlen – sofern sie richtig formuliert sind – arbeiten. Die Ergebnisse des SQL-Strings werden an PHP zurückgegeben. Hierbei kann es sich je nach SQL-Statement mal um eine einfache Erfolgsmeldung (`true`) oder um eine Ergebnisliste mit mehreren Hundert Datensätzen handeln.

Im Klartext: in PHP sind die SQL-Statements zunächst reiner »wertloser« Text, die erst durch das Senden an die Datenbank einen Wert erhalten und Ergebnisse liefern oder Aktionen durchführen. Das Erstellen der SQL-Statements für die Datenbank wird im Folgenden kurz behandelt.

Zurück zu `mysql_db_query`: Sofern Sie regelmäßig mit nur einer Datenbank arbeiten, können Sie diese einmalig auswählen und anschließend mit `mysql_query` arbeiten. `mysql_query` ist identisch mit `mysql_db_query`, nur dass Sie das erste Argument (den Datenbanknamen) weglassen. Der Befehl zur Vorauswahl der Datenbank heißt `mysql_select_db()`. Als Argumente werden der Datenbankname und die Verbindungskennung erwartet.

Nun setzen Sie SQL ein. Die Tabelle wird erstellt mit einem SQL-String, der folgendermaßen geschrieben wird:

```
create table Tabellenname(Feldname1, Feldname2,etc.);
```

Dieses SQL-Statement wird erweitert, indem Sie nicht nur – jeweils durch Komma getrennt – die Spaltennamen angeben, sondern auch den Datentyp und weitere Attribute. Gebräuchlich sind die Datentypen *Text* (CHAR) und *Integer* (abgekürzt mit INT). Außerdem ist es ratsam, eine erste Spalte (Id) mit einem Primärschlüssel zu kreieren, in der jede neue Zeile automatisch um eins hochgezählt wird. Eine solche Abfrage würde dann so aussehen:

```
$abfrage= "create table Tabellenname(Id INT NOT Null AUTO_INCREMENT Primary key,
Daten CHAR)";
```

INT und *CHAR* (Char steht für Characters. Dies zu wissen, macht die Abkürzung etwas verständlicher!) geben also den Typ der Daten an, die in der jeweiligen Spalte gesammelt werden. Angelegt werden: das Feld »Id« und das Feld »Daten«. `AUTO_INCREMENT` legt fest, dass das Feld bei jedem neuen Datensatz um eins erhöht wird.

Dann stellen Sie wie bereits oben erläutert die Verbindung her und schicken den SQL-String an den Datenbank-Server, damit er diesen ausführt und – in diesem Fall – die Tabelle erstellt. So ist die Schreibweise:

```
$link = mysql_connect("localhost", $benutzer, $passwort);
mysql_db_query($dbname, $abfrage, $link);
```

Mit

```
mysql_close($link);
```

wird die Verbindung zum Datenbankserver dann beendet.

Hinweis

Jeder Verbindungsaufbau zur Datenbank nimmt eine gewisse Zeit in Anspruch. Greifen Sie in einem Script mehrfach auf die Datenbank zu, ist es ratsam, die Verbindung nicht zu schließen, um diese Zeit einzusparen. Die eigentlichen SQL-Abfragen benötigen verglichen zum Verbindungsaufbau fast gar keine Zeit.

Hinweis

```
int mysql_close([int Verbindungskennung])
```

Die Funktion beendet die Verbindung zum Datenbankserver.

Die Datenbanktabelle mit Daten füllen

Die Tabelle, die Sie eben angelegt haben, soll Daten aufnehmen. Diese Daten können z. B. von einem HTML-Formular stammen. Der Prozess, diese Informationen an die Datenbanktabelle zu schicken, ähnelt der Aktion, eine Tabelle zu erstellen. Das SQL-Statement ist natürlich ein anderes. Es lautet:

```
insert into $tabellenname (spaltenname1, spaltenname2,etc.) values(Wert1, Wert2,
etc.);
```

Die erste Klammer enthält, durch Komma separiert, die Namen der Felder/Spalten, denen Sie Werte zuweisen möchten. In der zweiten Klammer hinter `values` befinden sich die Werte der einzelnen Spalten, getrennt durch ein Komma. Die Zuordnung von Werten zu Feldern/Spalten wird über die Reihenfolge in den beiden Klammern gewährleistet. Felder/Spalten, die nicht in der Auflistung vertreten sind, werden mit dem Standardwert, den Sie als Attribut einer Spalte in der Tabellendefinition festlegen können, ausgefüllt. Die so definierte Abfrage

```
$abfrage ="insert into $tabellenname (Daten) values(Wert1)";
```

muss dann an den Datenbank-Server geschickt werden, wobei Sie wieder die Funktion `mysql_db_query()` verwenden.

Daten aus einer Datenbank gewinnen

In diesem Abschnitt wollen wir noch ansprechen, wie Sie Daten aus einer existierenden Datenbank herausziehen. Für diesen Prozess verwenden Sie ebenfalls die Funktion `mysql_db_query()`, aber im Unterschied zum Einfügen von Daten arbeiten Sie bei dieser Aktion mit einer Variablen, der Sie die aus der Datenbank stammenden Daten zuweisen.

Der SQL-Befehl, mit dem Sie Daten aus einer Tabelle lesen, lautet

```
select * from $Tabellenname;
```

Das Sternchen besagt, dass alle Felder der Tabelle abgefragt werden, was oft auch genau das ist, was benötigt wird. Sie können die Abfrage aber auch auf einzelne Felder begrenzen. Dann schreiben Sie statt des Sternchens die Namen der Felder, aus denen Daten gesammelt werden sollen, beispielsweise

```
select Nachname from $Tabellenname;
```

Es gibt noch eine andere Möglichkeit, die Abfrage einzuschränken. Sie können auch anweisen, dass nur bestimmte Eingaben gesammelt werden. Wenn Sie beispielsweise schreiben

```
select * from $Tabellenname where Vorname="Hans";
```

sagen Sie im Klartext, dass zwar die Daten aus allen Spalten aufgenommen werden, aber nur, wenn in der Spalte *Vorname* der Name *Hans* auftaucht.

Den wichtigsten Unterschied zwischen dem Einfügen von Daten in eine Datenbank und der Gewinnung von Daten aus einer Datenbank entdecken Sie im Umgang mit der Abfrage. Empfehlenswert ist es, das Ergebnis der Abfrage einer Variablen zuzuweisen. Das würde z.B. so aussehen, (wobei der SQL-String in der Variablen `$query` gespeichert ist) :

```
$ergebnis=mysql_db_query($dbname, $query, $link);
```

In der Variablen `$ergebnis` steht nun die gesamte Ergebnismenge, anders ausgedrückt: Alle Datensätze, die Ihre Abfragebedingung erfüllen, stehen nun in der Variablen `$ergebnis`.

Um die Informationen Zeile für Zeile zu erhalten, steht die Funktion `mysql_fetch_array()` zur Verfügung. Mit dieser Funktion schreiben Sie die Information einer Zeile in ein assoziatives Array, wobei die Feldnamen der befragten Datenbanktabelle als Schlüssel verwendet werden. Die Funktion rutscht in der Ergebnisliste automatisch eine Zeile weiter. Sie müssen diese Funktion für alle Zeilen der Ergebnismenge anwenden. Die Anzahl der Zeilen des Ergebnisses können Sie mit `mysql_num_rows()` ermitteln. Um alle Daten der Varia-

blen `$ergebnis` in ein mehrdimensionales Array zu schreiben, können Sie folgende for-Schleife verwenden:

```
for($i=0;$i<mysql_num_rows($ergebnis);$i++)
{
$array[$i]=mysql_fetch_array($ergebnis);
}
```

Hinweis

`array mysql_fetch_array(int Ergebnisliste[, int Ergebnistyp])`

Die Funktion schreibt aus der Ergebnisliste einer Datenbankabfrage den aktuellen Datensatz in ein assoziatives Array.

Alternativ kann man auch eine `while`-Schleife verwenden. Die sieht dann folgendermaßen aus:

```
$i=0;
while($array[$i]=mysql_fetch_array($ergebnis))
{
$i=++$i;
}
```

Um die Datensätze der Ergebnisliste zu sortieren, müssen Sie den SQL-String erweitern. Mit `ORDER BY` Feldname sagen Sie der Datenbank, nach welchem Feld sortiert werden soll. Mit `ORDER BY` Feldname `ASC` wird absteigend sortiert, mit `DESC` aufsteigend. Ein `Select`-String sieht dann so aus:

```
select * from $Tabellenname where Vorname="Hans" ORDER BY Nachname ASC
```

Nachdem Sie das Ergebnis Ihrer Abfrage in ein Array geschrieben haben, können Sie den Arbeitsspeicher, den die Ergebnisliste belegt, wieder freigeben. Dies machen Sie mit dem Befehl `mysql_free_result()`. Also:

```
mysql_free_result($ergebnis);
```

Hinweis

`int mysql_free_results(int Ergebnisliste)`

Mit dieser Funktion wird Arbeitsspeicher, der belegt wurde, wieder freigegeben.

Werte manipulieren

Um die Werte eines vorhandenen Datensatzes zu ändern, steht das SQL-Statement

```
update $tabellenname set spaltenname1=wert1, spaltenname2=wert2 where bedingung
```

zur Verfügung. Möchten Sie z.B. einen Datensatz mit der ID-Nr. 47 im Feld *ID* den neuen Wert *Willi* im Feld *Daten* zuweisen, würde der SQL-String folgendermaßen geschrieben werden:

```
$abfrage="update $tabellenname set Daten="willi" where ID=47";
```

Diesen SQL-String schicken Sie wieder mit `mysql_db_query` an die Datenbank. Alle Datensätze, die die Bedingung hinter `where` erfüllen, werden aktualisiert. Wählen Sie die Bedingung so, dass sie nur auf einen Datensatz zutrifft, ändern Sie auch nur diesen einen Datensatz.

Datensätze löschen

Sie können sowohl einzelne als auch mehrere Datensätze in einem Rutsch aus einer Tabelle löschen, je nach dem wie Sie die Bedingung formulieren. Das SQL-Statement lautet:

```
$abfrage ="delete from $tabellenname where bedingung";
```

Das Tool phpMyAdmin

Bisher haben Sie die MySQL-Datenbank sozusagen im Blindflug genutzt, denn *MySQL* umfasst – im Gegensatz zu beispielsweise *Access* – keine komfortable Benutzeroberfläche, mit deren Hilfe Sie Tabellen anlegen oder Daten manipulieren können. Eine solche Benutzeroberfläche bietet jedoch *phpMyAdmin*, durch deren Einsatz die Arbeit mit Datenbanken erheblich erleichtert wird. Erfreulicherweise ist es gratis als PHP-Lösung downzuloaden (*http://phpwizard.net/projects/phpmyadmin*).

phpMyAdmin anpassen

Nachdem Sie die entsprechenden Dateien auf Ihre lokale Festplatte kopiert und anschließend entpackt haben, kopieren Sie sie in ein gesondertes Verzeichnis Ihres Web-Space. Anschließend öffnen Sie in Ihrem Editor die Datei *config.inc.php*. In dieser Datei müssen Sie einige Angaben zu Ihrer Datenbank machen, damit *phpMyAdmin* darauf zugreifen kann.

Zunächst geben Sie in der Zeile mit der Variablen `$cfgPmaAbsolutUri` den Pfad, in den Sie gerade *phpMyAdmin* kopiert haben, an, z.B. :

```
$cfgPmaAbsolutUri='http://www.ihredomain.de/phpMyAdmin'
```

Anschließend müssen Sie einige Zeilen weiter unten die Anbindung zur Datenbank bearbeiten:

```
$cfgServers[1]['host']='localhost';
```

Ersetzen Sie *localhost* durch die Angabe Ihres Providers:

```
$cfgServers[1]['user']='username';
```

Verwenden Sie in dieser Zeile den vom Provider genannten Username.

```
$cfgServers[1]['password']='password';
```

Ersetzen Sie *password* durch die Angabe Ihres Providers.

```
$cfgServers[1]['only_db']='datenbankname';
```

Ersetzen Sie *datenbankname* durch den Namen der Datenbank, die Ihr Provider für Sie eingerichtet hat. Verwenden Sie *phpMyAdmin* in Ihrer Testumgebung, können Sie diesen Eintrag leer lassen, also `''`, um alle Datenbanken des Servers angezeigt zu bekommen.

Hinweis

Wir empfehlen dringend, das Verzeichnis, in das Sie *phpMyAdmin* kopiert haben, zu schützen, da ansonsten jeder Internet-User mit einem Browser Ihre Datenbank bearbeiten und eventuell auch löschen kann!

Nachdem Sie diese Änderungen auf Ihrem Webserver gespeichert haben, können Sie im Browser *phpMyAdmin* aufrufen. Wählen Sie zunächst die richtige Sprache aus, danach werden Sie freundlich begrüßt.

Bild 9.1: phpMyAdmin begrüßt Sie.

Tabellen erstellen mit phpMyAdmin

Klicken Sie in der linken Spalte auf die vorhandenen Datenbanken, um mit *phpMyAdmin* eine neue Tabelle zu erstellen.

Geben Sie anschließend im Bereich NEUE TABELLE ERSTELLEN im Feld NAME den gewünschten Namen der neuen Tabelle an. Bestimmen Sie dann, wie viele Felder die neue Tabelle erhalten soll. In dem Bild 9.2 sehen Sie, dass es im Beispiel drei Felder sind.

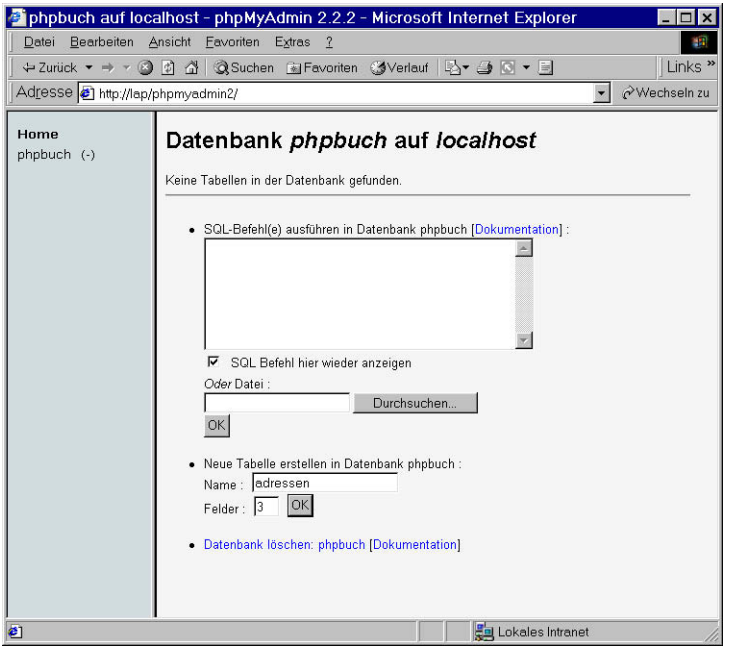

Bild 9.2: Bestimmen Sie die Anzahl der Felder und den Namen der Tabelle.

Anschließend füllen Sie das vorgefertigte Formular entsprechend dem Bild 9.3 für die drei Felder aus.

Die Tabelle wird mit einem Klick auf SPEICHERN angelegt. Danach wird Ihnen der SQL-Befehl zum Erstellen der Tabelle angezeigt sowie – sofern alles glatt ging – eine Liste der in der Tabelle vorhandenen Felder.

Mit Hilfe der Aktionen in der Auflistung der Felder können Sie die Eigenschaften der einzelnen Felder bearbeiten. Klicken Sie nun im linken Bereich auf den Datenbanknamen (Bild 9.5). Sie erhalten eine Auflistung aller in der Datenbank enthaltenen Tabellen. Bisher dürfte dies nur die eben angelegte sein.

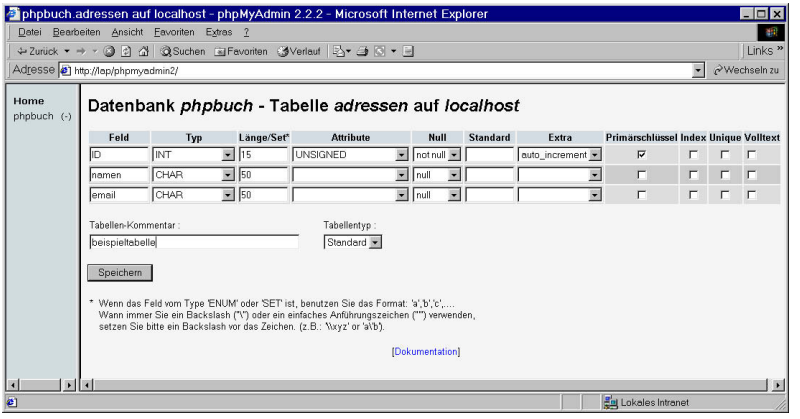

Bild 9.3: Die Definition der ersten Tabelle

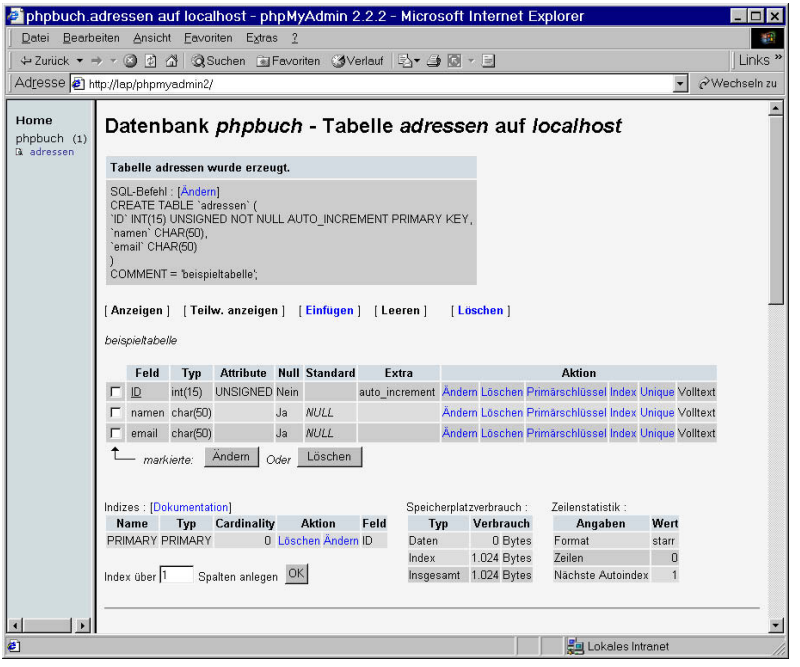

Bild 9.4: Die eben erstellte Tabelle

Bild 9.5: Die Datenbank phpbuch

Klicken Sie nun auf EINFÜGEN, um der Tabelle einen Datensatz zu gönnen.

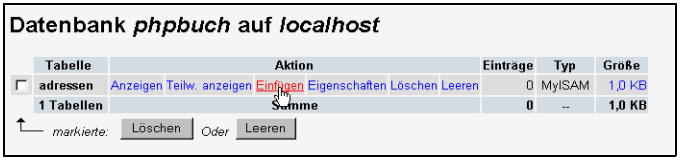

Bild 9.6: Einen Datensatz einfügen

Geben Sie im nachfolgenden Formular die Information des ersten Datensatzes ein. Speichern Sie die Eingaben mit einem Klick auf OK. Lassen Sie die Zeile für die ID leer, da es sich um einen AutoZähler handelt, den MySQL automatisch vergibt.

Bild 9.7: Einen Datensatz anlegen

Nachdem Sie einige Datensätze eingegeben haben, können Sie sie sich anzeigen lassen. Klicken Sie dazu auf ANZEIGEN in der Auflistung der Tabellen.

Bild 9.8: Die Datensätze der Tabelle auflisten lassen

Sie erhalten eine Liste aller Datensätze in der Tabelle. Sollten bereits viele Datensätze vorhanden sein, werden diese auf mehrere Seiten verteilt. Mit der Schaltfläche ZEIGE wechseln Sie zwischen den Seiten.

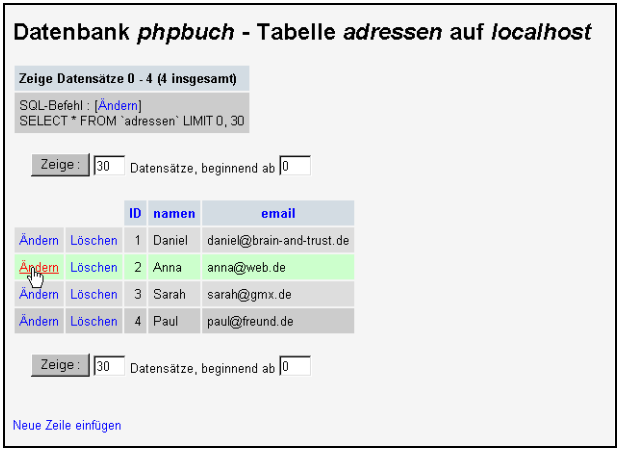

Bild 9.9: Die Auflistung der vorhandenen Datensätze

Um einen Datensatz zu bearbeiten, klicken Sie in der entsprechenden Zeile auf ÄNDERN. Im nachfolgenden Formular können Sie die Änderungen vornehmen und mit OK in der Datenbank speichern.

Der Umfang dieses Buches reicht nicht aus, um alle Funktionen von *phpMyAdmin* ausführlich zu beschreiben. Wir können Ihnen aber versichern, dass *phpMyAdmin* ein leistungsstarkes Werkzeug ist, das Ihnen den Umgang mit MySQL-Datenbanken wesentlich vereinfacht. Es bietet sich insbesondere an, um Datenbanken/Tabellen zu erstellen, Änderungen an Tabellenstrukturen oder Änderungen an einzelnen Datensätzen vorzunehmen.

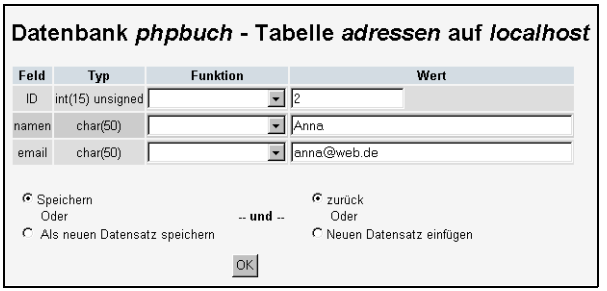

Bild 9.10: Einen Datensatz bearbeiten

Möchten Sie jedoch Informationen Ihrer Webseiten-Besucher sammeln, kommen Sie nicht umhin, Formulare zu erstellen und deren Eingaben per PHP-Script in die Datenbank zu schreiben.

Im folgenden Kapitel demonstrieren wir diesen Vorgang am Beispiel eines Datenbank-gestützten Gästebuchs.

Kapitel 10
Ein Gästebuch auf Datenbankbasis

In Kapitel 7 haben wir eine Seite mit einem Gästebuch programmiert, dessen Einträge in einer Textdatei gespeichert wurden. Da mittlerweile die Grundlagen von MySQL besprochen wurden, wollen wir uns an ein Gästebuch auf Datenbankbasis herantrauen. Die einzelnen Einträge werden in einer Tabelle gespeichert. Für das ganze Projekt erstellen Sie drei Seiten:

- Wie schon zuvor wird ein Formular benötigt, um neue Einträge in ein Gästebuch zu schreiben. Diese Einträge werden nach dem Abschicken des Formulars in die Tabelle der Datenbank gespeichert. Allerdings wollen wir dafür sorgen, dass die neuen Einträge in das Gästebuch erst nach Durchsicht für die Allgemeinheit sichtbar werden, damit unschöne Bemerkungen – wie sie von einigen »Scherzbolden« immer wieder in Gästebücher geschrieben werden – unterdrückt werden können. Nach dem Absenden eines neuen Gästebucheintrags erhält der Besucher der Seite also einen Hinweis, dass sein Eintrag empfangen wurde und in Kürze freigeschaltet wird.
- Um diesen Eingriff möglich zu machen, benötigen Sie eine Administratorenseite, auf der Sie die Gästebucheinträge sichten und freischalten oder auch löschen können.
- Selbstredend darf die eigentliche Gästebuchseite nicht fehlen, auf der die freigeschalteten Einträge aufgelistet werden, sodass alle Besucher der Seite einen Blick auf die Einträge werfen können.

Die Tabelle erstellen

Die Tabelle für das Gästebuch erstellen Sie am schnellsten und einfachsten mit dem Tool *phpMyAdmin*, das wir bereits im Kapitel 9 vorgestellt haben. Für unser Beispiel werden die folgenden Felder benötigt:

Name	Beschreibung
ID	Integer mit Autozähler zur eindeutigen Identifikation eines Datensatzes
absender	char – Textfeld, in dem der Absender seinen Namen oder Ähnliches angeben kann
betreff	char – Textfeld mit einer Länge von 255 Zeichen

Tabelle 10.1: Die Felder für das Gästebuch

Name	Beschreibung
message	Textfeld für den Eintrag im Gästebuch
datum	Datum des Eintrags
check	integer standardwert -1, wird auf 1 gesetzt, wenn der Datensatz/ Eintrag gesichtet und für OK befunden wurde

Tabelle 10.1: Die Felder für das Gästebuch (Forts.)

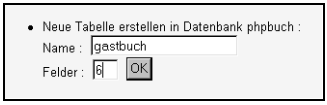

Bild 10.1: Eine neue Tabelle erstellen

Erstellen Sie also eine Tabelle mit 6 Feldern. Die einzelnen Attribute entnehmen Sie bitte dem Bild 10.2. Klicken Sie nach der Eingabe auf die Schaltfläche SPEICHERN.

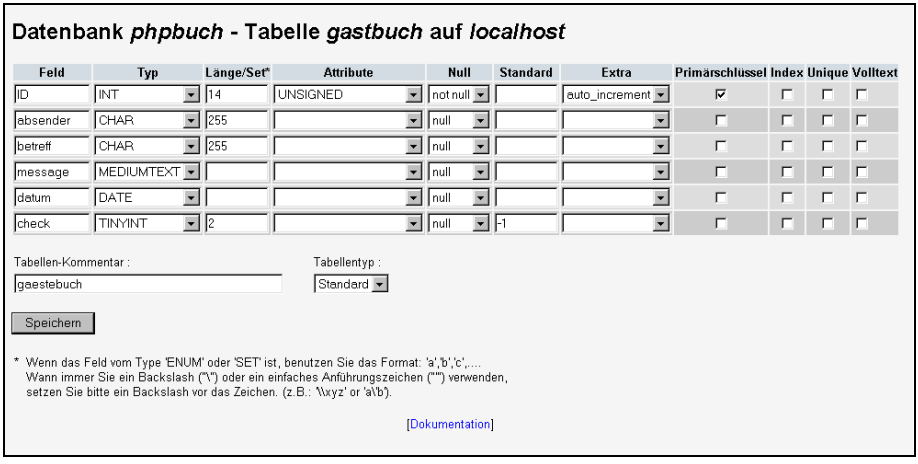

Bild 10.2: Die Felder der Tabelle anlegen

Das war's schon – fertig ist die Tabelle zum Speichern der Gästebucheinträge.

Das Formular zum Eingeben neuer Datenbankeinträge

Damit der User einen neuen Eintrag in das Gästebuch vornehmen kann, benötigen Sie ein Formular mit den Feldern, die ausgefüllt werden sollen. Das Formular soll sich selbst aufrufen, um nach dem Test der Eingaben entweder die Angaben zu speichern oder das Formular erneut anzuzeigen. Wie das Formular in etwa aussieht, zeigt das Bild 10.3.

Bild 10.3: Das Formular für das Gästebuch

Der Formularteil des Scripts

Der reine Formularteil sieht also im Script (Listing 10.1) folgendermaßen aus:

```
<form action='<?php echo $PHP_SELF; ?>' method=post>
<input type=hidden name=flag value=1>
<p>Ihr Name:<br>
<input type=text name=absender value='<?php echo $absender; ?>'>
<p>Betreff: <br>
<input type=text name=betreff value='<?php echo $betreff; ?>'>
<p>Ihre Nachricht:<br>
<textarea name=message>
<?php echo $message; ?>
</textarea>
<br>
<input type=submit>
</form>
</body></html>
```

Listing 10.1: Das Formular für das Gästebuch

Das Formular beinhaltet das versteckte Feld (hidden) mit dem Namen `flag`, um feststellen zu können, ob das Formular bereits ausgefüllt/abgeschickt wurde. Als vorbelegter Wert für den `value` der Formularfelder wird jeweils der Wert der korrespondierenden PHP-

Variablen ausgegeben, damit im Falle eines Fehlers, wenn also das Formular nochmals angezeigt wird, die bereits eingegebenen Werte angezeigt bleiben.

Außerdem soll berücksichtigt werden, dass das Formular nur beim ersten Aufruf oder im Falle eines Fehlers angezeigt wird. Dies erreichen wir wieder dadurch, dass das gesamte Formular in eine if-Anweisung gesetzt wird, die testet, ob `$flag` bereits gesetzt wurde. Folglich muss das Script so aufgebaut werden, wie in Listing 10.2:

```
<html>
<head>
<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>
<?php
if(!$flag)
{
?>
<form action='<?php echo $PHP_SELF; ?>' method=post>
<input type=hidden name=flag value=1>
<p>Ihr Name:<br>
<input type=text name=absender value='<?php echo $absender; ?>'>
<p>Betreff: <br>
<input type=text name=betreff value='<?php echo $betreff; ?>'>
<p>Ihre Nachricht:<br>
<textarea name=message>
<?php echo $message; ?>
</textarea>
<br>
<input type=submit>
</form>
<?php
}
?>
</body></html>
```

Listing 10.2: Listing: 10.2: Das Formular mit der if-Bedingung

Fehlermeldung und Speichern des Eintrags

Wenn das Formular abgeschickt wurde, müssen zunächst die Eingaben auf Fehler getestet werden. Sind Fehler vorhanden, soll eine Fehlermeldung und das gesamte Formular nochmals ausgegeben werden, sind keine Fehler vorhanden, soll der neue Gästebucheintrag gespeichert werden. Der User erhält also unter Umständen eine Seite wie in Bild 10.4 oder eine Erfolgsmeldung wie in Bild 10.5.

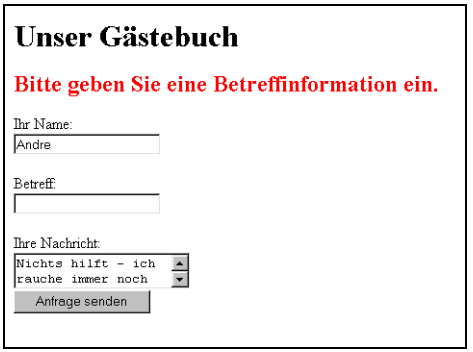

Bild 10.4: Bei unvollständigen Angaben soll eine Fehlermeldung auftauchen.

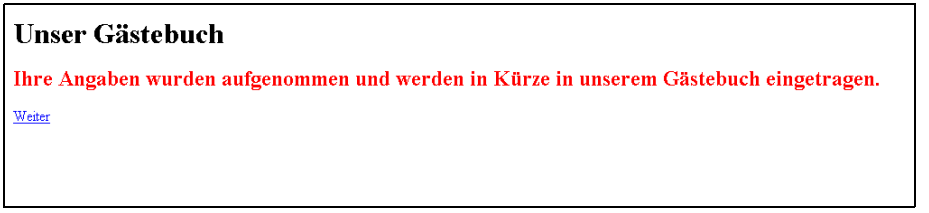

Bild 10.5: Eine Erfolgsmeldung nach dem Speichern des neuen Eintrags mit dem Link Weiter

Das Script, das für die Fehlermeldung sowie für das Speichern der Einträge in die Tabelle sorgt, entspricht dem Listing 10.3. Wie Sie sehen, enthält es Kommentare, sodass Sie an Ort und Stelle erkennen können, wozu bestimmte Befehle und Anweisungen dienen. Sie sehen auch an diesem Beispiel, dass Kommentare sehr sinnvoll sind, sobald ein Script etwas umfangreicher wird.

```
<html>
<head>
<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>
<?php

//$flag==1, wenn das Formular abgeschickt wurde
if($flag==1)
{
//Leerstellen entfernen
$absender=trim($absender);
```

```
$betreff=trim($betreff);
$message=trim($message);
//evt. vorhandenen HTML-Code entfernen
$absender=strip_tags($absender);
$betreff=strip_tags($betreff);
$message=strip_tags($message);

//testen, ob Felder ausgefüllt sind
if(!$absender){$fehler=”Bitte geben Sie eine Absenderinformation ein. <br>”;}
if(!$betreff){$fehler.=”Bitte geben Sie eine Betreffinformation ein. <br>”;}
if(!$message){$fehler.=”Bitte geben Sie eine Nachricht ein. <br>”;}

//Wenn Fehler vorhanden, die Textausgabe formatieren und $flag zurücksetzen,
damit das Formular erneut angezeigt wird
if($fehler)
{
$meldung="<h2><font color=red>".$fehler."</font></h2>";
unset($flag);
}
else//Es liegt kein Fehler vor, also Speichern der Informationen in der
Datenbank
{

//sql-String zusammenbauen
$tabellenname="gastbuch";
$sql="INSERT INTO $tabellenname (absender, betreff, message, datum) values
(‘$absender’,  ‘$betreff’, ‘$message’, now())";
//Verbinden zur Datenbank
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch”, $link);
mysql_query($sql, $link);
$meldung="<h2><font color=red>Ihre Angaben wurden aufgenommen und werden in
Kürze in unserem Gästebuch eingetragen.</font></h2><a
href=‘www.wohinauchimmer.de/seite.php’>Weiter</a>";
}

//Ausgeben der Meldung
echo $meldung;
}//Ende $flag==1
if(!$flag)
{
?>

<form action=‘<?php echo $PHP_SELF; ?>‘ method=post>
<input type=hidden name=flag value=1>
<p>Ihr Name:<br>
```

```
<input type=text name=absender value='<?php echo $absender; ?>'>
<p>Betreff: <br>
<input type=text name=betreff value='<?php echo $betreff; ?>'>
<p>Ihre Nachricht:<br>
<textarea name=message>
<?php echo $message; ?>
</textarea>
<br>
<input type=submit>
</form>

<?php
}
?>

</body></html>
```

Listing 10.3: Das Script mit eingebauter Fehlermeldung, Bearbeitung der Texteingaben, SQL-String und Speichern der Einträge in die Tabelle

Erläuterung des Scripts

Zunächst wird mit `if($flag==1)` getestet, ob das Formular abgeschickt wurde. Ist dies der Fall, werden die Angaben auf Fehler überprüft. Dabei werden erst einmal Leerstellen mit dem Befehlt `trim()` entfernt. Anschließend werden mit dem Befehl `strip_tags()` eventuell vorhandene HTML-Tags entfernt. Diese Bereinigung ist wichtig, weil Sie damit verhindern, dass Ihnen ein User automatische Umleitungen oder Ähnliches in das Gästebuch schmuggelt, die Ihre ahnungslosen Besucher der Seite auf die obskursten Seiten des Internets verlinkt.

Die Fehlermeldung

Sind diese beiden Aufgaben – Leerstellen und HTML-Tags entfernen – durchgeführt, wird getestet, ob alle Felder des Formulars ausgefüllt wurden. Sollte ein Feld nicht ausgefüllt worden sein, wird eine Fehlermeldung ausgegeben. Dafür sorgen die folgenden Programmzeilen:

```
if(!$absender){$fehler="Bitte geben Sie eine Absenderinformation ein. <br>";}
if(!$betreff){$fehler.="Bitte geben Sie eine Betreffinformation ein. <br>";}
if(!$message){$fehler.="Bitte geben Sie eine Nachricht ein. <br>";}
```

Sie sehen, dass wir hier wieder zu dem Trick greifen, die Fehlertexte hintereinander zu verketten, wenn mehrere if-Bedingungen erfüllt sind. Diese Methode haben wir bereits im Kapitel *Ein Gästebuch programmieren* angewandt, wo Sie eine ausführlichere Erklärung finden (blättern Sie also notfalls zurück). Mit Hilfe der `<br>`-Tags werden die Texte auf einzelne Zeilen platziert. Die endgültige Formatierung der Fehlermeldung erfolgt dann in der folgenden if-Anweisung:

```
if($fehler)
{
$meldung="<h2><font color=red>".$fehler."</font></h2>";
unset($flag);
}
```

In dieser if-Anweisung wird mit `unset()` auch `$flag` zurückgesetzt, damit das Formular erneut angezeigt wird, um die gefundenen Fehler korrigieren zu lassen.

Hinweis

`int unset(variablen)`

Die Funktion löscht eine Variable oder ein Array, so dass sie wieder vollständig freigegeben sind.

Die Angaben speichern

Wenn `$fehler` nicht gesetzt ist, also kein Fehler vorliegt, können die Angaben des Formulars in einer Tabelle gespeichert werden. Dies geschieht im else-Zweig:

```
else
{
$tabellenname="gastbuch";
$sql="INSERT INTO $tabellenname (absender,betreff,message,datum) values
(,$absender',  ,$betreff', ,$message', now())";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
mysql_query($sql, $link);
$meldung="<h2><font color=red>Ihre Angaben wurden aufgenommen und werden in
Kürze in unserem Gästebuch eingetragen.</font></h2>
<a href='www.wohinauchimmer.de/seite.php'>Weiter</a>";
}
```

Zunächst wird der SQL-String zum Speichern des neuen Datensatzes in der Tabelle zusammengesetzt und in der Variablen `$sql` gespeichert. Mit `INSERT INTO` weisen Sie die Datenbank an, einen neuen Datensatz in der Tabelle `$tabellenname` – also gastbuch – anzulegen. In der ersten Klammer geben Sie die Feldnamen der Tabelle an, denen Werte zugewiesen werden sollen. In der zweiten Klammer folgen die Werte, die entsprechend der Reihenfolge den Feldern zugewiesen werden sollen. Hierbei werden die Variablen aus dem Formular verwendet.

Sie müssen beachten, dass die Variablen-Werte in Anführungszeichen gesetzt werden, damit die Texte als solche von der Datenbank erkannt werden. Wichtig hierbei ist, dass Sie einfache Anführungszeichen verwenden, damit der Textstring, der `$sql` zugewiesen wird, nicht beendet wird und PHP einen Fehler ausgibt.

Mit `now()` weisen Sie die Datenbank an, das aktuelle Datum dem Feld DATUM zuzuweisen.

Hinweis

Dem Feld CHECK wird kein Wert explizit zugewiesen, da der Standardwert (-1) laut Definition der Tabelle verwendet werden soll.

Verbindung zur Datenbank

Damit der in `$sql` gespeicherte SQL-String an die Datenbank geschickt werden kann, muss eine Verbindung zur Datenbank hergestellt werden. Hierfür wird der Befehl `mysql_connect()` verwendet. Dieser Befehl erwartet den Servernamen der Datenbank, den Usernamen und das Passwort als Parameter. Diese Angaben müssen Sie naturgemäß entsprechend Ihres Datenbank-Servers anpassen. Um diese Verbindung ansprechen zu können, wird sie der Variablen `$link` zugewiesen.

Anschließend wird mit `mysql_select_db` eine Datenbank ausgewählt, die mit der Verbindung `$link` verwendet werden soll. In unserem Beispiel ist es die Datenbank *phpbuch*. Auch diese Angabe müssen Sie natürlich an Ihre Umgebung anpassen.

Mit dem Befehl `mysql_query()` schicken Sie den SQL-String, der in `$sql` gespeichert wird, an die Datenbank, die für die Verbindung `$link` festgelegt wurde.

Hinweis

An dieser Stelle könnten Sie eine Abfrage einbauen, die testet, ob der SQL-String ausgeführt wurde, da `mysql_query() false` zurückgibt, wenn ein Fehler beim Ausführen des SQL-Strings auftritt, und eventuell eine weitere Fehlermeldung anzeigen lassen (`if(!mysql_query($sql, $link)){echo "Datenbankfehler!!!";}`). In dem Beispiel wird auf diesen Fehlertest verzichtet, um das Beispiel nicht zu unübersichtlich werden zu lassen.

Erfolgsmeldung und Link setzen

Anschließend wird eine Erfolgsmeldung in der Variablen `$meldung` gespeichert. Hierbei ist es wichtig daran zu denken, dass Sie einen Link in diese Meldung integrieren, damit der Surfer einen Weg vorfindet, diese Seite zu verlassen. Im Beispiel wurde der Link `Weiter` kreiert.

```
href='www.wohinauchimmer.de/seite.php'>Weiter</a>";
```

Abschließend müssen Sie `$meldung` ausgeben, entweder gefüllt mit der Erfolgsmeldung, die dann solo auf der Seite steht oder mit den Fehlermeldungen, die vom Formular gefolgt werden, da dieses nochmals angezeigt wird.

Testen der Datenbank

Um zu testen, ob die ganze Sache geglückt ist und mithilfe des Formulars Daten in die Datenbank geschrieben werden, schicken Sie das Formular mehrfach ab. Lassen Sie sich anschließend die Datensätze der entsprechenden Tabelle in phpMyAdmin anzeigen.

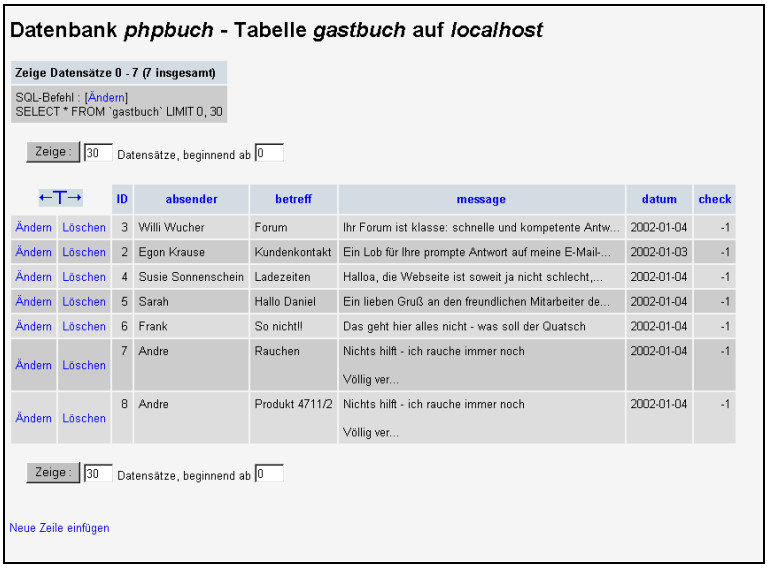

Bild 10.6: Die Einträge in der Tabelle gastbuch werden mit phpMyAdmin angezeigt.

Die Gästebucheinträge auflisten

Ihr Gästebuch möchten Sie sicherlich nicht nur allein im stillen Kämmerlein mit Hilfe von phpMyAdmin betrachten, sondern die freigeschalteten Einträge Ihren Webseitenbesuchern auch präsentieren. Für diesen Zweck wird jetzt eine Seite erstellt, die die freigeschalteten Einträge der Tabelle *gastbuch* auflistet. Das sieht im Browser dann in etwa aus wie in Bild 10.7.

Bild 10.7: Die Liste der freigeschalteten Einträge des Gästebuchs

Sollten sich irgendwann viele Einträge ansammeln, werden die Einträge mithilfe einer Erweiterung seitenweise aufgelistet. Diese Erweiterung wird im letzten Abschnitt (des Kapitels) hinzugefügt.

Bild 10.8: Teilen Sie umfangreiche Gästebücher auf mehrere Seiten auf.

Das Grundgerüst

Sie benötigen also zunächst einmal wieder das Grundgerüst einer HTML-Datei:

```
<html>
<head>

<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>

</body></html>
```

Jetzt erweitern Sie dieses Grundgerüst derart, dass zunächst die Einträge aus der Tabelle *gastbuch* ausgelesen und anschließend im `<body>`-Bereich der Datei ausgegeben werden. Das Grundgerüst sieht dann folgendermaßen aus:

```
<?php
//Auslesen der Einträge
?>
<html>
```

```
<head>
<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>
<?php
//Ausgeben der Einträge
?>

</body> </html>
```

Listing 10.4: Das erweiterte Grundgerüst

Auslesen der Einträge

Um die Einträge aus der Tabelle *gastbuch* auslesen zu können, müssen Sie zunächst wieder eine Verbindung zur Datenbank aufbauen und anschließend das entsprechende SQL-Statement an die Datenbank schicken.

```
$tabellenname="gastbuch";
$sql="SELECT absender, betreff, message, datum FROM
$tabellenname WHERE check=1 ORDER BY datum DESC ";
```

Mit `SELECT` werden zunächst Datensätze ausgewählt, wobei die nachfolgend gelisteten Felder zurückgegeben werden. Die Felder werden einfach hintereinander durch Komma getrennt in das Statement geschrieben. Nach `FROM` folgt die Tabelle, der die Datensätze entnommen werden sollen. Anschließend folgt nach `WHERE` die Bedingung, die die Datensätze erfüllen müssen, um ausgesucht zu werden.

Im Beispiel sollen nur die Einträge angezeigt werden, die bereits durchgesehen und für gut befunden wurden, bei denen CHECK also gleich 1 ist. Mit `ORDER BY` wird eine Spalte als Sortierkriterium angegeben. In diesem Fall ist es das Datum, da die neuesten Einträge an oberster Stelle stehen sollen. Mit `DESC` (absteigend) bzw. ASC (aufsteigend) wird die Sortierreihenfolge angegeben.

```
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
```

Alle Datensätze, die der Abfrage entsprechen, sind jetzt in der Ergebnisliste `$result`.

Anschließend schreiben wir die Datensätze der Einfachheit halber in ein mehrdimensionales Array, das die Bezeichnung `$ergebnis` erhält.

```
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}
```

Die Zuweisung des Datensatzes erfolgt mit `$ergebnis[$i]=mysql_fetch_array($result)`. `$i` wird mit der for-Schleife jeweils um eins hochgezählt, um die nächste Nummer für das Array `$ergebnis` zu erhalten.

Ausgabe der Daten

Nachdem die Daten in das Array `$ergebnis` geschrieben sind, können Sie über eine for-Schleife die Anzahl der Elemente anzeigen. Im Beispiel soll für jeden Datensatz eine kleine Tabelle angelegt werden.

```
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][message]." </td></tr></table>";
}
```

Der Quelltext

Listing 10.5 zeigt das komplette Script für die Ausgabe der Daten in einer Tabelle.

```
<?php
//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT absender, betreff, message, datum FROM $tabellenname WHERE check=1
ORDER BY datum DESC ";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}
?>
<html>
<head>
<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>

<?php

//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
```

```
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td>
<td> ".$ergebnis[$i][absender]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][message]." </td></tr></table>";
}
?>

</body> </html>
```

Listing 10.5: Das Script für die Ausgabe der Daten in einer Tabelle

Sollte ein Test bei Ihnen noch keine Einträge anzeigen, bedenken Sie bitte, dass zunächst der Wert im Feld CHECK der Datensätze per Standardwert auf -1 gesetzt wird. Somit werden die neuen Datensätze zunächst nicht angezeigt. Dieses Manko können Sie provisorisch umgehen, indem Sie in *phpMyAdmin* bei einige Datensätzen diese Werte auf 1 setzen oder das nachfolgende Formular zum Bearbeiten der Gästebucheinträge erstellen. Bild 10.9 zeigt das Gästebuch mit freigeschalteten Einträgen.

Unser Gästebuch

2002-01-04	Willi Wucher
Forum	
Ihr Forum ist klasse: schnelle und kompetente Antworten	
2002-01-04	Susie Sonnenschein
Ladezeiten	
Halloa, die Webseite ist soweit ja nicht schlecht, aber verzichtet doch auf einige Grafiken. Die Ladezeiten sind mitunter unerträglich. Gruß aus Berlin	
2002-01-04	Sarah
Hallo Daniel	
Ein lieben Gruß an den freundlichen Mitarbeiter der Hotline Sarah	
2002-01-04	Frank
So nicht!!	
Das geht hier alles nicht - was soll der Quatsch	
2002-01-04	Andre
Rauchen	
Nichts hilft - ich rauche immer noch Völlig verärgert mit freundlichen Grüßen aus Tütendorf	
2002-01-04	Andre
Produkt 4711/2	
Nichts hilft - ich rauche immer noch Völlig verärgert mit freundlichen Grüßen aus Tütendorf	
2002-01-03	Egon Krause
Kundenkontakt	
Ein Lob für Ihre prompte Antwort auf meine E-Mail-Anfrage Gruß aus Oberursel	

Bild 10.9: Das Gästebuch mit einigen freigeschalteten Einträgen

Die Seite zum Bearbeiten der Gästebucheinträge

Bevor die Datensätze im Gästebuch angezeigt werden, müssen Sie von Ihnen freigeschaltet werden. Dies geschieht, indem der Wert des Feldes CHECK des betreffenden Datensatzes auf 1 gesetzt wird. Weiterhin benötigen Sie die Möglichkeit, Datensätze zu löschen oder wieder zu sperren.

Auf die Möglichkeit, Gastbucheinträge zu bearbeiten, also den Text direkt ändern zu können, verzichten wir hier, da das Gästebuch den Gästen gehört. Wir möchten keine nachträglichen Manipulationen am Text unterstützen. Nur die Freigabe soll implementiert werden, da Gästebücher mitunter von böswilligen Surfern zum Beschimpfen von »Gott und der Welt« missbraucht werden. – Wenn Sie aber auf diese Manipulationsmöglichkeit nicht verzichten möchten, können Sie auch die Texteinträge Ihres Gästebuchs editieren. Verwenden Sie dazu z.B. einfach phpMyAdmin.

Bild 10.10: Die Einträge des Gästebuchs zum Bearbeiten und Freischalten/Sperren

Beginnen Sie eine neue Datei, wie gehabt mit dem Grundgerüst einer HTML-Seite:

```
<html>
<head>
<title>Gaestebuch editieren</title>
</head>
<body>
</body> </html>
```

In diesem Fall sollen alle Datensätze der Tabelle *gastbuch* ausgelesen werden, sodass das SQL-Statement (in Listing 10.5) von oben mit nur kleinen Änderung übernommen werden kann. Die WHERE-Bedingung kann entfallen, außerdem werden alle Felder benötigt, sodass die Feldauswahl mit einem * erfolgen kann.

```
<?php

//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT * FROM $tabellenname ORDER BY datum DESC ";
```

Anschließend wird die Verbindung zur Datenbank hergestellt und alle Ergebnisse in ein Array geschrieben:

```
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}

?>
```

Das fast komplette Script wird also so geschrieben wie das in Listing 10.5 dargestellte. Was noch fehlt, ist die Ausgabe der einzelnen Datensätze mit den Links, um die einzelnen Bearbeitungsmöglichkeiten aufrufen zu können und der Bereich, der die Bearbeitungen durchführt.

```
<?php
//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT * FROM $tabellenname ORDER BY datum DESC ";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
```

```
}
?>

<html>
<head>
<title>Gaestebuch editieren</title>
</head>
<body>

</body></html>
```

Listing 10.6: Das Script-Fragment zum Auslesen der Einträge

Die Ausgabe der einzelnen Datensätze wird in den `<body>`-Bereich integriert. Hier wird wieder fast wie oben eine for-Schleife durchlaufen, mit deren Hilfe je Datensatz eine Tabelle erstellt wird. Die Ausgabe wird allerdings um die Information des Feldes CHECK erweitert.

```
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr></table>";
}
```

Listing 10.7: Das Script-Fragment für die Ausgabe der Daten

Das Script sieht (fast) komplett aus wie in Listing 10.8 dargestellt. Was noch fehlt, ist die Ausgabe der Links, um die einzelnen Bearbeitungsmöglichkeiten aufzurufen und der Bereich, der die Bearbeitungen durchführt. Dieser Aufgabe wenden wir uns anschließend (im nächsten Abschnitt) zu.

```
<?php

//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT * FROM $tabellenname ORDER BY datum DESC ";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}
?>
```

```
<html>
<head>
<title>Gaestebuch editieren</title>
</head>
<body>

<?php
//Ausgabe der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr></table>";
}
?>

</body> </html>
```

Listing 10.8: Das Script für die Seite zum Bearbeiten der Gästebucheinträge

Löschen von Einträgen

Zunächst soll ein Bearbeitungslink integriert werden, der den entsprechenden Datensatz löscht. Der Link ruft die gleiche Seite erneut auf. Mit Hilfe der Get-Methode wird an die URL die ID des betreffenden Datensatzes angehängt und eine Variable `$flag` gesetzt, die anzeigt, dass der Datensatz gelöscht werden soll. Anhand des Wertes der Variablen `$flag` wird unterschieden, welche Aktion durchgeführt werden soll.

Im nächsten Abschnitt werden die Links zum Freischalten bzw. Sperren von Gastbucheinträgen erstellt. Anhand des Wertes von `$flag` wird dann die gewünschte Aktion unterschieden:

Ist `$flag = 1`, soll der Datensatz gelöscht werden, ist `$flag = 2`, soll der Datensatz freigeschaltet und bei `$flag = 3` gesperrt werden. Dieser Link wird in einer neuen Zeile an die Tabelle jedes Datensatzes angehängt:

```
<?php

//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr>";
echo "<tr><td colspan=3>
```

```
<a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=1'>Löschen</a>
</td></tr></table>";
}

?>
```

Listing 10.9: Das Script ergänzt um den Link zum Löschen eines Eintrags

Mit der neuen Zeile wird über `<a href='".$PHP_SELF.` sichergestellt, dass die aktuelle Datei aufgerufen wird, anschließend wird der Link noch mit `?ID=".$ergebnis[$i][ID]."&flag=1'>` um die benötigten Informationen erweitert: um die ID-Nummer, damit bekannt ist, welcher Datensatz gelöscht werden soll, und um `flag=1`, wodurch anzeigt wird, dass der Datensatz gelöscht werden soll. Abschließend folgt die Beschriftung des Links und das Ende-Tag: `Löschen </a>`.

Die Löschaktion einbauen

Nachdem nun der Link vorhanden ist, muss die gewünschte Aktion, also das Löschen des betreffenden Datensatzes, noch durchgeführt werden. Diese Aktion soll aber nur stattfinden, wenn `$flag==1` ist. Somit bietet sich eine if-Anweisung im oberen Bereich der Datei an:

```
If($flag==1)//Ein Datensatz soll gelöscht werden
{
$tabellenname="gastbuch";
$sql=" DELETE FROM $tabellenname WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
}
```

In der if-Anweisung muss das SQL-Statement generiert werden:

```
$sql=" DELETE FROM $tabellenname WHERE ID=$ID";
```

`DELETE FROM` zeigt an, dass ein oder mehrere Datensatz/sätze gelöscht werden soll/en. Anschließend folgt die Tabelle, aus der entfernt werden soll. Mit der `WHERE`-Bedingung wird der Datensatz spezifiziert. Im Beispiel ist es der Datensatz, dessen ID-Nummer im Feld ID der über die Get-Methode übermittelten ID – also `$ID` – entspricht.

Anschließend wird wieder die benötigte Verbindung zur Datenbank hergestellt und das SQL-Statement an die Datenbank übermittelt.

Mit diesen Ergänzungen sieht das Script nun wie folgt aus (Listing 10.10):

```
<?php

If($flag==1)//Ein Datensatz soll gelöscht werden
{
```

```
$tabellenname="gastbuch";
$sql=" DELETE FROM $tabellenname WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
}

//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT * FROM $tabellenname ORDER BY datum DESC ";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);

?>

<html>
<head>
<title>Gaestebuch editieren</title>
</head>
<body>

<?php

//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr>";
echo "<tr><td colspan=3> <a
href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=1'>Löschen</a></td></tr></
table>";
}
?>

</body> </html>
```

Listing 10.10: Das Script zum Bearbeiten der Einträge – ergänzt um die Aktion »Löschen«

Hinweis

Es ist wichtig, die Änderungen, also das Löschen und Freischalten oder Sperren, vor der Abfrage, die für die Auflistung benötigt wird, auszuführen, damit die Änderungen bei dieser Abfrage bereits berücksichtigt werden. Ansonsten sind die Änderungen erst bei wiederholtem Aufruf der Seite sichtbar. Die Änderungen wurden dann zwar durchgeführt, sie werden in der Auflistung aber nicht angezeigt.

Freischalten/ Sperren von Datensätzen

Nach dem gleichen Muster wie das Löschen von Datensätzen, kann man auch das Freischalten/Sperren der Datensätze bewerkstelligen. Zunächst wird ein entsprechender Link eingefügt und anschließend der Bereich des PHP-Scriptes, der die gewünschte Aktion durchführt. Hier ist aber eine Besonderheit zu beachten, da es sich bei diesen Aktionen mehr oder minder um einen Umschalter (Freischalten oder Sperren) handelt. Entsprechend ist die Beschriftung des Links abhängig vom Wert des Feldes CHECK und ebenso die anschließende Aktion.

Zunächst soll der Link erzeugt werden: Für die eben angesprochene Unterscheidung bietet sich wieder einmal ein kleine if-Anweisung an, die den derzeitigen Wert des Feldes CHECK prüft:

```
//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr>";
echo "<tr><td> <a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=1'>Löschen
</a></td><td></td>";

if($ergebnis[$i][check]==-1)
{echo "<td> <a
href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=2'>Freischalten</a></td></
tr></table>";}
else
{echo "<td> <a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=3'>Sperren</
a></td></tr></table>";}
}
```

Richten Sie Ihr Augenmerk auf die Zeile `if($ergebnis[$i][check]==-1)` Hier wird getestet, ob der Datensatz bisher gesperrt war. Ist dies der Fall, wird mit

```
{echo "<td> <a
href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=2'>Freischalten</a></td></
tr></table>";}
```

der Link zum Freischalten des betreffenden Datensatzes angezeigt. Ist der Datensatz bisher freigeschaltet, wird im else-Zweig der Link zum Sperren des Datensatzes angezeigt:

```
{echo "<td> <a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=3'>Sperren</
a></td></tr></table>";}
```

Nachdem die betreffenden Links integriert sind, werden in den oberen Teil der PHP-Datei die Bereiche gesetzt, die die entsprechenden Aktionen (Sperren oder Freischalten) ausführen. Wie schon beim Löschen, kommt hier wieder eine if-Anweisung zum Einsatz. Für das Freischalten sieht diese Anweisung folgendermaßen aus:

```
If($flag==2)//Ein Datensatz soll freigeschaltet werden
{
$tabellenname="gastbuch";
$sql=" UPDATE $tabellenname SET check=1 WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
}
```

Mit Hilfe des SQL-Statements wird der Wert des Feldes CHECK auf 1 gesetzt für den Datensatz, dessen ID-Nummer im Feld ID der übergebenen Variablen `$ID` entspricht.

Nach dem Zusammensetzten des SQL-Strings folgt die inzwischen bekannte Verbindung zur Datenbank und das Übermitteln des SQL-Statements.

Das Sperren eines Datensatzes sieht fast identisch aus, nur dass `$flag==3` sein muss, damit die if-Bedingung erfüllt ist, und dass der Wert des Feldes CHECK auf -1 gesetzt wird. Diese Variante wird dann folgendermaßen geschrieben:

```
If($flag==3)//Ein Datensatz soll gesperrt werden
{
$tabellenname="gastbuch";
$sql=" UPDATE $tabellenname SET check=-1 WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
}
```

Soweit haben Sie es geschafft! Der gesamte Quelltext inklusive der Möglichkeit der Editierung der Einträge wird aus Listing 10.11 ersichtlich.

```
<?php

If($flag==1)//Ein Datensatz soll gelöscht werden
{
```

```
$tabellenname="gastbuch";
$sql=" DELETE FROM $tabellenname WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch”, $link);
$result = mysql_query($sql, $link);
}

If($flag==2)//Ein Datensatz soll freigeschaltet werden
{
$tabellenname="gastbuch";
$sql=" UPDATE $tabellenname SET check=1 WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch”, $link);
$result = mysql_query($sql, $link);
}

If($flag==3)//Ein Datensatz soll gesperrt werden
{
$tabellenname="gastbuch";
$sql=" UPDATE $tabellenname SET check=-1 WHERE ID=$ID";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch”, $link);
$result = mysql_query($sql, $link);

}

//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT * FROM $tabellenname ORDER BY datum DESC ";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch”, $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}
?>

<html>
<head>
<title>Gaestebuch editieren</title>
</head>
<body>

<?php
```

```
//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td><td> ".$ergebnis[$i][check]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=3> ".$ergebnis[$i][message]." </td></tr>";
echo "<tr><td> <a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=1'>Löschen</
a></td><td></td>";
if($ergebnis[$i][check]==-1)
{echo "<td> <a
href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=2'>Freischalten</a></td></
tr></table>";}
else

{echo "<td> <a href='".$PHP_SELF."?ID=".$ergebnis[$i][ID]."&flag=3'>Sperren</
a></td></tr></table>";}
}

?>

</body></html>
```

Listing 10.11: Das gesamte Script für das Formular zum Bearbeiten der Einträge inklusive Lösch-, Freischaltungs- und Sperrmöglichkeiten

Schützen der Seite

Es ist auf jeden Fall ratsam, diese Seite vor unbefugtem Zugriff zu schützen, da ansonsten jeder Internet-User im Browser Ihr Gästebuch bearbeiten kann. Wir gehen mal davon aus, dass das nicht das »Gelbe vom Ei« wäre. Hierbei ist es kein ausreichender Schutz, wenn Sie keinen Link auf Ihrer Webseite zu dieser Datei integrieren, da moderne Suchmaschinen alle Verzeichnisse Ihrer Webpräsenz durchforsten. Als sicherste Methode gilt ein Verzeichnisschutz auf Basis des Webservers. Dies wird in Apache mithilfe der Dateien .htaccess und .htpasswd realisiert.

Sie können die Seite auch mit dem in Kapitel 14 vorgestellten Zugriffsschutz sicher vor Fremdbenutzung bewahren, ohne sich mit dem Apache-Webserver auseinandersetzen zu müssen.

Hilfreiche Erweiterungen

Im Prinzip kann jedes Script um mehr oder minder kleine Funktionen ergänzt werden, um es zu verbessern. Dies gilt auch für das eben erstellte. In diesem Abschnitt werden einige Möglichkeiten aufgezeigt, das Gästebuch durch nützliche Funktionen zu bereichern.

Eine E-Mail versenden bei einem neuen Gästebucheintrag

Wenn ein Surfer einen Eintrag in Ihrem Gästebuch schreibt, können Sie sich eine E-Mail schicken lassen, damit Sie den neuen Eintrag unverzüglich freischalten können. Sie ersparen sich damit häufige Kontrollblicke auf Ihr Gästebuch, ohne dass es neue Einträge gibt. Möchten Sie diese Möglichkeit integrieren, muss das Formular zum Erstellen neuer Einträge um die folgende Zeile ergänzt werden:

```
mail('ihre@emailadresse.de', 'Neuer Eintrag im Gaestebuch',
'Es liegt ein neuer Eintrag im Gaestebuch vor');
```

Passen Sie die E-Mail-Adresse an und ändern Sie die Texte nach Belieben. Fügen Sie diese Zeile dann hinter die Zeile, die für das Speichern des neuen Datensatzes in der Tabelle *gastbuch* sorgt, ein. Das nachstehende Listing 10.12 zeigt den entsprechenden Ausschnitt des Scripts:

```
else
//Es liegt kein Fehler vor, also speichern der Informationen in der Datenbank
{
//sql-String zusammenbauen
$tabellenname="gastbuch";
$sql="INSERT INTO $tabellenname (absender,betreff,message,datum) values
(,$absender',  ,$betreff', ,$message', now())";
//Verbinden zur Datenbank
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
mysql_query($sql, $link);
$meldung="<h2><font color=red>Ihre Angaben wurden aufgenommen und werden in
Kürze in unserem Gästebuch eingetragen.</font></h2><a
href='www.wohinauchimmer.de/seite.php'>Weiter</a>";
mail('ihre@emailadresse.de', 'Neuer Eintrag im Gaestebuch', 'Es liegt ein neuer
Eintrag im Gaestebuch vor');
}
```

Listing 10.12: Das Script, ergänzt durch die mail-Funktion

Hinweis

Sie können bei Befehlen Fehlermeldungen von PHP unterdrücken, indem Sie direkt vor den Befehl ein @-Zeichen schreiben. Im Falle des Mailversands sieht es dann folgendermaßen aus:

```
@mail('ihre@emailadresse.de', 'Neuer Eintrag im Gaestebuch', 'Es liegt
ein neuer Eintrag im Gaestebuch vor');
```

Formulare nur einmal abschicken

Ein häufig auftauchendes, leider nicht ganz einfach zu behebendes Problem ist es, zu kontrollieren, dass Formulare nur einmal abgeschickt werden. Das mehrfache Abschicken von Formularen geschieht im Allgemeinen aus zwei Gründen:

- Der User klickt mehrfach auf die Schaltfläche ABSCHICKEN DES FORMULARS, da die Antwortseite des Formulars nicht sofort angezeigt wird, weil der Server nicht schnell genug reagiert oder die Leitungen überlastet sind.
- Wenn ein User ein Formular abgeschickt und die Antwortseite bereits erhalten hat, werden die Formulardaten nochmals zum Server geschickt, sofern der User auf die Schaltfläche AKTUALISIEREN des Browsers klickt. Zwar erscheint bei den gängigen Browsern ein Hinweis (Bild 10.11), dass die Formulardaten nochmals übertragen werden, aber vielen Usern ist die Bedeutung dieser Meldung nicht bekannt.

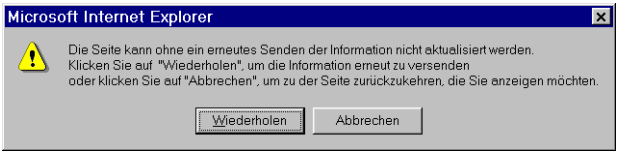

Bild 10.11: Der Browser gibt nach dem Klick auf Aktualisieren eine Meldung aus.

Die Lösung des Problems ist, wie bereits gesagt, nicht einfach. Wir werden hier aus Platzgründen nicht den Quelltext im Einzelnen behandeln, sondern nur kurz das Verfahren beschreiben, das häufig angewendet wird:

Beim ersten Aufrufen des Formulars wird eine eindeutige Zahl generiert. Hierzu können Sie die Funktion `uniqid()` verwenden. Diese Zahl wird in einem versteckten Feld in das Formular integriert. Weiterhin wird diese Zahl in einer Hilfstabelle gespeichert.

Beachten Sie, dass Sie diese Zahl nur beim ersten Aufruf des Formulars erzeugen dürfen. Sie dürfen diese Zahl nicht noch einmal neu berechnen, wenn das Formular erneut angezeigt wird, weil z.B. Fehler aufgetreten sind. Sind Fehler aufgetreten, müssen Sie die bereits erzeugte einmalige Zahl wieder verwenden.

Wird das Formular nun abgeschickt, lassen Sie nach der im versteckten Feld übermittelten Zahl in der Hilfstabelle suchen. Werden Sie fündig, löschen Sie den Datensatz in der Hilfstabelle und speichern die Formulardaten wie gewohnt ab. Ist diese Zahl in der Hilfstabelle nicht mehr enthalten, ist dies das Zeichen dafür, dass das Formular bereits einmal geschickt und gespeichert wurde, sodass Sie eine Fehlermeldung ausgeben und das Speichern der Formulardaten abbrechen können.

Zeilenumbrüche beibehalten

Die Eingabe der Nachricht für das Gästebuch erfolgt in einem Formularfeld des Typs `Textarea`. Bei diesen Feldern werden je nach Browser und gesetztem Attribut `wrap` (virtual, physical, off) Zeilenumbrüche mit übertragen. Diese Information wird in der Datenbank mit gespeichert, allerdings werden diese Zeilenumbrüche bei der Ausgabe nicht sichtbar, da der Browser nur HTML-Zeilenumbrüche mit dem Tag `<br>` erwartet und anzeigt.

Möchten Sie diese Zeilenumbrüche auch in der Auflistung der Gastbucheinträge wiederfinden, stellt PHP eine Funktion zur Verfügung, die die Zeilenumbrüche in `<br>`-Tags wandelt. Die Funktion heißt `nl2br()` und erwartet als Parameter den Text, der nach Zeilenumbrüchen durchforstet werden soll. Der Scriptbereich (Listing 10.13) zur Auflistung der Gastbucheinträge würde sich wie folgt ändern:

```
<?php
//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
$text=nl2br($ergebnis[$i][message]);
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=2> ".$text." </td></tr></table>";
}

?>
```

*Listing 10.13: Das Script-Fragment, ergänzt um die Funktion nl2br(), sodass Zeilenumbrüche in
-Tags verwandelt werden*

Bild 10.12: Die Eingabe mit Zeilenumbruch im Textarea-Bereich

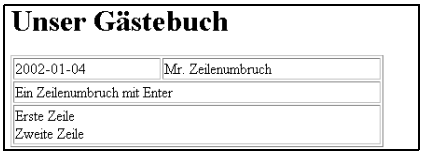

Bild 10.13: Die Wiedergabe des Eintrags mit nl2br

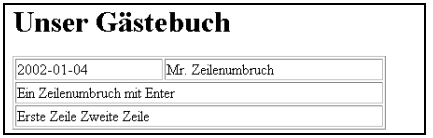

Bild 10.14: Die Wiedergabe des Eintrags ohne nl2br

Smileys ersetzen

Als nettes Feature können Sie in Ihrem Gästebuch integrieren, dass die typischen Tastaturkürzel für Smileys z. B. : –) durch entsprechende Grafiken ersetzt werden. Im Browser strahlt dann ein echter Smiley, wie in Bild 10.15 zu sehen.

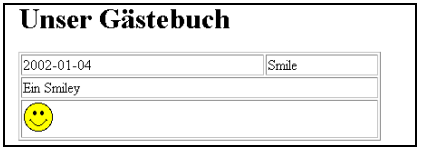

Bild 10.15: Ein Smiley im Gästebuch

Das Prinzip ist ganz einfach: Sie durchsuchen den Text der Einträge des Gästebuchs nach den Tastaturkürzeln und ersetzen diese durch einen Verweis auf die entsprechende Grafikdatei. Für das Ersetzen steht der Befehl `str_replace()` parat.

Dieser Befehl erwartet drei Parameter: der String, nach dem gesucht wird, der String der eingesetzt werden soll und der String, der durchsucht werden soll.

Angenommen, die Grafik mit dem lachenden Smiley liegt im gleichen Verzeichnis wie die Webseite und trägt den aussagekräftigen Namen smile.gif, sieht der geänderte Quelltext für die Wiedergabe von lachenden Smileys im Gästebuch folgendermaßen aus:

```
<?php

//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
$text=nl2br($ergebnis[$i][message]);
```

```
$text=str_replace(': -)', '<img src=smile.gif>', $text);
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=2> ".$text." </td></tr></table>";
}

?>
```

Listing 10.14: In dieser Form wird im Gästebuch gegebenenfalls ein Smiley angezeigt.

Wenn Sie mehrere Smileys austauschen möchten, müssen Sie die angepasste Zeile mit dem `str_replace`-Befehl mehrfach nacheinander einfügen.

Die Anzahl der angezeigten Gastbucheinträge begrenzen

Wie weiter oben bereits erwähnt, können Sie die Anzahl der je Seite angezeigten Gastbucheinträge begrenzen und gegebenenfalls Links einblenden, die in den Einträgen vor- und zurückblättern.

Diese Änderung nehmen Sie an der Webseite vor, die die Auflistung der Einträge enthält. Die Anzahl der von der Datenbanktabelle abgefragten Einträge können Sie mit einer Ergänzung des SQL-Statements begrenzen: Sie schränken die Abfrage mit `LIMIT` ein. `LIMIT` erlaubt zwei Parameter: der erste gibt den Startdatensatz an, der zweite die Anzahl der Datensätze. Mit anderen Worten: mit `LIMIT` sagen Sie der Datenbank: ich hätte gern ab Datensatz x die nächsten y Datensätze geliefert.

Der SQL-String ist entsprechend schnell angepasst, aber man benötigt noch die Funktionalität, dass man durch die Gesamtzahl der möglichen Einträge blättern kann. Dies erschwert die Sache etwas. Hier wird wieder einmal die gleiche Seite mit einem Link aufgerufen. Der Link zeigt an, wie viele Datensätze zuvor angezeigt wurden und ob vor- oder zurückgeblättert werden soll. Je nach Ergebnis wird `LIMIT` angepasst.

Listing 10.15 zeigt das entsprechend ergänzte Script:

```
<?php

$anzeigen=10;//Anzahl der Datensätze, die auf einer Seite angezeigt werden
sollen

if($vor){$start=$limit+$anzeigen;}
if($back){$start=$limit-$anzeigen;}
if(!$start){$start=0;}

//Auslesen der Einträge
$tabellenname="gastbuch";
$sql="SELECT absender, betreff, message, datum FROM $tabellenname WHERE check=1
ORDER BY datum DESC LIMIT $start , $anzeigen";
```

```
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$result = mysql_query($sql, $link);
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ergebnis[$i]=mysql_fetch_array($result);
}

//Ermitteln, wie viele Einträge vorliegen
$sql1="SELECT check FROM $tabellenname WHERE check=1 ";
$result1 = mysql_query($sql1, $link);
$max= mysql_num_rows($result1);

?>

<html>
<head>
<title>Gaestebuch</title>
</head>
<body>
<h1>Unser Gästebuch</h1>

<?php

//Ausgeben der Einträge
for($i=0;$i<count($ergebnis);$i++)
{
echo "<table border=1 width=400><tr><td> ".$ergebnis[$i][datum]." </td><td>
".$ergebnis[$i][absender]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][betreff]." </td></tr>";
echo "<tr><td colspan=2> ".$ergebnis[$i][message]." </td></tr></table>";
}

//Ausgeben der Links vor und zurück
echo "<table border=0 width=400><tr><td align=left>";
if($start>=$anzeigen){echo "<a
href='".$PHP_SELF."?limit=".$start."&back=1'>Zurück</a>";}
echo "</td><td align=right>";
if($start+$anzeigen<$max){echo "<a
href='".$PHP_SELF."?limit=".$start."&vor=1'>Weiter</a>";}
echo "</tr></table>";

?>

</body> </html>
```

Listing 10.15: Das Script zum Blättern durch die Gästebucheinträge

Bild 10.16 zeigt die Seite mit den Links zum Blättern.

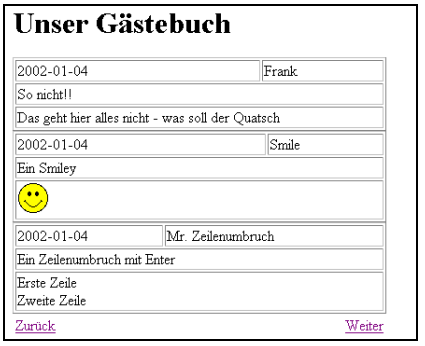

Bild 10.16: Die Links zum Blättern durch das Gästebuch

Erklärung

In der Variablen `$anzeigen` legen Sie fest, wie viele Datensätze je Seite angezeigt werden sollen. Ändern Sie diesen Wert nach Ihren Wünschen:

```
$anzeigen=10;
```

In der Variablen `$start` wird die Nummer des Datensatzes geschrieben, ab dem mit Hilfe von `LIMIT` im SQL-String die Datensätze angezeigt werden sollen. In `$limit` wird über die Links zum Blättern der Startwert der vorherigen Seite übergeben, sodass sich beim Vorblättern der neue Wert für `$start` aus dem alten Wert von `$start`, der in `$limit` übergeben wurde plus die bereits angezeigten Anzahl an Datensätzen, also `$anzeigen`, ergibt:

```
if($vor){$start=$limit+$anzeigen;}
```

Entsprechend muss beim Zurückblättern der neue Startwert in `$start` um die Anzahl der angezeigten Datensätze verringert werden:

```
if($back){$start=$limit-$anzeigen;}
```

Wird die Seite das erste Mal aufgerufen, ist weder `$vor` noch `$back` für das Vor- oder Zurückblättern gesetzt und somit wird auch `$start` kein Wert zugewiesen. Um die ersten Datensätze anzuzeigen, muss `$start` gleich Null sein:

```
if(!$start){$start=0;}
```

Nach dieser Vorbereitung wird der SQL-String angepasst:

```
$sql="SELECT absender, betreff, message, datum FROM $tabellenname WHERE check=1
ORDER BY datum DESC LIMIT $start , $anzeigen";
```

Mit LIMIT wird die Selection begrenzt, wobei $start den ersten anzuzeigenden Datensatz bestimmt. Entsprechend wurde $start zuvor berechnet. Mit $anzeigen wird die Anzahl der auszuwählenden Datensätze festgelegt.

Nun ist eine weitere Ergänzung notwendig, um die maximale Anzahl der Datensätze zu ermitteln. Dieser Wert wird gebraucht, um auf der letzten Seite beim Durchblättern den Link zum Weiterblättern nicht anzuzeigen.

```
$sql1="SELECT check FROM $tabellenname WHERE check=1 ";
$result1 = mysql_query($sql1, $link);
$max= mysql_num_rows($result1);
```

Es wird nur das Feld CHECK abgefragt. Welches Feld Sie hier verwenden, ist Ihnen überlassen, da nur die Anzahl der Datensätze relevant ist. Diese Anzahl wird mit mysql_num_rows() ermittelt und der Variablen $max zugewiesen. Es wird hier keine Verbindung aufgebaut, da von der vorherigen Abfrage die Verbindung noch besteht, so dass der SQL-String ohne Umschweife an die Datenbank geschickt werden kann.

Die Links zum Blättern werden in eine Tabelle geschrieben, wobei die eine Zelle links und die andere Zelle rechtsbündig ausgerichtet ist. Mit Hilfe der If-Bedingungen werden die entsprechenden Links nur angezeigt, wenn es sinnvoll ist, also der Link zum Zurückblättern nur, wenn noch eine vorhergehende Seite vorhanden ist und der Link zum Weiterblättern nur, wenn noch weitere Datensätze vorliegen. Das unten stehende Script-Fragment zeigt nochmals die eingebauten Links zum Blättern:

```
echo "<table border=0 width=400><tr><td align=left>";
if($start>=$anzeigen){echo "<a
href='".$PHP_SELF."?limit=".$start."&back=1'>Zurück</a>";}
echo "</td><td align=right>";
if($start+$anzeigen<$max){echo "<a
href='".$PHP_SELF."?limit=".$start."&vor=1'>Weiter</a>";}
echo "</tr></table>";
```

Listing 10.16: Die Tabelle mit den Links zum Blättern

Kapitel 11
Dateien hochladen

Sind Sie ein eifriger Surfer, dann sind Sie sicherlich schon mehr als einmal auf eine Seite gestoßen, die die Möglichkeit bietet, Dateien hochzuladen – wobei einschränkend zu erwähnen ist, dass es sich in der Regel um Bereiche handelt, die nur angemeldeten/eingeloggten Usern zur Verfügung steht. Das Hochladen von Dateien kann aus unterschiedlichen – und mehr oder minder sinnvollen – Gründen geschehen bzw. angeboten werden. In manchen Chatrooms findet man beispielsweise die Möglichkeit, ein Foto (von sich selbst) upzuloaden, sodass andere Teilnehmer wissen, wer SusieX und MartinY sind, ein typisches Beispiel sind auch die Seiten, auf denen Sie aufgefordert werden, Ihre Urlaubsfotos hochzuladen, sodass die Welt (oder eine eingeschränkte Gruppe) sich an ihnen erfreuen kann.

Auch im E-Commerce- und Dienstleistungsbereich wird oft die Möglichkeit eines Dateien-Uploads angeboten. Die Seiten, auf denen Auktionen stattfinden, und die Anbietern das zu versteigernde Produkt mit einem Foto schmackhaft machen, sind beispielsweise weit verbreitet.

Sie sehen, es ist – um auch in dieser rational-logischen Welt des Programmierens mit Fontane zu sprechen – ein weites Feld. Das Bild 11.1 zeigt eine typische Webseite zum Upload von Dateien.

Während es sich bisher so verhielt, dass der Upload von Dateien eigentlich nur mithilfe eines FTP-Programms ermöglicht wurde, können Sie mit neueren Browsern und PHP den Dateien-Upload auch via Browser durchführen. (Ganz so neu müssen die Browser übrigens gar nicht sein. Der Internet-Explorer 3 bzw. Netscape 2 sollten diese Aufgabe bereits ausführen können.) Deswegen widmen wir uns in diesem Kapitel der Darstellung und Erklärung der Programmcodes zum Hochladen von Dateien.

- Im ersten Abschnitt wird ein einfaches Formular zum Upload von JPG-Dateien erstellt. Die jeweils zuletzt hochgeladene Datei kann dann im Browser betrachtet werden.

  In den folgenden Abschnitten wird der Umgang mit der hochgeladenen Datei variiert, das Formular zum Upload bleibt dabei fast unverändert.
- Die Datei wird in einem Verzeichnis abgelegt. Sodann erstellen wir eine Linkliste, die alle Dateien in diesem Verzeichnis auflistet.
- Als Alternative können hochgeladene Dateien in einer Datenbank abgelegt werden. Um die in diesem Kapitel verwendeten Scripte überschaubar zu halten, und Sie zunächst generell mit dem Upload vertraut zu machen, folgt dieser Schritt erst im nächsten Kapitel *(Kapitel 12, Datei-Upload in eine Datenbank)*.

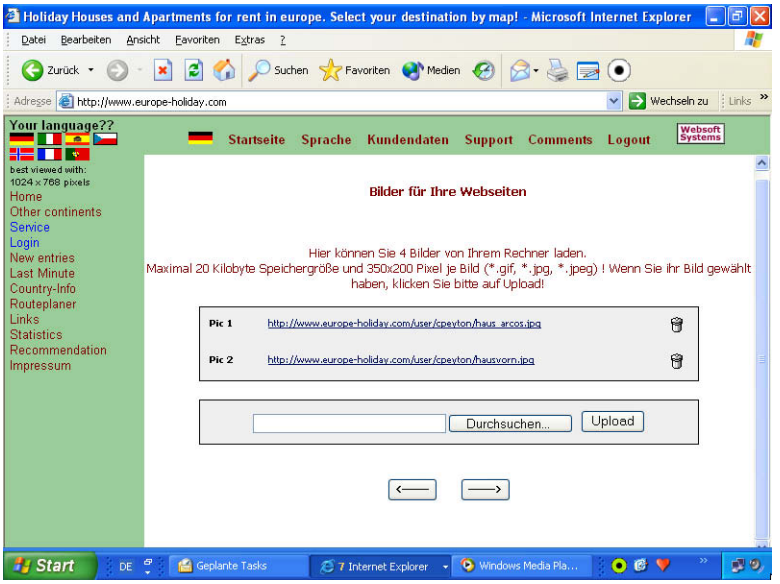

Bild 11.1: Eine Seite, die die Möglichkeit bietet, Fotos hochzuladen

Für hochgeladene Grafiken bietet es sich übrigens an, kleine Vorschaugrafiken zu erstellen, die auf einer Seite aufgelistet werden. Nach einem Klick auf die Vorschaugrafik wird dann die große Grafik in einem neuen Fenster geöffnet. Diese Darstellungsform bietet sich z. B. für ein digitales Fotoalbum an. Dieses Verfahren spielen wir im *Kapitel 13, Grafiken mit PHP*, durch.

Einschränkungen beim Upload

Der Upload via Browser unterliegt in PHP einigen Beschränkungen. Die meisten Provider beschränken die Größe der Dateien serverseitig auf 2 Mbyte. Größere Dateien werden einfach ohne Fehlermeldung verworfen. Das ärgerliche an dieser Methode ist, dass zunächst die Datei hochgeladen wird, also die entsprechende Zeit vergeht, bis die Datei auf dem Server angelangt ist, bevor man über das Script eingreifen kann.

Einer weiteren Beschränkung unterliegen Sie insofern, als die maximal ausführbare Zeit der Scripte bei den meisten Providern begrenzt ist. Alles, was innerhalb dieser Zeit vom Script nicht empfangen und abgearbeitet werden kann, geht verloren bzw. bricht im Bearbeitungsprozess ab. Dieses Zeitlimit ist insbesondere für User mit langsamen Verbindungen hinderlich, da bereits der reine Upload einer mittelgroßen Datei die maximale Laufzeit überschreiten kann. Die maximale Laufzeit der Scripte ist bei den meisten Providern auf ca. 30 Sekunden beschränkt. Sie können zwar mit einem Befehl die maximale Laufzeit für einzelne Scripte verlängern, allerdings nur, wenn der Provider PHP nicht im abgesicherten

Modus betreibt, was aber fast immer der Fall ist. Nur zur Information: Der Befehl, der die maximale Laufzeit setzt, lautet: `set_time_limit()`. Als Argument erwartet der Befehl die Zeit in Sekunden.

Fazit: Aufgrund dieser Beschränkungen bietet sich der Upload via Browser nur für kleinere Dateien an. Wenn Sie umfangreiche Präsentationen oder PDF-Dateien, die die Größenbegrenzung überschreiten, hochladen möchten, kommen Sie um Ihren FTP-Client nicht herum.

Das Formular für den Upload

Zunächst wird das Formular erstellt, das dem Surfer die Möglichkeit gibt, eine lokale Datei auszusuchen, um diese an den Server zu schicken. Anschließend wird die Datei ergänzt, damit die hochgeladene Datei gespeichert und auf der Seite angezeigt wird. Damit die ersten Versuche nicht zu kompliziert sind, sorgen wir dafür, dass zunächst nur JPG-Dateien richtig angezeigt werden.

Achtung

Wenn Sie einfach nur eine Datei per Formular zum Server schicken, ohne die Datei anschließend per PHP zu speichern oder anderweitig zu bearbeiten, geht die Datei einfach verloren. Dies liegt daran, dass der Server die hochgeladene Datei temporär speichert, aber mit Beendigung des Scripts diese temporäre Datei wieder löscht. Die Datei muss also weiterverarbeitet werden.

Vorüberlegungen zum Formular

Das Formular zum Upload unterscheidet sich nur geringfügig von anderen Formularen. Der gewichtigste Unterschied ist der, dass Sie dem Server im `<form>`-Tag mitteilen, dass Dateien geliefert werden. Dies erreichen Sie mit einem besonderen Attribut: `enctype="mulitpart/form-date"`. Weiterhin benötigen Sie ein Formularelement, das die Möglichkeit zum Auswählen einer lokalen Datei anbietet. Dieses Element erzeugen Sie mit `<input type=file name="neuedatei">`.

Diese Angabe macht sich im Browser mit zwei Elementen bemerkbar: zum einen gibt es eine Textbox, in der nach dem Aussuchen des Elements der lokale Pfad und der Dateiname stehen, zum anderen werden Sie die Schaltfläche DURCHSUCHEN entdecken, mit der Sie den von Windows bekannten Dialog zum Öffnen von Dateien aufrufen. Mit Hilfe dieses Dialogs bestimmen Sie die lokale Datei, die hochgeladen werden soll. Bild 11.2 zeigt ein solches Formular.

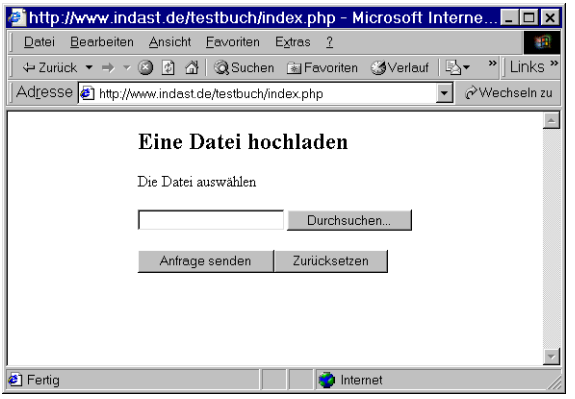

Bild 11.2: Das Formular zum Auswählen der Datei

Bild 11.3: Der Windows-Dialog, in dem Sie die lokale Datei aussuchen

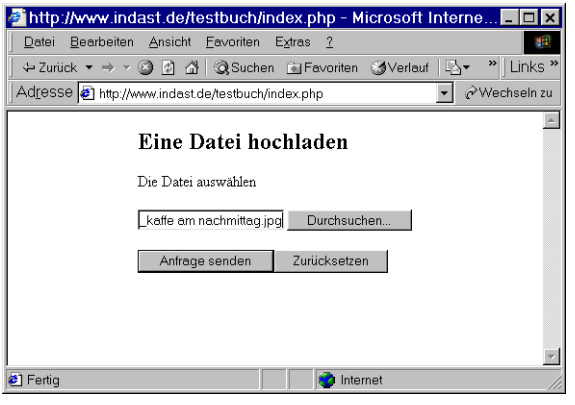

Bild 11.4: Der Pfad zur ausgewählten Datei im Textbereich

Der Quelltext für das Formular

Der Quelltext des Formulars sieht folgendermaßen aus (Listing 11.1):

```
<html><head>
<title>Upload</title>
</head><body>

<h2>Eine Datei hochladen</h2>

<form action='<? echo $PHP_SELF; ?>' method='post' enctype='multipart/form-
data'>
<input type="hidden" name="sent" value="1">
<p>Die Datei ausw&auml;hlen</p>
<input type=file name='neuedatei'><br><br>
<input type="submit"><input type="Reset" value"Zur&uuml;cksetzen">
</form>

</body></html>
```

Listing 11.1: Der Quelltext für das Formular zum Aussuchen der Datei

Erklärung des Quelltextes

Wie oben bereits erwähnt, handelt es sich um ein Formular, dem mithilfe der Angabe `enctype='multipart/form-data'` im `<form>`-Tag mit auf den Weg gegeben wird, dass unter anderem Dateien hochgeladen werden sollen. Als `action` im `<form>`-Tag wird wie in den meisten Fällen die Datei selbst aufgerufen, da das Verarbeiten der hochgeladenen Dateien später in dieses Script integriert wird. Auch das versteckte Feld mit dem Namen `sent` ist wieder vorhanden, um nach dem Absenden dem Script anzuzeigen, dass das Formular abgesendet wurde.

Mit Hilfe weiterer Attribute des Feldes `<input type=file>` können Sie das Verhalten des Browsers näher beeinflussen. Mit `size=30`, legen Sie z. B. fest, dass der Textbereich, in dem der Pfad zur ausgewählten Datei angezeigt wird, eine Größe von 30 Zeichen hat. Mit dem Attribut `value` können Sie einen Pfad für die lokale Datei vorbelegen. Des Weiteren gibt es noch die Attribute `maxlength`, mit dem Sie die maximale Größe der Datei in Bytes angeben können, und `accept`, mit dem Sie den Mime-Typ der hochladbaren Dateien einschränken können z. B. image/gif, text/plain oder application/msword.

Achtung

Beide Attribute – `maxlength` und `accept` – sind mit Vorsicht zu genießen, da die meisten Browser die mit diesen Attributen getroffenen Einschränkungen nicht umsetzen. Es ist also ratsam, solche Einschränkungen im Script, das die hochgeladene Datei weiterverarbeitet, zu implementieren.

Verarbeiten der hochgeladenen Datei

Nachdem eine Datei mithilfe des oben erstellten Formulars hochgeladen wurde, muss sie, wie bereits erwähnt, weiterverarbeitet werden, da sie sonst mit Beendigung der Sricptausführung gelöscht wird. Der Pfad zu der vom Webserver angelegten temporären Datei steht in der Variablen mit dem Namen des Feldes <input type=file>, im Beispiel also $neuedatei.

Informationen zur Datei

PHP stellt Ihnen weitere Informationen über die Datei zur Verfügung: Ergänzen Sie den Variablennamen mit einem Unterstrich (_) und size – also $neuedatei_size –, so erhalten Sie die Größe der Datei in Byte. In _name – also $neuedatei_name – wird der Originalname der lokalen Datei gespeichert. Der Mime-Type der Datei wird in _type – also $neuedatei_type – abgelegt.

Kopieren der Datei

Temporäre Dateien werden gelöscht. Um die hochgeladene Datei in Sicherheit zu bringen, müssen Sie die Datei kopieren. Dazu steht der Befehl copy() bereit. Dieser Befehl erwartet zwei Argumente: zum einen den Pfad zur Quelldatei (die temporäre Datei), die kopiert werden soll und zum anderen das Ziel, also die Angabe, wohin die Kopie geschrieben werden soll. Der Pfad wird in der Variablen mit dem Namen des Feldes <input type=file> gespeichert. Als Ziel können Sie zunächst einfach einen Namen vergeben, um die Datei in das gleiche Verzeichnis wie das Script zu schreiben.

Hinweis

```
int copy(string Quelle, string Ziel)
```

Die Funktion kopiert eine Quelldatei (Quelle) zu einer Zieldatei (Ziel).

Etwas komplizierter wird die ganze Geschichte, wenn Sie die Datei in ein anderes Verzeichnis schreiben und einen Namen generieren möchten – dazu im Abschnitt *Datei-Upload mit Linkliste* mehr. Der Befehl copy($neuedatei, 'datei.jpg') speichert also die hochgeladene Datei unter dem Namen *datei.jpg* in das gleiche Verzeichnis wie das Script.

Achtung

Achten Sie darauf, dass das Script auch Schreibrechte für das entsprechende Verzeichnis hat, ansonsten kann die Datei nicht angelegt werden.

Das Script zum Sichern der Datei

Zum Anfreunden mit dem Datei-Upload wird das oben erstellte Upload-Formular zunächst so modifiziert, dass die Informationen über die hochgeladene Datei ausgegeben werden. Anschließend werden wir die temporäre Datei in das gleiche Verzeichnis wie das

Formular kopieren. Hierbei wird immer ein fester Name verwendet, sodass die Datei unter dem Formular angezeigt werden kann, sofern es sich um JPG-Dateien handelt. Durch den festen Dateinamen überschreibt die zuletzt hochgeladene Datei immer den Vorgänger.

In einem weiteren Schritt wird der Upload auf JPG-Dateien beschränkt, damit garantiert ist, dass die Anzeige klappt, außerdem werden wir eine Größenbegrenzung integrieren.

Bild 11.6 gibt wieder, wie eine Seite im Browser aussieht (aussehen kann), die eine hochgeladene Datei sowie Informationen zu dieser Datei anzeigt.

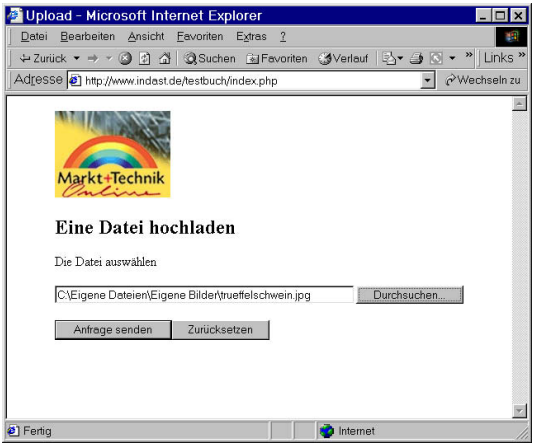

Bild 11.5: Eine zuvor hochgeladene Datei wird angezeigt, und eine neue Datei ausgewählt.

Bild 11.6: Die zuvor ausgewählte Datei wird nun angezeigt, über der Grafikdatei sehen Sie die Informationen dieser Datei.

Listing 11.2 zeigt das modifizierte Script (Listing 11.1). Es enthält nun die Attribute für die Informationen über die Datei, Fehlermeldungen und sorgt dafür, dass die Datei kopiert wird.

```
<html><head>
<title>Upload</title>
</head><body>

<?php

if($sent)
{
echo "Pfad zur tempor&auml;ren Datei: $neuedatei";
echo "<br>Gr&ouml;&szlig;e: $neuedatei_size";
echo "<br>Originalname: $neuedatei_name";
echo "<br>Typ: $neuedatei_type";

if($neuedatei_size>100000)
{$fehler="<br>Die Datei ist zu groß";}
if($neuedatei_type != 'image/pjpeg')
{$fehler.="<br>Der Dateityp ist nicht zulässig";}
if(!$fehler){copy($neuedatei, 'datei.jpg');}
else{echo "<h2><font color=red> $fehler </font></h2>";}

}
?>

<img src="datei.jpg">
<h2>Eine Datei hochladen</h2>

<form action='<? echo $PHP_SELF; ?>' method='post' enctype='multipart/form-
data'>
<input type="hidden" name="sent" value="1">
<p>Die neue Datei auswählen</p>
<input type=file name='neuedatei'><br><br>
<input type="submit"><input type="Reset" value"Zurücksetzen">
</form>

</body></html>
```

Listing 11.2: Ein einfacher Datei-Upload

Erklärung des Scripts

Wenn das Formular abgeschickt wurde, ist `$sent` gesetzt und die Informationen zur Datei werden mit `echo` ausgegeben:

```
echo "Pfad zur tempor&auml;ren Datei: $neuedatei";
echo "<br>Gr&ouml;&szlig;e: $neuedatei_size";
echo "<br>Originalname: $neuedatei_name";
echo "<br>Typ: $neuedatei_type";
```

Anschließend wird mit einer if-Anweisung die Größe der Datei getestet. In dem Fall, dass die Datei größer ist als 100000 Byte, wird eine Fehlermeldung in die Variable `$fehler` geschrieben.

```
if($neuedatei_size>100000)
{$fehler="<br>Die Datei ist zu groß";}
```

Um nur JPG-Dateien zuzulassen, wird der Typ der Datei getestet. Sollte der Mime-Type nicht JPG entsprechen, wird an die Variable `$fehler` eine weitere Fehlermeldung angehängt (achten Sie hier auf den zu verwendenden Vergleichsoperator `!=` für ungleich):

```
if($neuedatei_type != 'image/pjpeg')
{$fehler.="<br>Der Dateityp ist nicht zulässig";}
```

Sofern die beiden Fehlertests negativ waren, also kein Fehler vorliegt, und die Variable `$fehler` leer ist, wird die Datei kopiert. Ansonsten wird die Fehlermeldung, die in `$fehler` gespeichert ist, formatiert ausgegeben.

```
if(!$fehler){copy($neuedatei, 'datei.jpg');}
else{echo "<h2><font color=red> $fehler </font></h2>";}
```

Anschließend wird die Grafikdatei mit `<img src="datei.jpg">` angezeigt. Sollten Sie das Script das erste Mal aufrufen, taucht naturgemäß nur der Platzhalter für Grafiken auf, da noch keine Datei hochgeladen wurde. Ist ein Fehler aufgetreten, wird die zuvor hochgeladene Datei angezeigt (da der Kopiervorgang nicht durchgeführt wurde) und die entsprechende Fehlermeldung ausgegeben. Dies sehen Sie in Bild 11.7.

Achtung

Mitunter – je nach Einstellungen des Browsers – muss man aktualisieren, damit die soeben hochgeladene Datei angezeigt wird, da der Browser die Datei noch aus dem Cache liest. Leider lädt der Browser dann die gesamte Datei noch einmal hoch.

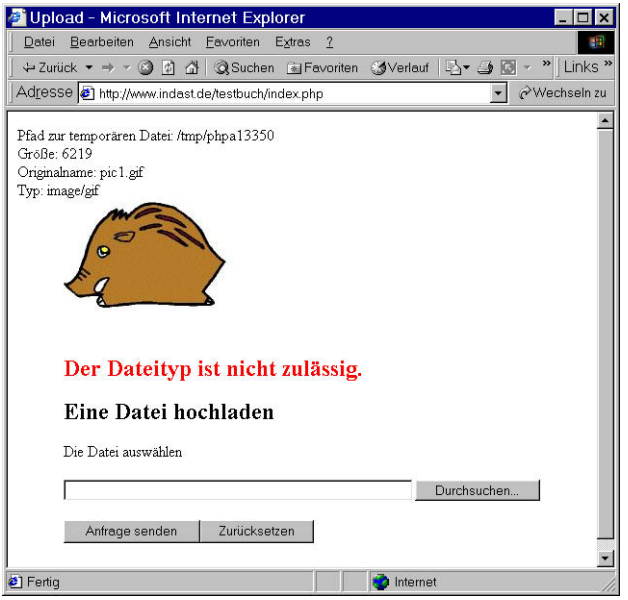

Bild 11.7: Der Versuch, eine GIF-Datei hochzuladen (zu sehen in der Zeile Typ), wird mit einer Fehlermeldung quittiert und das zuvor geladene Bild mit dem Trüffelschwein wird weiterhin angezeigt.

Sicherheitsbetrachtung

Sie sind gut beraten, wenn Sie den Upload von Dateien auf bestimmte Typen beschränken, da Sie Hackern ansonsten geradezu eine Einladungskarte in die Hand geben.

Die meisten Webserver verweigern im Webspace zwar das Ausführen von EXE- oder BATCH-Dateien, die sicherlich das größte Risiko darstellen, aber auch mit beispielsweise PHP- oder PEARL- Scripten, die hochgeladen werden, kann der geübte Programmierer auf alle Informationen Ihrer Webseite zugreifen und Ihre Seiten manipulieren. Unterbinden Sie den Upload dieser Scripte nicht, können Sie eigentlich gleich Ihre FTP-Zugangsdaten auf der Webseite veröffentlichen!

Da die findige Gemeinde der Hacker sich immer wieder neue Wege und Verfahren einfallen lässt, wie Webseiten zu knacken sind, ist es vergleichbar mit dem Wettlauf zwischen Hase und Igel, wenn Sie versuchen, auf dem Stand der Hackertricks zu bleiben und gefährliche Dateitypen zu sperren. Aus diesem Grund überlegen Sie sich gut, welche Dateitypen hochgeladen werden müssen und lassen Sie nur diese Dateitypen zu.

Mime-Types bestimmen und zulassen

Um den Mime-Typ der von Ihnen gewünschten Dateien zu ermitteln, können Sie meterlange Listen durchsuchen oder einfach eine entsprechende Datei hochladen und sich die Ausgabe von `$neuedatei_type` anschauen. Verwenden Sie zum Ausprobieren einfach das oben erstellte Script.

Nachdem Sie den Mime-Type auf diesem Weg herausgefunden haben, müssen Sie nur noch die if-Bedingung anpassen, die den Mime-Type testet. Möchten Sie z.B. zusätzlich zu JPG-Dateien auch noch GIF-Dateien zulassen, sieht die if-Anweisung folgendermaßen aus:

```
if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig";}
```

Um weitere Dateitypen zuzulassen, erweitern Sie die innerste Klammer der if-Bedingung um `OR $neuedatei_type == 'Mime-Type'`. Zusätzlich lassen Sie Worddokumente also zu mit folgender Ergänzung:

```
if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif' OR
$neuedatei_type == 'application/msword'))
```

Datei-Upload mit Linkliste

Nach diesem ersten einfachen Upload soll eine Linkliste der bisher hochgeladenen Dateien angeboten werden. Die hochgeladenen Dateien werden dazu in ein Unterverzeichnis kopiert. Im Gegensatz zum obigen Script dürfen Sie dabei die vorgehenden Dateien natürlich nicht überschreiben. Die Linkliste generieren Sie, indem Sie per PHP ein Array mit allen Dateinamen des Unterverzeichnisses erstellen und aus diesem Array eine Liste von Hyperlinks anzeigen lassen.

Hyperlinks zum Aufrufen der Dateien

Der Surfer kann die Dateien dann mit den Hyperlinks aufrufen. Je nach Dateityp werden beim Anklicken der Links unterschiedliche Aktionen ausgeführt. Wenn der Dateityp dem Rechner des Users unbekannt ist, wird ein Dialog zum Speichern der Datei angezeigt. Ist der Dateityp bekannt, wird die Datei in das temporäre Verzeichnis des User-Rechners geladen, und die Datei in dem zugeordneten Programm geöffnet. Hierbei zeigt Microsoft in den neueren Versionen des Internet-Explorers z.B. Worddateien im Browser an. Damit der Besucher Ihrer Seite die Dateien downloaden kann, steht im Kontextmenü des Hyperlinks der Eintrag ZIEL SPEICHERN UNTER oder VERKNÜPFUNG SPEICHERN UNTER (je nach Browser).

Das Formular zum Upload der Datei kann unverändert übernommen werden. Allerdings muss der Bereich, in dem die Datei gespeichert wird, angepasst und außerdem die eben angesprochene Linkliste generiert werden.

Bild 11.8: Die Upload-Seite mit der Linkliste. Im neuen Fenster ist die geöffnete, hochgeladene Grafik zu sehen.

Das Script für den Upload in ein Unterverzeichnis und die Linkliste

Nachfolgend sehen Sie in Listing 11.3 das vollständige Script; die ausführlichen Erklärungen dazu finden Sie in den folgenden Abschnitten.

```
<html><head>
<title>Upload</title>
</head><body>

<?php

//Namen des Unterverzeichnisses in eine Variable schreiben
$uverz="dateien";
if($sent)
{
//Wenn das Unterverzeichnis noch nicht existiert, dieses anlegen
if(!file_exists($uverz))
{mkdir($uverz,0755);}
```

```
//Dateigröße kontrollieren
if($neuedatei_size>100000)
{$fehler="<br>Die Datei ist zu groß";}

//Zulässige Dateitypen kontrollieren
if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig.<br>";}

//Datei speichern, wenn kein Fehler vorliegt
if(!$fehler)
{
//Eindeutigen Namen generieren
$name=uniqid("");

//Dateiendung je nach Typ an den Namen anhängen
switch($neuedatei_type)
{
case 'image/pjpeg':
$name.=".jpg";
break;
case 'image/gif':
$name.=".gif";
break;
}
//Den Dateinamen um die Pfadangabe erweitern
$name=$uverz."/".$name;

//Die Datei in das Unterverzeichnis kopieren
copy($neuedatei, $name);
}
else
{ echo "<h2><font color=red> $fehler </font></h2>";}
}

//Ausgeben der bereits vorhandenen Dateien im Unterverzeichnis
echo "<h2>Bisher hochgeladene Dateien</h2>";
$i=1;
$verzeichnis=opendir($uverz);
while ($file = readdir($verzeichnis))
{
if(is_file($uverz."/".$file))
{
echo "<a href='$uverz/$file' target=_blank>Datei $i</a><br>";
$i++;
}
}
```

```
closedir($verzeichnis);
?>

<h2>Eine neue Datei hochladen</h2>
<form action='<? echo $PHP_SELF; ?>' method='post' enctype='multipart/form-
data'>
<input type="hidden" name="sent" value="1">
<p>Die Datei ausw&auml;hlen</p>
<input type=file name='neuedatei' size=50><br><br>
<input type="submit"><input type="Reset" value"Zurücksetzen">
</form>

</body></html>
```

Listing 11.3: Das Script für den Upload und eine Linkliste der hochgeladenen Dateien, die in einem Unterverzeichnis gespeichert werden.

Das Verzeichnis für die Dateien

Die hochgeladenen Dateien sollen in einem Unterverzeichnis des aktuellen Ordners – also des Ordners, in dem das Script zum Upload der Dateien liegt – gespeichert werden. Dieser Ordner wird angelegt, falls er noch nicht existiert. Sie legen mit `mkdir()` neue Verzeichnisse an. Dieser Befehl erwartet zwei Argumente: Den Ordnernamen, wenn ein Unterordner des aktuellen Ordners erstellt werden soll, oder den absoluten Pfad inklusive neuem Ordnernamen und die Zugriffsrechte, die im UNIX-Dateisystem gewährt werden sollen. Lesen Sie für eine kurze Einführung in dieses Rechtesystem den anschließenden Exkurs.

Hinweis

```
int mkdir(string Pfadname, int modus)
```

Die Funktion legt ein Verzeichnis an.

Für jede Datei benötigen Sie einen eindeutigen Dateinamen, mit dem die Datei in das Verzeichnis kopiert wird.

Das UNIX-Rechtesystem

Bei UNIX kann man über die Rechtevergabe für den Eigentümer/Owner, die Gruppe/Group und den Rest der Welt/Public verschiedene Zugriffsrechte vergeben. Die Vergabe der Rechte bei `mkdir()` erfolgt über eine dreistellige Zahl, der eine Null vorangestellt wird. Die erste Ziffer gibt das Recht des Eigentümers/Owners, die zweite das der Gruppe/Group und die dritte das der Welt/Public an. Die Ziffern ermittelt man aus den drei möglichen Rechten Lesen/Schreiben/Ausführen, wobei jedem Recht ein Wert zugeordnet ist, der für jedes gewährte Recht addiert wird.

Werte für die einzelnen Rechte:

Lesen/Read = r = 4

Schreiben/Write = w = 2

Ausführen/Execute = x = 1

Rechenbeispiel:	User	Group	World	
drwxr-x-w-	r + w + x	r + – + x	- + w + -	0752
das entspricht	4 + 2 + 1 =7	4 + 0 + 1 =5	0 + 2 + 0 =2	0752

Beispiele:

Gesamt	User/Group/World	Oktal
drwxrwxrwx	rwx/rwx/rwx	0777
drwxr-xr-x	rwx/r-x/r-x	0755
drwx------	rwx/---/---	0700
drwxr-x---	rwx/r-x/---	0750
drwxr-xr--	rwx/r-x/r--	0754

Zunächst wird der gewünschte Name des Unterverzeichnisses in eine Variable geschrieben, damit der Name des Unterverzeichnisses bei gewünschten Änderungen nur einmal an dieser Stelle angepasst muss. Wichtig ist, dass diese Zuweisung vor der if-Anweisung steht, die `$sent` testet, da `$uverz` auch in der Auflistung der Dateien des Unterverzeichnisses verwendet wird, die ebenfalls angezeigt werden soll, wenn keine Datei gesendet wurde.

```
$uverz="dateien";
```

Anschließend wird getestet, ob das Unterverzeichnis bereits existiert. Für diesen Test wird, wie Sie sehen, auch der Befehl `file_exists` verwendet, da PHP in diesem Fall nicht zwischen Dateien und Ordnern unterscheidet.

Sollte das Unterverzeichnis nicht vorhanden sein, wird es mit `mkdir()` angelegt. Als Recht wird 0755 vergeben. Dies ist ein sehr weitreichendes Recht. Wir verwenden es hier, damit der Upload und das Anzeigen der Dateien auf jeden Fall funktioniert. Passen Sie das Recht gegebenenfalls auf die Einstellungen Ihres Webservers an. Das Verzeichnis wird nur beim ersten Upload, oder wenn Sie den Verzeichnisnamen geändert haben, angelegt.

```
if(!file_exists($uverz)){mkdir($uverz,0755);}
```

Die Größe der hochgeladenen Datei wird getestet und bei Überschreitung ein Fehlertext in die Variable `$fehler` geschrieben. Dies geschieht mit den beiden Programmzeilen:

```
if($neuedatei_size>100000)
{$fehler="<br>Die Datei ist zu groß";}
```

Nun testen Sie die erlaubten Dateitypen. Sollte die hochgeladene Datei nicht einem der erlaubten Typen entsprechen, wird die Variable `$fehler` um einen weiteren Fehlertext erweitert.

```
if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig.<br>";}
```

Liegt kein Fehler vor, geht es an das Speichern der hochgeladenen Datei. Zunächst wird mit der Funktion `uniqid()` ein eindeutiger Name für die Datei erzeugt und der Variablen `$name` zugewiesen.

Die Funktion `Uniqid()` arbeitet mit folgendem »Trick«: sie generiert den eindeutigen Namen auf Basis der Microsekunden zum Zeitpunkt des Aufrufs. Wenn Sie mehrere Aufrufe der Seite pro Microsekunde erwarten, ist es ratsam, den Befehl weiter auszureizen (über die Beschreibung in der Befehlsreferenz können Sie sich näher informieren). `Uniqid()` wird mit einer leeren Zeichenkette aufgerufen, da, wie Sie dem Hinweis entnehmen können, das Präfix kein optionaler Parameter ist, es braucht also eine Angabe.

Hinweis

```
int uniqid(string prefix [,boolean lcg])
```

Die Funktion erzeugt eine eindeutige ID mit dem angegebenen Präfix.

```
if(!$fehler)
{
$name=uniqid("");
```

Der eindeutige Name wird mit Hilfe einer switch-Anweisung um die Dateiendung erweitert. Je nach Mime-Type, der in `$neuedatei_type` gespeichert ist, wird die Dateierweiterung an `$name` angehängt. Wenn Sie andere oder zusätzliche Dateitypen zulassen wollen, müssen Sie diese `switch`-Anweisung anpassen bzw. um weitere `cases` ergänzen. Denken Sie daran, im Falle einer Änderung den Test auf die zulässigen Dateitypen auch mit anzupassen.

```
switch($neuedatei_type)
{
case 'image/pjpeg':
$name.=".jpg";
break;
case 'image/gif':
```

```
$name.=".gif";
break;
}
```

Nach der Ergänzung des Dateinamens mit der Dateiendung muss für den Kopiervorgang noch die relative Pfadangabe hinzugefügt werden, damit die Datei mit dem `copy`-Befehl auch in das Unterverzeichnis kopiert wird.

```
$name=$uverz."/".$name;
copy($neuedatei, $name);
```

Die Linkliste erstellen

Die Auflistung der vorhandenen Dateien mit Hyperlinks auf die einzelnen Dateien sollte nach dem Speichern generiert werden, damit die zuletzt hochgeladene Datei in der Auflistung mit enthalten ist.

Beachten Sie: Damit die Auflistung bei jedem Aufruf der Datei angezeigt wird, auch wenn zuvor keine Datei hochgeladen wurde, dürfen Sie diesen Scriptteil nicht innerhalb der if-Anweisung, die `$sent` testet, schreiben.

Sie können den Upload – also das Formular und den Scriptteil, der die hochgeladenen Datei speichert – und diese Auflistung auch auf zwei verschiedene Dateien aufteilen. Achten Sie dann aber darauf, dass beide Dateien im gleichen Verzeichnis liegen, damit die relativen Pfadangaben noch stimmen.

Da die Dateinamen beim Speichern zufällig erzeugt werden, sind diese Namen nicht sehr aussagekräftig. Aus diesem Grund wird für die Links einfach eine durchnummerierte Liste verwendet. Die Links öffnen die Dateien in einem neuen Browserfenster.

Möchten Sie Kurzbeschreibungen zu jeder Datei mit angeben, müssen Sie auf die im nächsten Kapitel beschriebene datenbankbasierte Lösung zurückgreifen.

Die Auflistung wird mit einer kurzen Überschrift eingeleitet.

```
echo "<h2>Bisher hochgeladene Dateien</h2>";
```

Mithilfe der Variablen `$i` werden die Links zu den Dateien im Unterverzeichnis durchnummeriert. Da die meisten Menschen es gewöhnt sind, bei eins zu zählen zu beginnen – außer Programmierer, die lieber bei `0` beginnen –, wird `$i` dieser Wert zugewiesen.

```
$i=1;
```

Um den Inhalt eines Verzeichnisses auszulesen, muss ähnlich wie bei Dateien mit `fopen()` zunächst mit `opendir()` ein Handle auf das Unterverzeichnis erstellt werden. Das Handle wird aufgrund der Zuweisung mit `$verzeichnis` angesprochen.

Hinweis

```
int opendir(string Pfad)
```

Die Option (vergleichbar mit fopen) erzeugt ein Handle auf ein Unterverzeichnis (Pfad).

Mit diesem Handle sagen Sie PHP: Hallo PHP, ich habe die Absicht, gleich was mit dem Verzeichnis xy anzustellen. Bereite dich bitte schon einmal darauf vor.

```
$verzeichnis=opendir($uverz);
```

Der Befehl `readdir()` liest alle Einträge eines Verzeichnisses aus. `readdir()` arbeitet vergleichbar zu `fgets()`. So wie `fgets()` Zeile für Zeile einer Datei einliest, liest `readdir()` jeweils einen Eintrag des Verzeichnisses aus, wobei bei jedem erneuten Aufruf automatisch der nächste Eintrag des Verzeichnisses zurückgegeben wird. Ist `readdir()` am Ende der Einträge angelangt, gibt der Befehl `false` zurück. Aus diesem Grund bietet sich eine While-Schleife an, um alle Einträge aufzulisten. Die Zuweisung des Verzeichniseintrags zu einer Variablen – im Beispiel `$file` – kann gleichzeitig als Abbruchbedingung verwendet werden, da am Ende des Verzeichnisses `false` zurückgegeben und somit die while-Schleife beendet wird.

```
while ($file = readdir($verzeichnis))
{
```

Hinweis

```
string readdir(int Handle)
```

Die Funktion liest den gesamten Inhalt eines Verzeichnisses aus.

Leider ist `readfile()` nicht exakt auf unsere Wünsche zugeschnitten, da es sowohl Dateien als auch Verzeichnisse ausliest. Damit die Ausgabe auf Dateien beschränkt wird, wird jeder gefundene Eintrag des Unterverzeichnisses `$uverz` noch einmal darauf getestet, ob es sich wirklich um eine Datei handelt. Dieser Test geschieht mit `is_file()`. Dieser Befehl gibt `true` zurück, wenn eine Datei vorliegt. Als Argument wird `is_file()` der relative Pfad inklusive Dateiname übergeben.

```
if(is_file($uverz."/".$file))
{
```

Wenn dieser Test ergeben hat, dass eine Datei vorliegt, kann der Hyperlink zusammengebaut werden. Mit `$uverz/$file` wird der relative Pfad als Ziel des Hyperlinks erstellt. Die Angabe `target=_blank` sorgt übrigens dafür, dass die Datei in einem neuen Fenster geöffnet wird.

```
echo "<a href='$uverz/$file' target=_blank>Datei $i</a><br>";
```

`$i` wird je Schleifendurchlauf hochgezählt, aber nur, wenn eine Datei vorliegt und ein Hyperlink ausgegeben wird, damit die Nummerierung der Dateien gesichert ist. Am Ende teilen Sie mit `closedir($verzeichnis)` PHP mit, dass Sie das Verzeichnis nicht länger bearbeiten möchten.

```
$i++;
}
}
closedir($verzeichnis);
```

Hinweis

```
bool is_file(string dateiname)
```

Diese Funktion testet, ob eine Datei (dateiname) existiert und ob es sich tatsächlich um eine Datei handelt.

Hinweis

```
void closedir (int Handle)
```

Die Funktion schließt ein Verzeichnis (Handle).

Löschen der hochgeladenen Dateien

Nachdem Sie Dateien per Browser hochladen können, liegt es nahe, diese Dateien auch per PHP-Script löschen zu können. Für diese Aufgabe wird wiederum eine Auflistung der vorhandenen Dateien benötigt, die je Datei um einen Link ergänzt wird, der die betreffende Datei in die »ewigen Jagdgründe« schickt. Dieser Link ruft dieselbe Datei erneut auf, nur dass bei diesem Aufruf ein Bereich des Scripts durchlaufen wird, der die gewünschte Datei löscht.

Für dieses Vorhaben, wird eine neue Datei erstellt, die allerdings auf bereits vorhandenen Scriptteilen aufbaut. Bild 11.9 zeigt die Dateiliste mit den Links zum Löschen.

Bild 11.9: Die hochgeladenen Dateien wieder löschen

Es ist ratsam, die Datei in ein gesondertes Verzeichnis zu legen, damit Sie dieses Verzeichnis mit einem Zugriffsschutz auf Webserverbasis versehen können. Im Beispiel wird davon ausgegangen, dass dieses Verzeichnis auf der gleichen Ebene liegt wie das Verzeichnis, in dem die Webseite zum Upload der Dateien liegt.

Das Script zum Löschen von Dateien

Das nachfolgend in Listing 11.4 gezeigte Script setzt voraus, dass das Script zum Upload der Dateien im Verzeichnis UPLOAD und die hochgeladenen Dateien im Unterverzeichnis DATEIEN des Verzeichnisses UPLOAD liegen. Das Script zum Löschen der Dateien muss in einem Ordner, der in der gleichen Hierarchieebene wie das Verzeichnis UPLOAD liegt, gespeichert sein, sodass der relative Verweis `../upload/dateien` in den Ordner mit den hochgeladenen Dateien zeigt.

Zum Löschen wird der Befehl `unlink()` verwendet. `Unlink()` erwartet als Argument nur den Pfad inklusive Dateinamen zu der zu löschenden Datei.

Hinweis

```
int unlink(string dateiname)
```

Die Funktion löscht eine Datei (dateiname) vom Server.

```
<html><head>
<title>Löschen von Dateien</title>
</head><body>

<?php

$uverz="../upload/dateien";

//Datei löschen
if($sent==1)
{
$name=urldecode($name);
$pfad=$uverz."/".$name;
@unlink($pfad);
}

echo "<h2>Bisher hochgeladene Dateien</h2>";
$i=1;
$verzeichnis=opendir($uverz);

while ($file = readdir($verzeichnis))
{
if(is_file($uverz."/".$file))
```

```
{
echo "<a href='$uverz/$file' target=_blank>Datei $i</a> - - ";
$file=urlencode($file);
echo "<a href='$PHP_SELF?sent=1&name=$file' >Datei $i löschen</a><br>";
$i++;
}
}
closedir($verzeichnis);

?>

</body></html>
```

Listing 11.4: Das Script zum Löschen von hochgeladenen Dateien

Erklärung des Scripts

Die folgenden Erklärungen zu dem Script beschränken sich auf Änderungen, die in den vorhergehenden Scripten zum Datei-Upload noch nicht behandelt wurden.

In der Variablen `$uverz` wird der relative Pfad zu dem Ordner mit den hochgeladenen Dateien gespeichert. Passen Sie diese Angabe gegebenenfalls an Ihre Verzeichnisstruktur an.

```
$uverz="../upload/dateien";
```

Die nachfolgenden drei Zeilen werden nur ausgeführt, wenn die Variable `$sent` gleich 1 ist. `$sent` ist dann 1, wenn das Script mit einem Link zum Löschen einer Datei aufgerufen wird. Mit diesem Aufruf wird auch der Dateiname der zu löschenden Datei mit übergeben. Diese Links werden weiter unten in der Auflistung der vorhandenen Dateien entsprechend erstellt.

Zunächst wird der Name der Datei, der mithilfe der eben erwähnten Links in der Variablen `$name` übergeben wird, mit `urldecode()` wieder dekodiert. Anschließend wird der komplette relative Pfad inklusive Dateiname zu der fraglichen Datei in der Variablen `$pfad` zusammengebaut.

Hinweis

```
string urldecode(string zeichenkette)
```

Die Funktion dekodiert einen string (zeichenkette), der vorher kodiert wurde.

Dann wird die fragliche Datei mit dem Befehl `unlink()` gelöscht. Dieser Befehl ist mit einem führenden @-Zeichen versehen, um eventuell auftretende Fehlermeldungen zu unterbinden. Diese Fehlermeldungen können auftreten, wenn die Datei aktualisiert wird und somit versucht wird, eine beim ersten Aufruf bereits gelöschte Datei ein weiteres Mal zu löschen.

```
$name=urldecode($name);
$pfad=$uverz."/".$name;
@unlink($pfad);
```

Die Auflistung der vorhandenen Dateien wird um zwei Zeilen ergänzt. Zunächst wird der Dateiname, der in `$file` gespeichert ist, mit dem Befehl `urlencode()` für den Fall der Fälle so kodiert, dass die Weitergabe via `GET`-Methode nicht an unerlaubten Zeichen im Dateinamen scheitert. `Encode` sorgt dafür, dass alle alphanumerischen Zeichen nach einem speziellen Muster kodiert werden, z.B. werden alle Leerstellen durch das Pluszeichen ersetzt.

Hinweis

`string urlencode(string name)`

Diese Funktion kodiert einen String (name) für die Übertragung via URL.

Anschließend wird ein Link zum Löschen der Datei ausgegeben. Dieser Link ruft das Script erneut auf, damit die Datei gelöscht wird. An die URL wird der kodierte Dateiname und `sent=1` gehängt, sodass beim erneuten Aufruf die if-Anweisung `$sent==1` erfüllt ist und die Datei mit dem entsprechenden Namen gelöscht werden kann.

```
$file=urlencode($file);
echo "<a href='$PHP_SELF?sent=1&name=$file' >Datei $i löschen</a><br>";
```

Ausblick: Upload von Textdateien

Die hier vorgestellten Techniken können Sie verwenden, um das in *Kapitel 6, Wechselnde Textausgabe* kurz angesprochene Verfahren (im Ausblick) zu nutzen, per Browser hochgeladene Textdateien in Ihrer Webseite einzubinden. Auf diese Art können Sie den Inhalt Ihrer Webseite schnell per Browser anpassen.

Sofern Sie diese Methode interessiert, müssen Sie nicht bei Null anfangen, sondern Sie können auf das in *Listing 11.2, Ein einfacher Datei-Upload,* gezeigte Script zurückgreifen.

Ein paar Änderungen und Anpassungen sind notwendig. Diese Änderungen betreffen die zulässigen Dateitypen, hier müssen Sie TXT-Dateien erlauben (`if($neuedatei_type != 'text/plain'))`, sowie den Namen, mit der die Datei gespeichert wird. Hier sollten Sie einen Dateinamen mit der Endung .txt verwenden.

Für die weitere Erklärung nennen wir die Datei *aktuell.txt* (`copy($neuedatei, 'aktuell.txt');`). Passen Sie gegebenenfalls den Speicherort mithilfe eines relativen Pfades an: `copy($neuedatei, '../pfad/aktuell.txt');`

Weiterhin können Sie das Script etwas abspecken und die Anzeige der Datei-Informationen und das Anzeigen der Datei als Image weglassen.

Nachdem Sie diese Änderungen an der Upload-Datei vorgenommen haben, bearbeiten Sie nun die Datei, in der der Text der Datei angezeigt werden soll. An der Stelle, an der der

Text erscheinen soll, legen Sie zunächst die Formatierungsinformationen für den Browser fest. Anschließend geben Sie mit Hilfe von `readfile()` den Dateiinhalt der TXT-Datei direkt an den Browser aus. Abschließend schließen Sie die Tags der Formatierungsinformationen.

```
echo "<h2><font color=green>";
readfile('aktuell.txt');
echo "</font></h2>";
```

Passen Sie, falls notwendig, die Pfadangabe zur TXT-Datei an. Übrigens können Sie die HTML-Tags zur Formatierung auch in die TXT-Datei schreiben – ganz nach Belieben. Bild 11.10 zeigt das Ergebnis im Browser und die Datei im Editor.

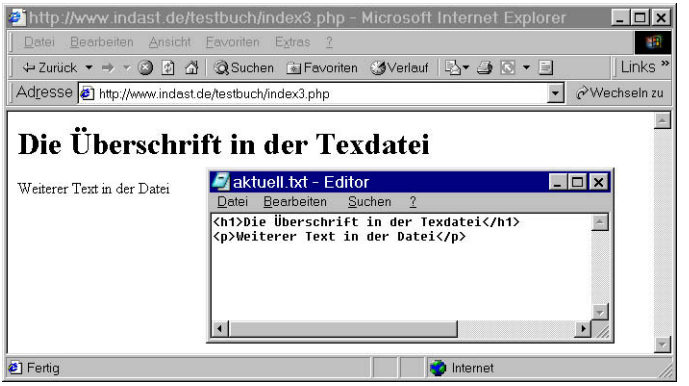

Bild 11.10: Die hochgeladene TXT-Datei im Editor und die Anzeige der über `readfile()` *eingebundenen Datei im Browser*

Kapitel 12
Datei-Upload in eine Datenbanktabelle

Das Speichern von hochgeladenen Dateien in einer Datenbank bietet einige Vorteile. Zusammen mit den Dateien können Sie beispielsweise weitere Informationen erfassen. Welche Informationen Sie zusätzlich zu einer Datei speichern, bleibt letztendlich Ihrem Einfallsreichtum und ihrer Informationsbegierde überlassen. Meistens werden eine Kurzbeschreibung, das Upload-Datum, die Dateigröße, die Person, die die Datei hochgeladen hat und der Dateityp zusätzlich ermittelt.

Die nachfolgenden Beschreibungen setzen voraus, dass Sie bereits einen Blick (oder auch zwei!) in das vorhergehende Kapitel über den Datei-Upload geworfen haben. Insbesondere mit den Grundlagen des Datei-Uploads, die im ersten Abschnitt des vorhergehenden Kapitels beschrieben werden, sollten Sie sich auseinandergesetzt haben, da dieses Verfahren hier nicht mehr ausführlich beschrieben wird. Weiterhin werden einige Scriptfragmente »recycled«, die in diesem Kapitel nur kurz erklärt werden.

- Für den Umgang mit hochgeladenen Dateien, die in Datenbanken gespeichert werden, benötigen Sie zunächst eine Webseite, auf der die lokale Datei in einem Formular ausgewählt und anschließend in der Datenbank gespeichert wird.
- Des Weiteren ist ein Script notwendig, das eine bestimmte Datei aus der Datenbank ausliest und zurückgibt. Was Sie mit der zurückgegebenen Datei anfangen, können Sie nach Bedarf variieren; entweder kann die Datei, sofern es sich um eine Grafik handelt, als Bild in Ihrer Webseite angezeigt werden, oder aber der Surfer kann die Datei öffnen bzw. downloaden.
- Um vorhandene Dateien in der Datenbank zu löschen bzw. die zusätzlichen Informationen zu bearbeiten, benötigen Sie eine weitere Webseite, mit deren Hilfe Sie diese Aufgaben erledigen.

Ein kleiner Tipp nebenbei: Sie kommen in den Genuss eines besonderen Vorteils, wenn Sie alle auf Ihrer Webseite verwendeten Grafiken (oder andere große Dateien) in einer Datenbank speichern, denn bei den meisten Providern wird der durch die Nutzung der Datenbank belegte Speicher nicht berechnet. Sie können also mit einem relativ günstigen Account mit wenig Webspace auskommen, wenn Sie die platzfressenden Grafiken in einer Datenbank speichern. Einige Provider sind allerdings auch hinterhältig! Nur ganz versteckt, teilsweise auch gar nicht, weisen sie auf eine maximale Datenbankgröße hin. Bei einem großen bekannten Provider hat uns die Suche nach der Information der maximalen Datenbankgröße eine gute halbe Stunde gekostet. Sie beträgt 100 MB (nur, um einmal einen Eindruck von der Größenordnung zu vermitteln).

Ein Nachteil sei hier aber auch gleich erwähnt. Wenn der Datenbankserver an seiner Leistungsgrenze betrieben wird, was bei vielen Providern der Fall ist, die verhältnismäßig günstigen Webspace anbieten, kann es zu Problemen beim Auslesen größerer Daten-

mengen kommen. Mitunter sind die Server derartig überlastet, dass der Server gar keine Verbindungen mehr akzeptiert, sodass Ihre Webseite mit Fehlermeldungen übersät wird.

Die Tabelle erstellen

Logischerweise besteht der erste Schritt wieder darin, eine passende Tabelle zu erstellen, in der die Dateien gespeichert werden. Sie müssen sich also überlegen, welche Felder die Tabelle zur Verfügung stellen soll. Naturgemäß benötigt diese Tabelle auf jeden Fall ein Feld für die Datei selbst. Zum Speichern von Binärdaten werden Felder des Typs BLOB verwendet. Die BLOB-Felder gibt es in unterschiedlichen Ausführungen, die sich in der Kapazität, also der speicherbaren Datenmenge, unterscheiden. Weiterhin wird ein eindeutiges ID-Feld benötigt, damit einzelne Datensätze zum Löschen, Bearbeiten oder Anzeigen eindeutig angesprochen werden können. Hierfür bietet sich ein INTEGER-Feld mit Autozähler an.

Die weiteren Felder der Tabelle sind abhängig von den zusätzlichen Informationen, die Sie gedenken zu speichern. Im Beispiel werden eine Kurzbeschreibung, der Dateityp, eine Kategorie, ein Text für das Hyperlink und das Speicherdatum erfasst. Sie benötigen also sieben Felder. Im Beispiel nennen wir die Tabelle *buchdateien*.

Feld	Speicherplatz in Byte
TINYBLOB	2^8
BLOB	2^16
MEDIUMBLOB	2^24
LONGBLOB	2^36

Tabelle 12.1: Der Speicherplatz der BLOB-Felder

Feld	Beschreibung
ID	Eindeutige über einen Autozähler generierter Integer Wert je Datensatz
dateityp	Textfeld (CHAR) zum Speichern des Mime-Type der Datei
datum	Das Datum, an dem die Datei hochgeladen wurde. Der Feldtyp ist DATE
linktext	Mit dem Hyperlink zum Aufrufen der Datei angezeigter Text. Das Feld hat den Typ CHAR

Tabelle 12.2: Die Felder der Tabelle buchdateien

Feld	Beschreibung
kategorie	Textfeld (CHAR) für die Zuordnung der Datei zu vorgegebenen Kategorien
beschreibung	Textfeld (CHAR) für eine kurze Beschreibung der Datei
datei	Die Daten der Datei. Das Feld hat den Dateitypen BLOB

Tabelle 12.2: Die Felder der Tabelle buchdateien (Forts.)

Die Tabelle erstellen Sie am einfachsten mit phpMyAdmin. Setzen Sie die Eigenschaften der Felder entsprechend dem Bild 12.1.

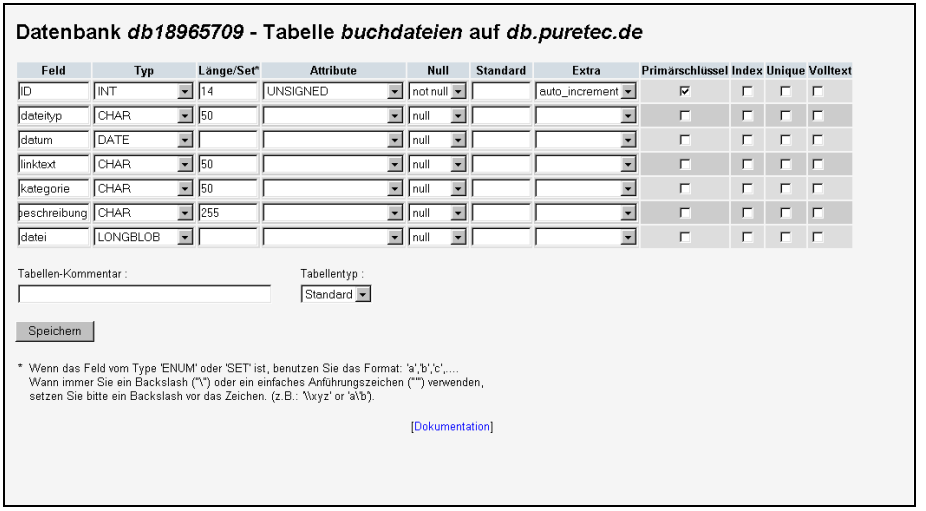

Bild 12.1: Die Felder der Tabelle zum Speichern der hochgeladenen Dateien

Sie können die Tabelle auch mit folgendem SQL-Statement erstellen:

```
CREATE TABLE `buchdateien` (`ID` INT(14) UNSIGNED NOT NULL AUTO_INCREMENT
PRIMARY KEY, `dateityp` CHAR(50), `datum` DATE, `linktext` CHAR(50), `kategorie`
CHAR(50), `beschreibung` CHAR(255), `datei` LONGBLOB);
```

Steht phpMyAdmin zur Verfügung, kann man nicht umhin festzustellen, dass die Verwendung dieses Tools deutlich angenehmer ist!

Das Upload-Script anpassen

Um die hochgeladenen Dateien nicht zu kopieren (wie wir es im vorherigen Kapitel gemacht haben), sondern in die Datenbanktabelle zu schreiben, müssen Sie die Verarbeitung der hochgeladenen Datei anpassen. Anstatt die Datei mit `copy()` zu kopieren, verwenden Sie ein SQL-Statement, das die Datei mit den zusätzlichen Informationen in die Datenbank schreibt.

Das Formular, das Sie zum Aussuchen der lokalen Datei verwenden, ergänzen wir lediglich um die Felder, die die zusätzlichen Informationen abfragen. Sie sehen das Formular in dem Bild 12.2.

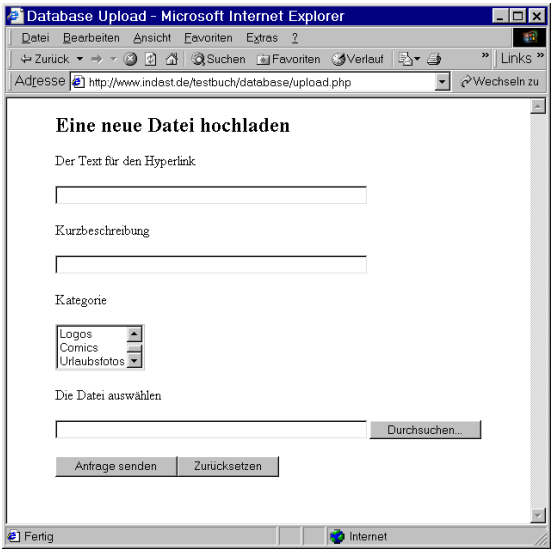

Bild 12.2: Das Formular zum Upload der Dateien

Der Programm-Code zum Schreiben der Datei in die Datenbank und für das Formular zum Auswählen

Sofern Sie das Beispiel des vorherigen Kapitels mitgespielt haben, können Sie bestimmte Teile aus dem Script für das Formular einfach kopieren und erneut benutzen. Listing 12.1 zeigt das Script zum Hochladen (Formular) und Speichern von Dateien in eine Datenbank.

```
<html><head>
<title>Database Upload</title>
</head><body>

<?php

if($sent)
```

```
{

if(!is_uploaded_file($neuedatei))
{$fehler="<br>Es wurde keine Datei hochgeladen";}
else
{

if($neuedatei_size>100000)
{$fehler.="<br>Die Datei ist zu groß;";}

if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig<br>";}

}//Ende ELSE keine Datei hochgeladen

if(!$fehler)
{
include('connect.php');
$datei=fopen($neuedatei,'r');
$data=addslashes(fread( $datei, $neuedatei_size));
$sql="INSERT INTO $tabellenname (linktext, beschreibung, kategorie, dateityp,
datum, datei) values ('$linktext',  '$beschreibung', '$kategorie',
'$neuedatei_type', now(), '$data')";

if(!mysql_query($sql, $link))
{$fehler.="<br>Der Upload ist aufgrund eines Datenbankfehlers gescheitert,
bittet versuchen Sie es später noch einmal<br>";}
}

if($fehler){ echo "<h2><font color=red> $fehler </font></h2>";}
else
{
echo "<h2><font color=blue>Der Upload war erfolgreich</font></h2>";
unset($linktext);
unset($beschreibung);
unset($kategorie);
}

}//Ende $sent==1

?>

<h2>Eine neue Datei hochladen</h2>

<form action='<? echo $PHP_SELF; ?>' method='post'
enctype='multipart/form-data'>
```

```
<input type="hidden" name="sent" value="1">
<p>Der Text für den Hyperlink</p>
<input type=text name='linktext' size=50 value='<? echo $linktext; ?>'><br>
<p>Kurzbeschreibung</p>
<input type=text name='beschreibung' size=50 value='<? echo $beschreibung;
?>'><br>
<p>Kategorie</p>
<select name='kategorie' size=3>
<option>Spanien
<option>Logos
<option>Comics
<option>Urlaubsfotos
</select>
<p>Die Datei auswählen</p>
<input type=file name='neuedatei' size=50><br><br>
<input type="submit"><input type="Reset" value"Zurücksetzen">
</form>
</body></html>
```

Listing 12.1: Das Script für das Schreiben der hochgeladenen Datei in eine Datenbank und für das Formular

Entgegen der bisherigen Praxis in diesem Buch erklären wir dieses Script nicht von oben nach unten, sondern in zwei Blöcken; zum einen erläutern wir das Formular, und zum anderen die Befehle zum Speichern der Datei in die Datenbank.

Anpassung am Formular

Das im vorherigen Kapitel *Dateien hochladen* verwendete Formular zum Upload der Dateien ergänzen wir um die Felder, die notwendig sind, um die zusätzlichen Informationen zu erheben, die mit der Datei gespeichert werden sollen. Im Beispiel sind dies Felder für die Kategorie, den Text für das Hyperlink und die Kurzbeschreibung.

```
<h2>Eine neue Datei hochladen</h2>

<form action='<? echo $PHP_SELF; ?>' method='post' enctype='multipart/form-
data'>
<input type="hidden" name="sent" value="1">
<p>Der Text für den Hyperlink</p>
<input type=text name='linktext' size=50 value='<? echo $linktext; ?>'><br>
<p>Kurzbeschreibung</p>
<input type=text name='beschreibung' size=50 value='<? echo $beschreibung;
?>'><br>
<p>Kategorie</p>
<select name='kategorie' size=3>
<option>Spanien
```

```
<option>Logos
<option>Comics
<option>Urlaubsfotos
</select>
<p>Die Datei auswählen</p>
<input type=file name='neuedatei' size=50><br><br>
<input type="submit"><input type="reset" value"Zurücksetzen">
</form>
```

Listing 12.2: Das um die Felder für die Kategorie, den Text für das Hyperlink und die Kurzbeschreibung ergänzte Scriptfragment

Erläuterung der Anpassung des Formulars

Das Formular ruft die gleiche Webseite erneut auf (`action='<? echo $PHP_SELF; ?>'`), da wir das Speichern der Informationen und der Datei ebenfalls in diese Webseite integrieren. Die entsprechende Beschreibung finden Sie im nächsten Abschnitt.

Mit `enctype='multipart/form-data'` wird angezeigt, dass unter anderem Dateien übertragen werden sollen.

```
<form action='<? echo $PHP_SELF; ?>' method='post'
enctype='multipart/form-data'>
```

Um anzuzeigen, dass das Formular abgeschickt wurde, verwenden wir wie gehabt das versteckte Feld.

Anschließend folgen die Felder zur Eingabe der zusätzlichen Informationen der Datei. Dem Attribut der Felder `value` weisen wir für den Fall, dass das Formular bei einem Fehler erneut angezeigt wird, den zuvor eingetragenen Wert des Feldes erneut zu, sodass der Surfer nicht erneut alles angeben muss.

Die Auswahl der Kategorie wird in diesem Script nicht auf den zuvor eingestellten Wert gesetzt, damit das Script leserlich bleibt. Wenn Sie diese Funktionalität implementieren möchten, setzen Sie mit Hilfe einer `if`-Anweisung das Attribut `selected` in das zuvor ausgewählte `<option>`-Tag. Eine genauere Erklärung zu dieser Vorgehensweise finden Sie in Kapitel 4 *Ein Kontaktformular*. Blättern Sie also notfalls zurück.

Die Schaltfläche zur Auswahl der Datei und das Textfeld, das den Pfad zur ausgewählten Datei anzeigt, generieren Sie mit dem folgenden `input`-Tag:

```
<input type=file name='neuedatei' size=50>
```

Die Datei und die Informationen in der Datenbank speichern

Nachdem die Datei ausgesucht und hochgeladen wurde, führen wir einige Tests durch, und anschließend bereiten wir die temporäre Datei für den Import in die Tabelle vor. Mit einem `INSERT`-SQL-Statement wird ein neuer Datensatz in der Tabelle angelegt, der die hochgeladene Datei im *BLOB*-Feld speichert. Für den Fall, dass der Upload scheitert, sorgen wir wieder für Fehlermeldungen. Sie sehen dies in dem Bild 12.3.

Bild 12.3: Der Upload ist gescheitert, da die Datei zu groß war.

Wenn es geklappt hat, erhalten Sie eine freundlichere Meldung (siehe Bild 12.4).

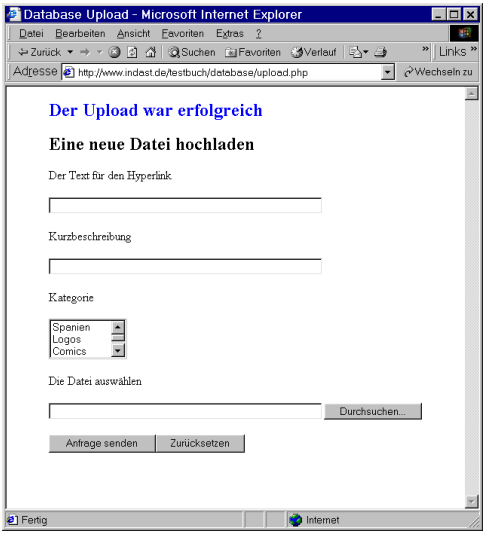

Bild 12.4: Ein erfolgreicher Upload

In diesem Script sind keine Tests der weiteren Formularfelder, wie `linktext` oder `beschreibung` enthalten. Wenn Sie derartige Tests durchführen möchten, fügen Sie entsprechende if-Anweisungen mit Fehlermeldungen ein. Lesen Sie über solche Fehlermeldungen beispielsweise *im Kapitel 4, Ein Kontaktformular*, nach. Das Listing 12.3 zeigt die Befehle zum Speichern in der Datenbank.

```
<?php

if($sent)
{

if(!is_uploaded_file($neuedatei))
{$fehler="<br>Es wurde keine Datei hochgeladen";}
else
{

if($neuedatei_size>100000)
{$fehler.="<br>Die Datei ist zu groß;";}

if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig<br>";}

}//Ende ELSE keine Datei hochgeladen

if(!$fehler)
{
include('connect.php');
$datei=fopen($neuedatei,'r');
$data=addslashes(fread( $datei, $neuedatei_size));
$sql="INSERT INTO $tabellenname (linktext, beschreibung, kategorie, dateityp,
datum, datei) values ('$linktext',  '$beschreibung', '$kategorie',
'$neuedatei_type', now(), '$data')";

if(!mysql_query($sql, $link))
{$fehler.="<br>Der Upload ist aufgrund eines Datenbankfehlers gescheitert,
bittet versuchen Sie es später noch einmal<br>";}
}

if($fehler){ echo "<h2><font color=red> $fehler </font></h2>";}
else
{
echo "<h2><font color=blue>Der Upload war erfolgreich</font></h2>";
unset($linktext);
unset($beschreibung);
unset($kategorie);
```

```
}

}//Ende $sent==1

?>
```

Listing 12.3: Speichern der Dateien in der Datenbank

Erklärung des Scripts zum Speichern der Dateien

Mit `if($sent){` testen wir, ob das Formular abgeschickt wurde, nur wenn dies der Fall ist, werden die Tests ausgeführt und wird – bei erfolgreichem Durchlaufen – der neue Datensatz angelegt.

Zunächst prüfen wir, ob überhaupt eine Datei hochgeladen wurde. Für den Fall, dass dies nicht geschehen ist, speichern wir eine Fehlermeldung in der Variablen `$fehler`; ansonsten fahren wir im else-Zweig mit den weiteren Tests fort.

Für den Test, ob eine Datei hochgeladen wurde, setzen wir den Befehl `is_uploaded_file()` ein. Der Befehl gibt `true` zurück, wenn eine Datei vorliegt. Durch die Negation mit dem Ausrufezeichen (!) wird die Fehlermeldung `$fehler` zugewiesen, wenn keine hochgeladene Datei vorhanden ist.

```
if(!is_uploaded_file($neuedatei))
{$fehler="<br>Es wurde keine Datei hochgeladen";}
else
{
```

Hinweis

```
bool is uploaded_file(string dateiname)
```

Die Funktion überprüft, ob eine Datei (dateiname) per `Post` hochgeladen wurde.

Im else-Zweig prüfen wir die Dateigröße und den Dateityp. Wenn einer der Tests fehlschlägt, wird an `$fehler` eine entsprechende Fehlermeldung angehängt.

```
if($neuedatei_size>100000)
{$fehler.="<br>Die Datei ist zu groß;";}

if(!($neuedatei_type == 'image/pjpeg' OR $neuedatei_type == 'image/gif'))
{$fehler.="<br>Der Dateityp ist nicht zulässig<br>";}
```

Passen Sie die Dateitypen entsprechend Ihren Wünschen an, verwenden Sie die Mime-Typen Ihres Servers und beachten Sie den Sicherheitshinweis (den Sie im vorhergehenden Kapitel bei Bedarf nachlesen können).

In diesem else-Zweig können Sie weitere Test nach dem bisherigen Muster für die weiteren Felder des Formulars integrieren. Wir haben in diesem Beispiel auf diese Tests verzichtet,

um das Script überschaubar zu halten. Weitere Erklärungen finden Sie wie erwähnt in *Kapitel 4, Ein Kontaktformular*.

Anschließend wird der else-Zweig beendet:

```
}//Ende ELSE keine Datei hochgeladen
```

Wenn kein Fehler vorliegt, ist `$fehler` leer, sodass das Speichern des neuen Datensatzes beginnen kann.

```
if(!$fehler)
{
```

Da die Verbindung zur Datenbank und die Zuweisung des Tabellenamens in die Variable `$tabellenname` für die verschiedenen Scripte des Datei-Uploads und die Bearbeitung der Dateien häufiger benötigt wird, haben wir für die entsprechenden Angaben eine separate Datei erstellt, die dann einfach eingebunden wird. Die Beschreibung dieser Datei finden Sie im Anschluss an diesen Abschnitt.

Hinweis

`include(string dateiname)`

Die Funktion liest die angegebene Datei ein und wertet sie aus.

Hinweis

`require(string dateiname)`

Die Funktion liest die angegebene Datei (dateiname) ein und wertet sie aus. Die Datei wird nur einmal eingelesen.

Wir binden diese Datei mit dem Befehl `include()` ein (und nicht mit `require()`), da die Verbindung und damit die Datei nur benötigt wird, wenn keine Fehler vorliegen. Über den Unterschied zwischen `include` und `reqire` können Sie sich in dem unten stehenden Hinweis informieren.

```
include('connect.php');
```

Hinweis

Der Unterschied zwischen `require()` und `include()` ist im Prinzip folgender: Wird eine Datei mit `include()` z.B. in einer if-Anweisung eingebunden, wird die Datei nur dann von PHP beachtet und gelesen, wenn die Datei auch benötigt – also die entsprechende Bedingung der if-Anweisung erfüllt ist. Hingegen werden mit `require()` eingebundene Dateien immer eingelesen, auch wenn die entsprechende if-Anweisung nicht erfüllt ist. In beiden Fällen wird der Scripttext der eingebundenen Dateien aber nur dann ausgeführt, wenn die Bedingung der if-Anweisung dies erfordert.

Verwenden Sie `include()` in einer Schleife, wird der Inhalt der Datei bei jedem Schleifendurchlauf erneut eingelesen.

Beachten Sie, dass die Ausführung von `require()` schneller ist als `include()`. Dieser Geschwindigkeitsvorteil wird aber aufgehoben, wenn eine sehr umfangreiche Datei in einer if-Anweisung (oder Ähnlichem) eingebunden wird, da diese Datei und damit der umfangreiche Inhalt mit `include` nur dann eingelesen wird, wenn die Bedingung erfüllt ist.

`Include()` empfiehlt sich also dann einzusetzen, wenn umfangreicher Scripttext nur selten unter besonderen Bedingungen benötigt wird oder wenn Sie bewusst in einer Schleife den Inhalt einer Datei jedes Mal einlesen möchten (z.B. wenn bei jedem Schleifendurchlauf eine andere Datei benötigt wird).

In unserem kleinen Beispiel wäre `require()` eigentlich sinnvoller, da die Datei nur drei Zeilen enthält und fast immer benötigt wird.

Die hochgeladene Datei kann nicht einfach so in die Datenbank geschrieben werden. Der Inhalt – die Bytes – der Datei müssen ausgelesen und einer Variablen zugewiesen werden. Erst in dieser Form können Sie die Daten in der Datenbank speichern.

Mit `fopen()` wird die temporäre Datei, deren Pfad in `$neuedatei` gespeichert ist, zum Lesen geöffnet, wobei mit dem Attribut `'r'` der Dateizeiger auf den Anfang der Datei gesetzt wird. `$datei` wird der Dateihandle zur temporären Datei zugewiesen.

```
$datei=fopen($neuedatei,'r');
```

Den Inhalt der temporären Datei können wir mit `fread()` in einem Rutsch bis zum Dateiende, das mittels der Dateigröße `$neuedatei_size` angegeben wird, auslesen. In einem Abwasch werden mit `addslashes()` Sonderzeichen mit einem Backslash (\) maskiert, bevor der gesamte Inhalt der temporären Datei der Variablen `$data` zugewiesen wird. Die Maskierung ist notwendig, damit die Daten in die Datenbank geschrieben werden können.

```
$data=addslashes(fread( $datei, $neuedatei_size));
```

Nachdem der Inhalt der temporären Datei derart vorbereitet in der Variablen `$data` steht, kann der SQL-String zum Anlegen des neuen Datensatzes in Angriff genommen werden.

Hinweis


```
string addlashes(string zeichenkette)
```

Die Funktion maskiert alle Vorkommen von bestimmten Zeichen mit einem Backslash.

Dazu brauchen wir `INSERT`. Die `INSERT`-Anweisung weisen wir der Variablen `$sql` zu. Beim SQL-Statement ist darauf zu achten, dass Sie die Texte, die in den Variablen aus dem

Formular gespeichert sind und in den SQL-String integriert werden, mit Anführungszeichen umgeben. Dies gilt auch für die Daten der Datei in `$data`. Beachten Sie bei den Anführungszeichen, dass Sie zwischen einfachen und doppelten Anführungszeichen unterscheiden müssen, damit der SQL-Text-String nicht beendet wird. Beginnen Sie den Textstring wie im Beispiel mit doppelten Anführungszeichen, so dürfen Sie zum Einschließen der Variablen nur einfache Anführungszeichen verwenden, und müssen den SQL-Text-String mit doppelten Anführungszeichen beenden.

Mit `now()` weisen Sie dem Feld `datum` das aktuelle Datum des Datenbank-Servers zu.

```
$sql="INSERT INTO $tabellenname (linktext, beschreibung, kategorie, dateityp,
datum, datei) values ('$linktext',  '$beschreibung', '$kategorie',
'$neuedatei_type', now(), '$data')";
```

Nachdem das SQL-Statement fertig ist, schicken Sie es mit `mysql_query()` an die Datenbank. Die Verbindung zur Datenbank muss nicht hergestellt werden, da dies bereits in der weiter oben eingeschlossenen Datei *connect.php* geschieht. `Mysql_query()` gibt im Erfolgsfall, also wenn die Datenbank den Datensatz anlegen kann, `true` zurück, bei Fehlern `false`. Diese Eigenschaft wird verwendet, um bei einem unvorhergesehenen Fehler der Datenbank eine Fehlermeldung an die Variable `$fehler` anzuhängen.

```
if(!mysql_query($sql, $link))
{$fehler.="<br>Der Upload ist aufgrund eines Datenbankfehlers gescheitert, bitte
versuchen Sie es später noch einmal<br>";}
```

Abschließend können Sie die if-Anweisung, die testet, ob alle vorherigen Tests erfolgreich bestanden wurden, beenden. Sie schreiben also einfach die schließende geschweifte Klammer.

```
}
```

Falls im vorstehenden Teil des Scripts ein oder mehrere Tests fehlgeschlagen sind, stehen die entsprechenden Fehlermeldungen in der Variablen `$fehler`, die in diesem Fall nicht leer ist. Die gesammelten Fehlermeldungen werden formatiert ausgegeben.

```
if($fehler){ echo "<h2><font color=red> $fehler </font></h2>";}
```

Wenn keine Fehlermeldung vorliegt, muss der Upload erfolgreich gewesen sein, und eine Erfolgsmeldung wird ausgegeben. Damit die Felder des Formulars nicht mehr die Informationen des erfolgreichen Uploads anzeigen, setzen wir nun die entsprechenden Variablen mit `unset()` zurück:

```
else
{
echo "<h2><font color=blue>Der Upload war erfolgreich</font></h2>";
unset($linktext);
unset($beschreibung);
unset($kategorie);
}
```

Jetzt fehlt nur noch die schließende Klammer für die if-Anweisung, die getestet hat, ob das Formular abgeschickt wurde.

```
}//Ende $sent==1
```

Hinweis

```
int unset(mixed variable)
```

Die Funktion löscht eine Variable oder ein Array.

Die hochgeladenen Dateien verschwinden einfach in der Datenbank. Zunächst können Sie die Dateien nicht betrachten. Dieses Manko wird mit der Webseite zum Auslesen der Dateien behoben. Zunächst können Sie den Erfolg des Uploads mit *phpMyAdmin* kontrollieren. Wenn Sie sich den Inhalt der Tabelle *buchdateien* anzeigen lassen, sehen Sie je hochgeladener Datei einen Datensatz wie in Bild 12.5.

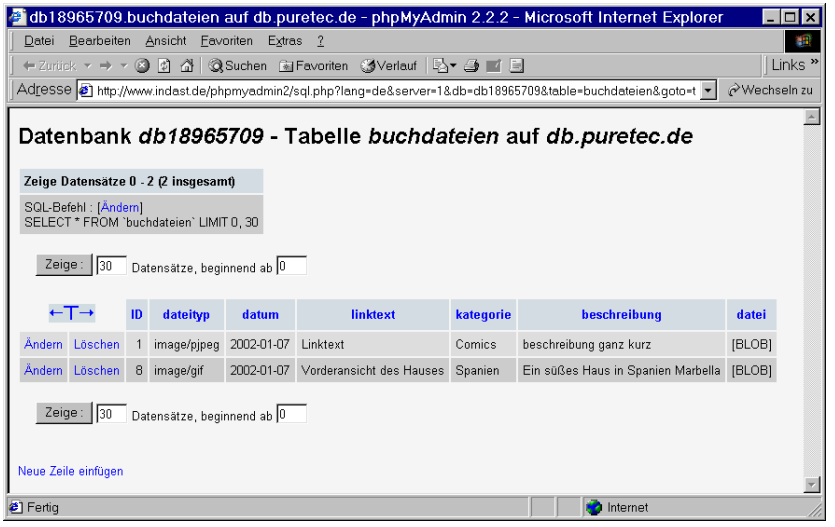

Bild 12.5: Die Einträge der hochgeladenen Dateien in phpMyAdmin

Die Verbindung zur Datenbank

Die Verbindung zur Datenbank mit der Tabelle *buchdateien* wird in allen in diesem Kapitel erstellten Scripten benötigt. Dewegen wenden wir einen kleinen »Trick« an: wir lagern den Verbindungsaufbau in eine gesonderte Datei aus, die mit `require()` oder `include()` bei Bedarf in die Scripte eingebunden wird. Der Inhalt der Datei *connect.php* ist nicht weltbewegend, es sind nur drei Zeilen (Listing 12.4):

```
<?php

$tabellenname="buchdateien";
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);

?>
```

Listing 12.4: Das kleine Script für die Verbindung zur Datenbank mit der Tabelle buchdateien

Ändern Sie diese Zeilen, um die Verbindung an Ihre Arbeitsumgebung anzupassen.

Die Datei zum Auslesen der Daten aus der Datenbanktabelle

Nachdem Sie die Dateien in der Datenbank gespeichert haben, benötigen Sie ein Script, das die hochgeladenen Dateien wieder an das Tageslicht befördert. Diesem Script müssen Sie die ID-Nummer der gewünschten Datei beim Aufruf mitteilen, damit es weiß, welche Datei es Ihnen aus den Regalen der Datenbanktabelle heraussuchen soll. Die ID-Nummer können Sie mithilfe der URL und der `Get`-Methode an das Script übergeben. Der Aufruf des Scripts, um beispielsweise die Datei des Datensatzes mit der ID-Nummer 9 zu erhalten, sieht folgendermaßen aus:

```
www.ihredomain.de/upload/anzeigen.php?ID=9
```

In dem Bild 12.6 sehen Sie den Aufruf und das Ergebnis.

Wenn Sie sich die ID-Nnummer mithilfe von *phpMyAdmin* herausfischen, können Sie dieses Script nun per Hand durch Eingabe der URL in die Adressleiste Ihres Browsers aufrufen, oder als Verweis auf eine Grafikdatei in Ihrer Webseite einbauen. Ein solcher Verweis hat die folgende Form:

```
<img src='anzeigen.php?ID=8'>
```

Sie können aber auch den Weg gehen, der im nächsten Abschnitt beschrieben wird. Hier erstellen wir eine Liste mit allen vorhandenen Dateien. Für jede Datei wird ein Link mit entsprechender ID-Nummer erstellt, der die Datei in einem neuen Browserfenster aufruft.

Auf jeden Fall benötigen Sie das in diesem Abschnitt erstellte Script, um auf die gespeicherten Dateien zuzugreifen.

Das Script zum Auslesen der hochgeladenen Dateien

Das Auslesen der betreffenden Datei ist in wenigen Zeilen programmiert. Im Gegensatz zu den meisten PHP-Scripten wird beim Auslesen von Dateien aus einer Datenbank nur in Ausnahmefällen (wenn Fehler vorliegen) Text an den Browser gesendet.

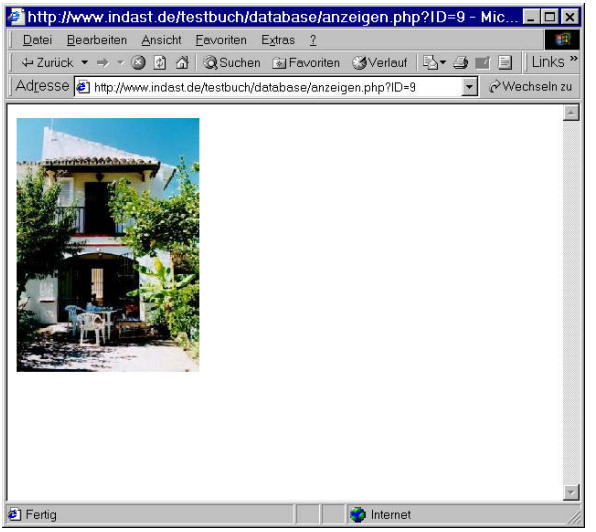

Bild 12.6: Die Datei mit der ID-Nummer 9 ist eineJPGg-Datei, die beim Aufruf direkt im Browser angezeigt wird. Achten Sie auf die Adressleiste, die ID-Nummer ist an die URL angehängt (und auf den eigentlich strahlend blauen Himmel!).

Sie dürfen nur die `HEADER`-Informationen – was es damit auf sich hat, erklären wir mit dem Script – und die Daten zum Browser schicken, selbst Leerzeichen oder leere Zeilen führen zu Fehlermeldungen. Das Listing 12.5 zeigt das komplette Script zum Auslesen und Anzeigen einer Datei aus der Datenbank.

```
<?php

if(!$ID)
{die('<h1>Keine Datei ausgesucht</h1>');}

require('connect.php');

$sqlS = "SELECT ";
$sqlS.=" datei, dateityp FROM $tabellenname";
$sqlS.=" WHERE ID = '$ID'";
$result=@mysql_query($sqlS, $link);

if(mysql_num_rows($result)==1)
{

$data = mysql_result($result,0, 0);
$typ = mysql_result($result,0, 1);
```

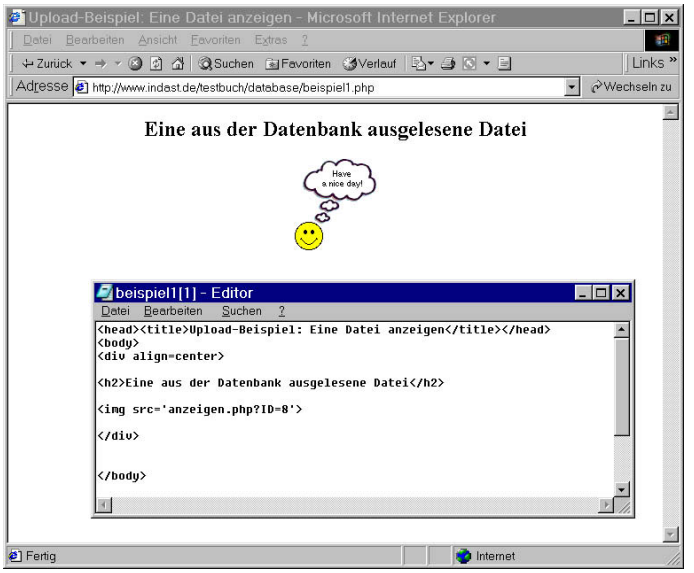

Bild 12.7: Eine einfache Webseite, die eine aus der Datenbank ausgelesene Datei anzeigt. Im Vordergrund sehen Sie den HTML-Quelltext der Webseite im Editor.

```
$kopf="Content-type: ".$typ;
header($kopf);
echo $data;

}

else
{
die('<h1>Der Download ist gescheitert</h1>');
}

?>
```

Listing 12.5: Das Script zum Auslesen und Anzeigen einer Datei aus der Datenbank

Erklärung des Scripts

Das Script kann nur eine Datei aus der Datenbank auslesen, wenn die ID-Nummer des Datensatzes bekannt ist. Aus diesem Grund prüfen wir als Erstes, ob eine ID-Nummer über die URL mit übergeben wurde. Ist dies nicht der Fall, wird die Ausführung des Scripts mit dem Befehl `die()` abgebrochen.

Hinweis

`void die(string message)`

Die Funktion beendet ein laufendes Script und sendet eine Nachricht (message) an den Browser.

Der Befehl beendet das Script und der folgende Programmcode wird nicht weiter ausgeführt, weiterhin wird der als Argument übergebene Text ausgegeben.

```
if(!$ID)
{die('<h1>Keine Datei ausgesucht</h1>');}
```

Liegt eine ID-Nummer vor, kann die Datenbank nach dem entsprechenden Datensatz befragt werden.

Mit `require()` binden wir die Datei *connect.php* ein, in der, wie oben beschrieben, die Verbindung zur Datenbank hergestellt wird.

```
require('connect.php');
```

Anschließend wird das SQL-Statement in der Variablen `$sqlS` zusammengesetzt. Hier kommt der Befehl `SELECT` ins Spiel, der angezeigt, dass Daten abgefragt werden sollen. In unserem Beispiel werden die Daten der gespeicherten Datei, die im Feld DATEI liegen, und der Dateityp, der im gleichnamigen Feld abgelegt ist, benötigt. Die Abfrage wird mit der `WHERE`-Bedingung auf den Datensatz eingeschränkt, der im Feld ID der Tabelle die in der Variablen `$ID` übergebene ID-Nummer besitzt.

```
$sqlS = "SELECT ";
$sqlS.=" datei, dateityp FROM $tabellenname";
$sqlS.=" WHERE ID = '$ID'";
```

Dann übermitteln wir die SQL-Anfrage an die Datenbank und speichern die Ergebnisliste in der Variablen `$result`. Um Fehlermeldungen von *MySQL* zu unterdrücken, verwenden wir erneut das @-Zeichen. Wir stellen es der Abfrage voran:

```
$result=@mysql_query($sqlS, $link);
```

Die Ergebnisliste darf in dem Beispiel nur einen Datensatz enthalten, da mithilfe der ID-Nummer genau ein Datensatz ausgewählt wurde. Die Anzahl der Datensätze in der Ergebnisliste werden mit `mysql_num_rows()` zurückgegeben. Wenn diese Anzahl 1 ist, hat die Abfrage genau einen Datensatz ermittelt, der anschließend für die Ausgabe vorbereitet wird.

```
if(mysql_num_rows($result)==1)
{
```

Hinweis

```
int mysql_num_rows(int Ergebniskennung)
```

Die Funktion gibt die Menge der Datensätze in der Ergebnisliste zurück.

In den vorherigen Beispielen wurde die Ergebnisliste für die Ausgabe zunächst in ein Array übergeben, um anschließend die Array-Elemente auszugeben. In diesem Beispiel greifen wir direkt auf die Ergebnisliste zu, ohne den Umweg über ein Array zu gehen. Mit dem Befehl `mysql_result()` greifen Sie direkt auf einzelne Felder der Ergebnisliste zu.

Hinweis

```
int mysql_result(int Ergebniskennung, int Datensatz-Index [, mixed
Feld])
```

Die Funktion gibt anhand der Ergebniskennung und der Angabe des Datensatz-Indexes den Inhalt eines Feldes zurück.

In die Variable `$data` schreiben wir die Daten der ehedem hochgeladenen Datei; dazu greifen wir mit `mysql_result()` auf die Ergebnisliste in `$result` zu. Mit dem nächsten Argument des Befehls wird der Datensatz bestimmt. Im Beispiel ist es der einzige vorhandene Datensatz mit der Nummer `0`, da auch die Ergebnisliste bei Null zu zählen beginnt. Das dritte Argument legt das Feld des ausgewählten Datensatzes fest. Im unserem Beispiel stehen die Daten der Datei im nullten Feld – auch die Felder der Ergebnisliste werden bei `0` beginnend durchnummeriert – , da im SQL-String das entsprechende Feld DATEI als Erstes aufgeführt wurde.

```
$data = mysql_result($result,0, 0);
```

Entsprechend weisen wir den Dateityp der Variablen `$typ` zu; hier brauchen wir allerdings das Feld mit der Nummer 1, da im SQL-String das Feld DATEITYP an zweiter Stelle steht.

```
$typ = mysql_result($result,0, 1);
```

Tipp

Sie können im Befehl `mysql_result()` die Felder nicht nur über die Reihenfolgenummer ansprechen, sondern auch über den im SQL-String verwendeten Namen, z.B. `mysql_result($result,0, 'dateityp')` anstatt `mysql_result($result,0, 1)`.

Informationen des Headers

Wichtig sind die Informationen des *Headers*. Sie dürfen nicht vergessen, dass der Server diese Informationen an den Browser sendet, bevor irgendein anderweitiger Inhalt der Seite

verschickt wird (verwechseln Sie diese `Header`-Informationen nicht mit dem `<head>`-Tag von HTML; sie werden vorher gesendet). Geben Sie – wie das nachfolgend geschieht – keine `Header`-Informationen explizit an, so werden die `Header`-Informationen automatisch vom Server generiert.

Mit dem `Header` können z. B. Informationen an Proxy-Server, die die Webseite im Internet zur Reduzierung des Datenverkehrs zwischenspeichern, oder den Browser geschickt werden, über die festgelegt wird, wie lange die Datei im *Cache* oder vom *Proxy* zwischengespeichert werden soll, sodass bei nochmaligem Aufruf die dort gespeicherte Seite angezeigt wird, bevor die neueste Version der Datei vom Server wieder angefragt wird.

Sie setzen Header-Informationen mithilfe des Befehls `header()`. Diesen Befehl müssen Sie, wie gesagt, verwenden, bevor Sie eine Ausgabe jedweder Art an den Browser senden. Hierzu zählen nicht nur gewollte Ausgaben, die Sie mit `echo()` oder `print()` anweisen, sondern auch Fehlermeldungen, die PHP ausgibt. Leider gehören auch Leerstellen oder Leerzeilen im HTML-Bereich der Webseite, die sich sonst nicht bemerkbar machen, zu den Ausgaben. Angesichts der vielen Zeilen, die ein Script mitunter umfassen kann, ist die Gefahr groß, versehentlich eine solche Leerstelle zu produzieren. Hüten Sie sich also vor diesen Missgeschicken, ansonsten werden Sie mit einer Fehlermeldung erschreckt, die so aussieht wie in dem Bild 12.8. Beachten Sie bitte den nachfolgenden Exkurs, in dem wir auf ein paar der Fehlerquellen in diesem Zusammenhang hinweisen.

Hinweis

```
int header(string zeichenkette)
```

Die Funktion sendet eine Zeile eines HTTP-Header. Headers müssen vor allen Angaben jedweder Art geschickt werden.

Im unserem Beispiel soll die Header-Information dem Browser anzeigen, welcher Typ Datei nachfolgend gesendet wird. Diese Information wird zunächst in der Variablen `$kopf` zusammengesetzt und anschließend mit dem Befehl `header()` abgeschickt.

```
$kopf="Content-type: ".$typ;
header($kopf);
```

Nachdem der Browser darüber informiert ist, welcher Typ Datei folgen wird, können die Daten der Datei mit `echo()` ausgegeben werden.

```
echo $data;
```

Das Ausgeben der ehedem hochgeladenen Datei ist damit beendet und der if-Zweig kann geschlossen werden. Für den Fall, dass nicht genau ein Datensatz in der Ergebnisliste vorliegt, sorgen wir im else-Zweig dafür, dass eine Fehlermeldung ausgegeben wird, da wir in diesem Fall leider davon ausgehen müssen, dass bei der Datenbankabfrage etwas schief gelaufen ist. Mit der Fehlermeldung wird der Programmablauf wieder mithilfe des Befehls `die()` abgebrochen.

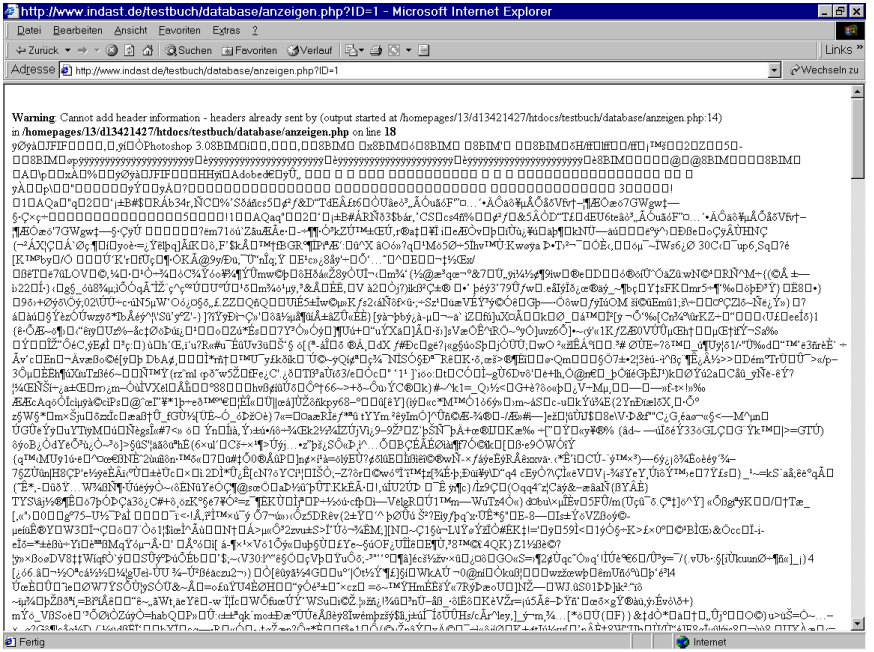

Bild 12.8: Die Fehlermeldung aufgrund einer Leerstelle im Script vor dem Header-Befehl steht am Anfang der Seite. Anschließend folgen die Daten der Grafikdatei, deren Bytes jetzt als Zeichen ausgegeben werden. Wegen der fehlenden Header-Information weiß der Browser nicht, dass es sich um eine Grafikdatei handelt.

```
  }

else
{
die('<h1>Der Download ist gescheitert</h1>');
}

?>
```

Typische Fehlerquellen bei header()

Wie oben bereits erwähnt, darf vor dem `header`-Befehl keine Ausgabe an den Browser erfolgen. Die entsprechende Fehlermeldung wurde in Bild 12.8 gezeigt.

Die Suche nach Ausgaben an den Browser ist relativ einfach, da Sie Befehle (z.B. `echo()` oder `print()`), die eine Ausgabe erzeugen, schnell finden können. Auch HTML-Bereiche vor dem Header-Bereich werden Sie meistens schnell ausfindig machen, aber mitunter liegt die Tücke im Detail. Insbesondere mit `include()` oder `require()` eingebundene Dateien führen häufig zu Fehlern, da man deren Scripttext bei der Fehlersuche nicht direkt vor Augen hat. Die folgenden Bilder zeigen Beispiele für erlaubte oder Fehlermeldungen generierende Scripttexte. Der Scripttext ist bei diesen Beispielen extrem reduziert. Die interessanten Bereiche haben wir hervorgehoben.

```
1
2  <?php
3  require('connect.php');
4
5  $sqlS = "SELECT ";
6  $sqlS.=" datei, dateityp FROM $tabellenname";
7  $sqlS.=" WHERE ID = '$ID'";
8
9  $result=@mysql_query($sqlS, $link);
10 $data = mysql_result($result,0, 0);
11 $typ = mysql_result($result,0, 1);
12 $kopf="Content-type: ".$typ;
13
14 Header($kopf);
15
16 echo $data;
17 ?>
```

Bild 12.9: Hier wird ein Fehler ausgegeben werden, da die erste Zeile, die leer ist, als HTML-Bereich angesehen wird.

```
1  <?php
2  require('connect.php');
3
4  $sqlS = "SELECT ";
5  $sqlS.=" datei, dateityp FROM $tabellenname";
6  $sqlS.=" WHERE ID = '$ID'"
7  ?>
8  <?php
9  $result=@mysql_query($sqlS, $link);
10 $data = mysql_result($result,0, 0);
11 $typ = mysql_result($result,0, 1);
12 $kopf="Content-type: ".$typ;
13
14 Header($kopf);
15
16 echo $data;
17 ?>
```

Bild 12.10: Diese Unterbrechung des PHP-Scripts ist zulässig, da keine Leerzeile zwischen den Bereichen liegt.

Bild 12.11: Wenn Sie diese Datei vor einem header-Befehl mit include() oder require() einbinden, werden Sie eine Fehlermeldung aufgrund der Leerzeilen vor dem PHP-Bereich ernten.

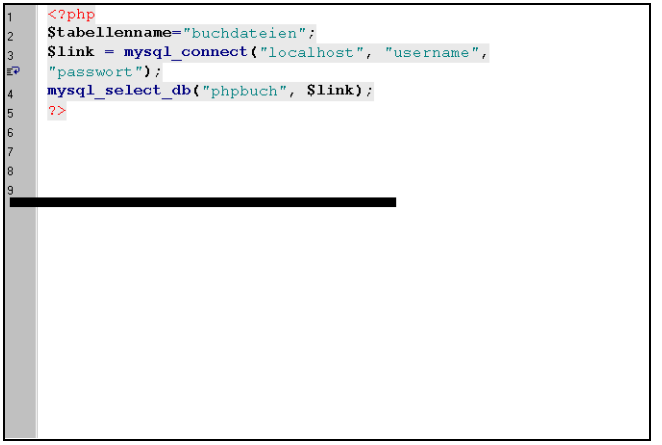

Bild 12.12: Auch wenn Sie diese Datei vor einem header-Befehl mit include() oder require() einbinden, werden Sie eine Fehlermeldung aufgrund der Leerzeilen nach dem PHP-Bereich ernten.

Eine Linkliste erstellen

Sinn und Zweck der Linkliste ist es, Hyperlinks zu allen in der Datenbank gespeicherten Dateien anzuzeigen, wobei diese Dateien mit dem Hyperlink in einem neuen Browserfenster aufgerufen werden. Um die Datei aus der Datenbank anzuzeigen, verwenden wir das oben erstellte Script (Listing 12.5) zum Auslesen der Dateien aus der Datenbank.

Die Hyperlinks zum Aufrufen der Dateien stehen in einer Tabelle zusammen mit der Beschreibung, dem Datum und der Kategorie. Dies sehen Sie in Bild 12.13.

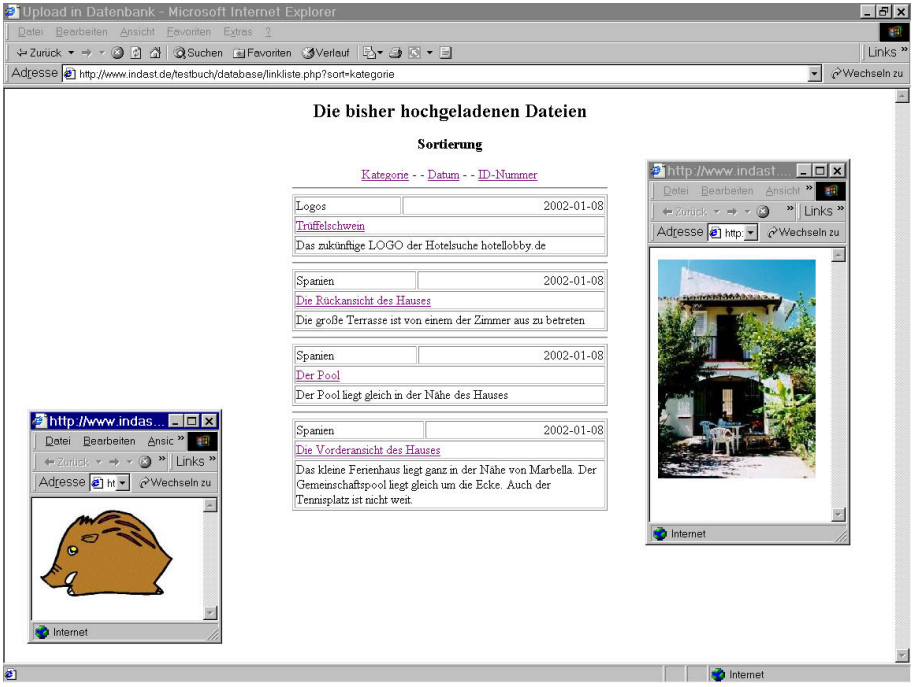

Bild 12.13: Die Liste der in der Datenbank gespeicherten Dateien. Im Vordergrund sind zwei Bilder der Datenbank aufgerufen.

Die Auflistung kann mit den Hyperlinks unterhalb der Überschrift nach unterschiedlichen Kriterien sortiert werden. Wenn keine Sortierung gewählt ist, wird die Auflistung in der Reihenfolge der Datensätze in der Datenbank ausgegeben. In dem Bild 12.14 sehen Sie eine Auflistung sortiert nach dem Upload-Datum und in dem Bild 12.15 eine Liste sortiert nach Kategorie.

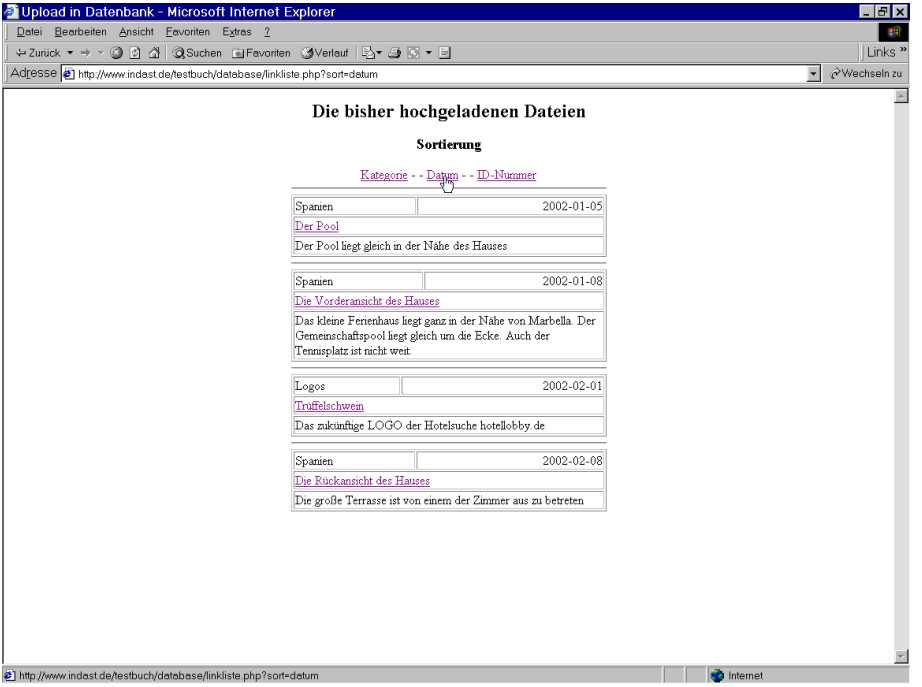

Bild 12.14: Die Auflistung sortiert nach dem Upload-Datum

Das Script für die Linkliste

In unserem Beispiel gehen wir davon aus, dass die Datei zum Auslesen der hochgeladenen Dateien *anzeigen.php* (Script entsprechend Listing 12.5) heißt und im gleichen Verzeichnis wie die in diesem Abschnitt erstellte Linkliste liegt. Die Datei *anzeigen.php* benötigt die ID-Nummer des Datensatzes, um die gewünschte Bilddatei aus der Datenbanktabelle herauszufiltern. Entsprechend werden die Hyperlinks für den Aufruf von *anzeigen.php* erstellt.

Die gewünschte Sortierung erreichen wir dadurch, dass mit den entsprechenden Hyperlinks das Feld, nach dem sortiert werden soll, an die URL, die die Auflistungsseite erneut aufruft, angehängt wird. Dieser Sortierwunsch wird beim erneuten Auslesen der Liste im SQL-Statement dann berücksichtigt.

Für die Auflistung lassen wir alle Einträge der Datenbank abfragen und anschließend aus der Ergebnisliste mithilfe einer for-Schleife ausgeben. Je Datensatz erstellen wir eine Tabelle. Das Listing 12.6 zeigt das komplette Script zum Erstellen der tabellarischen Linkliste.

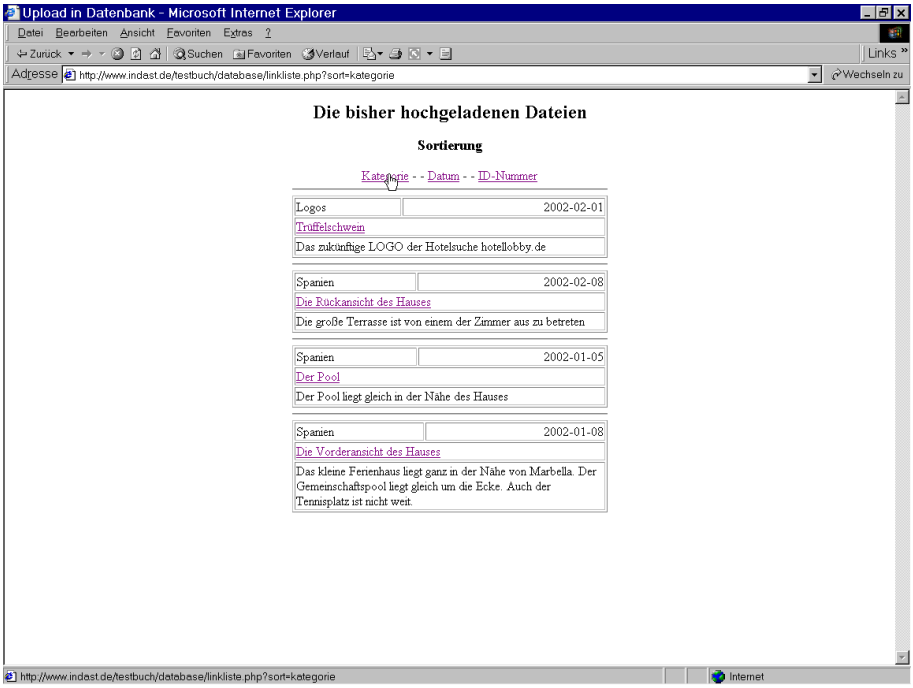

Bild 12.15: Die Auflistung sortiert nach der Kategorie

```
<html><head>
<title>Upload in Datenbank</title>
</head><body>
<div align=center>

<?php

require('connect.php');
$sql = "SELECT ";
$sql.=" ID, datum, linktext, kategorie, beschreibung FROM $tabellenname";

if($sort=="kategorie" OR $sort=="datum" OR $sort=="ID")
{

$sql.=" ORDER BY ";
$sql.=$sort;
$sql.=" DESC ";
```

```
}

$result=@mysql_query($sql, $link);

if(mysql_num_rows($result)>0)
{

echo "<h2>Die bisher hochgeladenen Dateien</h2>";
echo "<h3>Sortierung</h3>";
echo "<a href='$PHP_SELF?sort=kategorie'>Kategorie</a> - - ";
echo "<a href='$PHP_SELF?sort=datum'>Datum</a> - - ";
echo "<a href='$PHP_SELF?sort=ID'>ID-Nummer</a>";

for($i=0;$i<mysql_num_rows($result);$i++)
{

echo "<hr width=400>";
echo "<table border=1 width=400><tr><td>";
echo mysql_result($result,$i, 'kategorie');
echo "</td><td align=right>";
echo mysql_result($result,$i, 'datum');
echo "</td></tr>";
echo "<tr><td colspan=2>";
echo "<a href='anzeigen.php?ID=";
echo mysql_result($result,$i, 'ID');
echo "' target=_blank>";
echo mysql_result($result,$i, 'linktext');
echo "</a>";
echo "</td></tr>";
echo "<tr><td colspan=2>";
echo mysql_result($result,$i, 'beschreibung');
echo "</td></tr>";
echo "</table>";

}//Ende For-Schleife

}//Ende If(mysql_num_rows($result==1))

else
{

echo "<h2>Es liegen keine Einträge in der Datenbank vor</h2>";

}
```

```
?>

</div>
</body></html>
```

Listing 12.6: Das komplette Script für die Linkliste

Erklärung des Scripts

Zunächst wird der Kopf der Datei im HTML-Bereich ausgegeben. Hierbei legen wir im `div`-Tag gleich eine Zentrierung des Seiteninhalts fest.

Die Verbindung zur Datenbank haben wir – wie oben beschrieben – in der Datei *connect.php* erstellt. Diese Datei binden wir mit

```
require('connect.php');
```

nun ein.

Der SQL-String beginnt mit `SELECT`, da es darum geht, Informationen aus der Datenbank auszulesen. Anschließend werden die Felder, deren Informationen benötigt werden, durch Komma separiert gelistet. Nach `FROM` folgt der Name der Tabelle, der die Informationen entnommen werden sollen.

Beachten Sie, dass wir die Daten der hochgeladenen Dateien im Feld DATEI nicht mit auslesen lassen. Dies hat zwei Gründe:

- Die Daten werden für die Auflistung nicht benötigt.
- Wenn Sie eine Tabelle mit vielen und umfangreichen Dateien auslesen, kann die Datenmenge den Server »erschlagen«, sodass die Antwortzeit extrem lang ist und mitunter die maximale Laufzeit der Scripte überschreitet. Vermeiden Sie es aus diesem Grund in jedem Fall, Tabellen, die umfangreiche Daten enthalten, mit `SELECT * FROM` Tabellenname abzufragen (da das Sternchen alle Felder ausliest).

```
$sql = "SELECT ";
$sql.=" ID, datum, linktext, kategorie, beschreibung FROM $tabellenname";
```

Das SQL-Statement wird um die gewünschte Sortierung erweitert, wenn die Variable `$sort` gesetzt ist, wobei `$sort` durch die Links bestimmt wird, die die Seite mit dem Sortierwunsch erneut aufrufen. Diese Links erstellen wir aber erst weiter unten. Damit das SQL-Statement keinen Fehler ausgibt, testen wir zunächst, ob in `$sort` ein zur Sortierung benötigter Feldname enthalten ist. Beachten Sie, dass beim ersten Aufruf der Seite `$sort` nicht gesetzt ist, sodass die Bedingung nicht erfüllt ist und die Auflistung in der Reihenfolge der Datensätze in der Tabelle erfolgt.

```
if($sort=="kategorie" OR $sort=="datum" OR $sort=="ID")
{
```

Mit `ORDER BY` gefolgt von dem Feldnamen, nach dem sortiert werden soll, geben Sie im SQL-Statement an, dass Sie eine Sortierung der Datensätze wünschen. Dann spezifizieren Sie die Sortierrichtung, indem Sie entweder mit ASC die aufsteigende (Standard) Variante oder mit DESC die absteigende Variante angeben.

```
$sql.=" ORDER BY ";
$sql.=$sort;
$sql.=" DESC ";
}
```

Nachdem der SQL-String erstellt ist, wird er an die Datenbank geschickt und die Ergebnisliste in `$result` geschrieben. Dies macht die folgende Zeile.

```
$result=@mysql_query($sql, $link);
```

Die Überschrift, die Links für die Sortierung und die Liste sollen nur ausgegeben werden, wenn mindestens ein Datensatz vorliegt. Ob das der Fall ist, testen wir mit der Bedingung `mysql_num_rows($result)>0` in der if-Anweisung. Liegen keine Datensätze vor, weisen wir weiter unten einen entsprechender Hinweis an.

```
if(mysql_num_rows($result)>0)
{
```

Nun werden mit `echo` die Überschrift und der Hinweis auf die Sortierungslinks ausgegeben.

```
echo "<h2>Die bisher hochgeladenen Dateien</h2>";
echo "<h3>Sortierung</h3>";
```

Die Links für die Sortierung stehen in einer Zeile jeweils getrennt durch `- -`. Wichtig bei dem Verweis, den Sie jetzt einbauen, sind die folgenden Überlegungen:

- Die Seite muss mit Hilfe von `$PHP_SELF` erneut aufgerufen werden.
- An die URL wird mit ?SORT=FELDNAME das Feld angehängt, nach dem sortiert werden soll. Die Feldnamen, die Sie hier mit den Links an `$sort` übergeben, müssen auch in der obigen if-Anweisung getestet werden.

```
echo "<a href='$PHP_SELF?sort=kategorie'>Kategorie</a> - - ";
echo "<a href='$PHP_SELF?sort=datum'>Datum</a> - - ";
echo "<a href='$PHP_SELF?sort=ID'>ID-Nummer</a>";
```

Für die Auflistung entwickeln wir eine for-Schleife, die von Null bis zum letzten Datensatz in der Ergebnisliste durchlaufen wird. Je Datensatz wird dann eine Tabelle angezeigt.

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

Um die Tabellen optisch abzutrennen, weisen wir nun eine horizontale Linie an.

```
echo "<hr width=400>";
```

Dann folgt mit `<table><tr><td>` die Tabelle. In die erste Spalte der ersten Zeile schreiben wir die Kategorie. Die Kategorie ist im Feld KATEGORIE gespeichert, sodass mit `mysql_result($result,$i, 'kategorie')` der Inhalt dieses Feldes ausgegeben wird, wobei `$result` die Ergebnisliste benennt. Die Nummer des Datensatzes geben wir mit `$i` an, damit je Schleifendurchlauf der nächste Datensatz angesprochen wird. Mit dem letzten Argument geben Sie das gewünschte Feld aus der Ergebnisliste an.

```
echo "<table border=1 width=400><tr><td>";
echo mysql_result($result,$i, 'kategorie');
```

Anschließend schließen wir die Zelle mit `</td>` (achten Sie auch hier auf die Ausgabe mit `echo`) und öffnen eine neue Zelle, deren Inhalt rechtsbündig ausgerichtet sein soll. In diese Zelle wird das Upload-Datum, das im Feld DATUM abgelegt ist, ausgegeben.

```
echo "</td><td align=right>";
echo mysql_result($result,$i, 'datum');
echo "</td></tr>";
```

In die zweite Zeile der Tabelle fügen wir den Link zum Anzeigen der Datei über beide Spalten ein. Als Ziel für den Link geben wir die Datei *anzeigen.php* an. Mit Hilfe dieser Datei wird, wie oben gezeigt, eine hochgeladene Datei aus der Tabelle ausgelesen und an den Browser geschickt. Um dem Script in *anzeigen.php* mitzuteilen, welches Bild gewünscht wird, wird mit `?ID= mysql_result($result,$i, 'ID')` die ID-Nummer des Datensatzes übergeben. Der sichtbare Text wird dem Feld LINKTEXT entnommen.

```
echo "<tr><td colspan=2>";
echo "<a href='anzeigen.php?ID=";
echo mysql_result($result,$i, 'ID');
echo "' target=_blank>";
echo mysql_result($result,$i, 'linktext');
echo "</a>";
echo "</td></tr>";
```

In der letzten Zeile der Tabelle wird über die gesamte Breite der Tabelle der beschreibende Text ausgegeben.

```
echo "<tr><td colspan=2>";
echo mysql_result($result,$i, 'beschreibung');
echo "</td></tr>";
echo "</table>";
```

Nun folgen die beiden schließenden Klammern für die for-Schleife und die if-Anweisung und anschließend sorgt der `else`-Zweig dafür, dass der entsprechende Hinweis angezeigt wird für den Fall, dass keine Datensätze in der Tabelle vorhanden sind.

```
}//Ende For-Schleife
}//Ende If(mysql_num_rows($result==1))
else
{
echo "<h2>Es liegen keine Einträge in der Datenbank vor</h2>";
}
?>
```

Bearbeiten der Datensätze

Die Möglichkeit, Dateien in der Datenbanktabelle zu speichern und anschließend wieder anzeigen zu können, erfordert ein weiteres Script, mit dem man die Daten löschen und bearbeiten kann. Dieses Script schreiben wir in diesem Abschnitt. Allerdings beschränken wir die Bearbeitungsmöglichkeiten darauf, lediglich die zusätzlich zur Datei gespeicherten Informationen ändern zu können. Wir werden nicht implementieren, um eventuell auch die hochgeladenen Datei austauschen zu können.

Die zuvor erstellte Auflistung der in der Datenbank vorhandenen Datensätze bildet die Basis des nachfolgenden Scripts:

- Die Auflistung selbst wird je Datensatz um zwei Links zum Löschen und Bearbeiten ergänzt.
- Wir fügen eine if-Anweisung ein, die testet, ob ein Datensatz zuvor per Klick auf den entsprechenden Link zum Löschen freigegeben wurde. Der if-Block dieser Anweisung sorgt dann dafür, dass der entsprechende Datensatz entfernt wird.
- Eine weitere if-Anweisung prüft, ob über den entsprechenden Link ein Datensatz zum Bearbeiten ausgesucht wurde. Wenn dies der Fall ist, taucht ein Formular mit den Informationen des Datensatzes auf. In diesem Formular können die Änderungen vorgenommen werden.
- Um die im vorherigen Punkt durchgeführten Änderungen am Datensatz zu speichern, schreiben wir einen weiteren Block von Anweisungen, der nur nach dem Absenden des Formulars ausgeführt wird und die Änderungen speichert.

Diese vier Modifikationen und Anpassungen erklären wir nach der Auflistung des gesamten – inzwischen etwas umfangreicheren – Scriptes in den dann folgenden Abschnitten. Listing 12.7 zeigt das neue Script. Es integriert die Möglichkeit zum Bearbeiten und Löschen von Dateien, indem entsprechende Links gesetzt werden. Außerdem wird das Formular zum Bearbeiten eines Datensatzes erstellt.

```
<html><head>
<title>Bearbeiten der Inhalte der Datenbank</title>
</head><body>
<div align=center>

<?php
```

```
require('connect.php');

//Löschen des Datensatzes
if($sent==1)
{

$sql = "DELETE ";
$sql.=" FROM $tabellenname";
$sql.=" WHERE ID=";
$sql.=$ID;
@mysql_query($sql, $link);

}

//Speichern der neuen Werte, nachdem das Formular zum Bearbeiten abgeschickt
wurde
if($sent==2)
{

$sql = "UPDATE $tabellenname ";
$sql.=" SET ";
$sql.=" datum = '$datum', ";
$sql.=" linktext= '$linktext', ";
$sql.=" kategorie= '$kategorie', ";
$sql.=" beschreibung = '$beschreibung' ";
$sql.=" WHERE ID=";
$sql.=$ID;
@mysql_query($sql, $link);

}

//Anzeigen des Formulars zum Bearbeiten eines Datensatzes
if($sent==3)
{

$sql = "SELECT ";
$sql.=" ID, datum, linktext, kategorie, beschreibung FROM $tabellenname";
$sql.=" WHERE ID=";
$sql.=$ID;
$result=@mysql_query($sql, $link);

?>

<h2>Bearbeiten eines Datensatzes</h2>
<form action='<?php echo $PHP_SELF; ?>' method='post'>
```

```
<input type="hidden" name="sent" value="2">
<input type="hidden" name="ID" value="<?php echo mysql_result($result,0, 'ID');
?>">
<p>Der Text für den Hyperlink</p>
<input type=text name='linktext' size=50 value='<?php echo
mysql_result($result,0, 'linktext'); ?>'><br>
<p>Kurzbeschreibung</p>
<input type=text name='beschreibung' size=50 value='<?php echo
mysql_result($result,0, 'beschreibung') ?>'><br>
<p>Datum</p>
<input type=text name='datum' size=10 value='<?php echo mysql_result($result,0,
'datum') ?>'><br>
<p>Kategorie</p>
<select name='kategorie' size=3>
<option <?php if(mysql_result($result,0, 'kategorie')=='Spanien'){echo "
selected ";}?>>Spanien
<option <?php if(mysql_result($result,0, 'kategorie')=='Logos'){echo " selected
";}?>>Logos
<option <?php if(mysql_result($result,0, 'kategorie')=='Comics'){echo " selected
";}?>>Comics
<option <?php if(mysql_result($result,0, 'kategorie')=='Urlaubfotos'){echo "
selected ";}?>>Urlaubsfotos
</select><br><br>
<input type="submit"><input type="Reset" value"Zurücksetzen">
</form>

<?php

}

//Ausgeben der Liste
$sql = "SELECT ";
$sql.=" ID, datum, linktext, kategorie, beschreibung FROM $tabellenname";
if($sort=="kategorie" OR $sort=="datum" OR $sort=="ID")
{

$sql.=" ORDER BY ";
$sql.=$sort;
$sql.=" DESC ";

}

$result=@mysql_query($sql, $link);
if(mysql_num_rows($result)>0)
{
```

```
echo "<h2>Die bisher hochgeladenen Dateien</h2>";
echo "<h3>Sortierung</h3>";
echo "<a href='$PHP_SELF?sort=kategorie'>Kategorie</a> - - ";
echo "<a href='$PHP_SELF?sort=datum'>Datum</a> - - ";
echo "<a href='$PHP_SELF?sort=ID'>ID-Nummer</a>";
for($i=0;$i<mysql_num_rows($result);$i++)
{

echo "<hr width=400>";
echo "<table border=1 width=400><tr><td>";
echo mysql_result($result,$i, 'kategorie');
echo "</td><td align=right>";
echo mysql_result($result,$i, 'datum');
echo "</td></tr>";
echo "<tr><td colspan=2>";
echo "<a href='anzeigen.php?ID=";
echo mysql_result($result,$i, 'ID');
echo "' target=_blank>";
echo mysql_result($result,$i, 'linktext');
echo "</a>";
echo "</td></tr>";
echo "<tr><td colspan=2>";
echo mysql_result($result,$i, 'beschreibung');
echo "</td></tr>";

//Neue Zeile mit den Links zum Aufrufen des Bearbeitungsformulars und zum
Löschen des Datensatzes
echo "<tr><td>";
echo "<a href='$PHP_SELF?sent=1&ID=";
echo mysql_result($result,$i, 'ID');
echo "'>Löschen</a>";
echo "</td><td align=right>";
echo "<a href='$PHP_SELF?sent=3&ID=";
echo mysql_result($result,$i, 'ID');
echo "'>Bearbeiten</a>";
echo "</td></tr>";
echo "</table>";
}

}

else
{
echo "<h2>Es liegen keine Eintr&auml;ge in der Datenbank vor</h2>";
}
```

```
?>

</div></body></html>
```

Listing 12.7: Das Script – die Tabelle mit Links zum Löschen/Bearbeiten der Dateien und das Bearbeitungsformular

In dem Bild 12.16 sehen Sie die Auflistung der Dateien mit den Links zum Bearbeiten und Löschen.

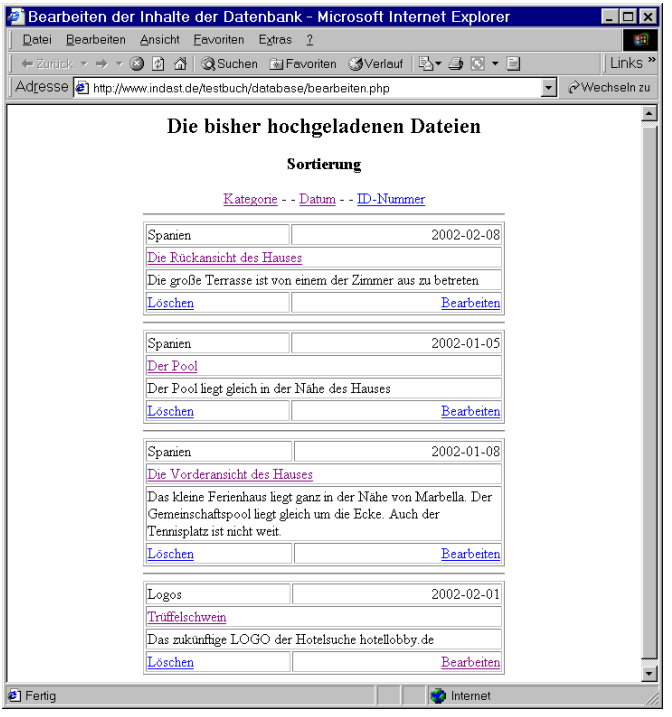

Bild 12.16: Die Links zum Bearbeiten und Löschen der einzelnen Datensätze

Hinzufügen der Links zum Löschen und Bearbeiten

Das Einfügen der Links in die im vorherigen Abschnitt erstellte Auflistung ist schnell erledigt. Fügen Sie in der Tabelle, die in der for-Schleife für jeden Datensatz erstellt wird, folgende Zeilen ein.

```
echo "<tr><td>";
echo "<a href='$PHP_SELF?sent=1&ID=";
echo mysql_result($result,$i, 'ID');
echo "'>Löschen</a>";
echo "</td><td align=right>";
echo "<a href='$PHP_SELF?sent=3&ID=";
echo mysql_result($result,$i, 'ID');
echo "'>Bearbeiten</a>";
echo "</td></tr>";
```

In der ersten Spalte rufen wir mit dem Hyperlink `Löschen` das Script erneut auf (`href='$PHP_SELF`). An die URL wird zur Kennzeichnung, dass der Datensatz gelöscht werden soll `?sent=1` angehängt. Dadurch kann beim erneuten Aufrufen mit `if($sent==1)` getestet werden, ob ein Datensatz gelöscht werden soll, sodass dann die entsprechenden Befehle ausgeführt werden. Mit `&ID="` und `echo mysql_result($result,$i, 'ID');` wird die ID-Nummer des Datensatzes an die URL angehängt, damit beim nächsten Aufruf bekannt ist, welcher Datensatz gelöscht werden soll. Der `echo`-Befehl vervollständigt den Link.

Nach dem gleichen Prinzip schreiben Sie den Link zum Bearbeiten des Datensatzes in die nächste Spalte der Tabelle. In diesem Fall wird `sent=3` an die URL angehängt. Dies signalisiert, dass das Bearbeitungsformular eingeblendet werden soll.

Löschen eines Datensatzes

Das Löschen eines Datensatzes sollte vor der Auflistung der Datensätze geschehen, aus dem einfachen Grund, dass der gelöschte Datensatz in der Auflistung nicht mehr mit angezeigt wird. Mit

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

testen Sie, ob das Script mit Hilfe des Links zum Löschen eines Datensatzes aufgerufen wurde. Nur dann ist `$sent==1`, da `sent` weiter oben mit diesem Wert an die URL des Links angehängt wurde. In der if-Anweisung erstellen wir nun das SQL-Statement zum Löschen des Datensatzes und senden es an die Datenbank. Anschließend folgt die schließende Klammer der if-Anweisung.

```
$sql = "DELETE ";
$sql.=" FROM $tabellenname";
$sql.=" WHERE ID=";
$sql.=$ID;
@mysql_query($sql, $link);

}
```

Es ist nicht notwendig, eine Verbindung zur Datenbank aufzubauen, da diese in der zuvor mit `require()` eingeschlossenen Datei *connect.php* bereits etabliert wurde.

Das Formular zum Bearbeiten eines Datensatzes

Wenn Sie das Script mit dem Link zum Bearbeiten eines Datensatzes aufrufen, wird – wie oben beschrieben – mittels der URL `$sent` auf `3` gesetzt. Aus diesem Grund wird das Formular entsprechend der if-Anweisung nur angezeigt, wenn `$sent==3` ist:

```
if($sent==3)
{
```

Anschließend werden die Informationen des gewünschten Datensatzes aus der Datenbank ausgelesen. Mithilfe der URL wurde die ID-Nummer des Datensatzes in `$ID` übergeben. Das benötigte SQL-Statement sieht somit folgendermaßen aus:

```
$sql = "SELECT ";
$sql.=" ID, datum, linktext, kategorie, beschreibung FROM $tabellenname";
$sql.=" WHERE ID=";
$sql.=$ID;
```

Dieses SQL-Statement senden wir an die Datenbank. Die Ergebnisliste, die nur einen Datensatz umfassen kann, wird in die Variable `$result` geschrieben.

```
$result=@mysql_query($sql, $link);
```

Anschließend verlassen wir den PHP-Bereich vorübergehend, da das Formular sich schneller im HTML-Modus schreiben lässt.

Nach der Überschrift folgt das einleitende `<form>`-Tag. Das Formular ruft das Script erneut auf. Für die Ausgabe von `$PHP_SELF` wird kurzfristig wieder in den PHP-Modus gewechselt (`<php ... ?>`). Dies wird auch für die weiteren Ausgaben der Ergebnisse der Datenbankabfrage getan.

```
<h2>Bearbeiten eines Datensatzes</h2>
<form action="<?php echo $PHP_SELF; ?>" method="post">
```

Mit den folgenden versteckten Feldern geben wir `sent` den Wert `2`, um anzuzeigen, dass ein Datensatz bearbeitet wurde und um diese Änderungen zu speichern. Die ID-Nummer des Datensatzes wird dem versteckten Feld ID zugewiesen.

```
<input type="hidden" name="sent" value="2">
<input type="hidden" name="ID" value="<?php echo mysql_result($result,0, 'ID');
?>">
```

Anschließend folgen die Felder mit den Informationen zu den Datensätzen. Als Vorbelegung werden jeweils über `value` die Werte des Datensatzes aus der vorherigen Abfrage verwendet.

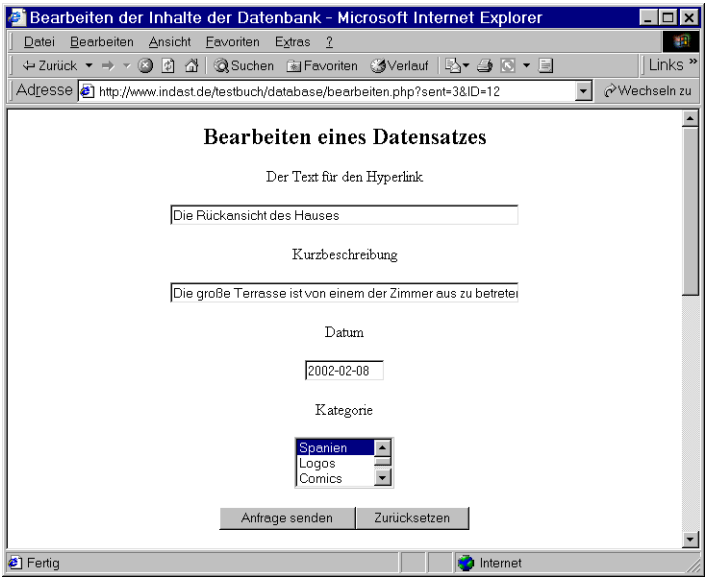

Bild 12.17: Das Formular zum Bearbeiten einzelner Datensätze

```
<p>Der Text für den Hyperlink</p>
<input type=text name='linktext' size=50 value='<?php echo
mysql_result($result,0, 'linktext'); ?>'><br>
<p>Kurzbeschreibung</p>
<input type=text name='beschreibung' size=50 value='<?php echo
mysql_result($result,0, 'beschreibung') ?>'><br>
<p>Datum</p>
<input type=text name='datum' size=10 value='<?php echo mysql_result($result,0,
'datum') ?>'><br>
```

Das Auswahlfeld der Kategorie sollte die gleichen Optionen bieten wie das Auswahlfeld beim Uploaden einer neuen Datei. Die Vorauswahl dieses Feldes ist etwas komplizierter zu realisieren, da im entsprechenden `<option>`-Tag das Attribut `selected` gesetzt werden muss. Diese Aufgabe erledigen Sie, indem Sie für jedes `<option>`-Tag eine if-Anweisung schreiben, die testet, ob der Wert aus der Datenbank mit dem Wert der Option übereinstimmt. Ist dies der Fall, wird `selected` ausgegeben.

```
<p>Kategorie</p>
<select name='kategorie' size=3>
<option <?php if(mysql_result($result,0, 'kategorie')=='Spanien'){echo "
selected ";}?>>Spanien
<option <?php if(mysql_result($result,0, 'kategorie')=='Logos'){echo " selected
```

```
";}?>>Logos
<option <?php if(mysql_result($result,0, 'kategorie')=='Comics'){echo " selected
";}?>>Comics
<option <?php if(mysql_result($result,0, 'kategorie')=='Urlaubfotos'){echo "
selected ";}?>>Urlaubsfotos
</select><br><br>
```

Es folgen die Schaltflächen zum Abschicken und Zurücksetzen des Formulars sowie das `</form>`-Endtag.

```
<input type="submit"><input type="Reset" value"Zurücksetzen">
</form>
```

Hinweis

Mit der Schaltfläche ZURÜCKSETZEN leeren Sie in diesem Fall den Feldinhalt nicht. Die mit den Attributen `value` gesetzten Werte werden wieder angezeigt.

Anschließend gehen wir zurück in den PHP-Modus, und setzen die schließende Klammer der obersten if-Anweisung.

Speichern der Änderungen an einem Datensatz

Auch den Befehl zum Speichern der Änderungen schreiben sie sinnvollerweise vor der Auflistung der Datensätze, damit die Änderungen in der Auflistung berücksichtigt werden.

Im Formular zum Bearbeiten der Datensätze haben wir im versteckten Feld `sent` auf `2` gesetzt; deswegen wird das Speichern nur ausgeführt, wenn `$sent==2` ist.

```
if($sent==2)
{
```

Zunächst bauen wir den SQL-String zusammen und senden ihn anschließend an die Datenbank. Nach `SET` werden den Feldern der Datenbank die neuen Werte zugewiesen, die vom Formular gesendet wurden. Beachten Sie, dass zwischen den Feldnamen/Wertepaaren Kommata stehen, aber hinter der letzten Zuweisung kein Komma mehr gesetzt wird. Mithilfe der im versteckten Feld übermittelten ID-Nummer wird in der `WHERE`-Bedingung der zu aktualisierende Datensatz eindeutig bestimmt.

```
$sql = "UPDATE $tabellenname ";
$sql.=" SET ";
$sql.=" datum = '$datum', ";
$sql.=" linktext= '$linktext', ";
$sql.=" kategorie= '$kategorie', ";
```

```
$sql.=" beschreibung = '$beschreibung' ";
$sql.=" WHERE ID=";
$sql.=$ID;
@mysql_query($sql, $link);

}
```

Damit haben Sie es geschafft. Sie können die Datensätze aufrufen, um sie zu editieren oder zu löschen und die Änderungen speichern. War doch lediglich eine Fingerübung, oder?

Kapitel 13
Grafiken mit PHP

Was wäre das World Wide Web ohne seine Hunderttausende von Bildern, Grafiken, Animationen etc. Auch wenn die Fülle an optischen Reizen oftmals regelrecht erschlagend ist, macht doch gerade die visuelle Vielfalt den ganz speziellen Reiz des WWW aus.

PHP ist naturgemäß kein Hauptakteur in Sachen Bilder. Aber es bietet, wie Sie in diesem Kapitel sehen, die Möglichkeit, auch in Sachen Bilder und Grafiken interessante Dynamik ins Spiel zu bringen.

Um den Umgang mit den Image-Befehlen zu üben, wird zunächst ein kleines »Kunstwerk« erstellt. Sie werden erstaunt sein, dass PHP über so viel kreative Möglichkeiten verfügt!

Anschließend erzeugen wir ein paar Grafiken, die Sie in zukünftigen Webprojekten als variable Schaltflächen einsetzen können. Das besondere daran ist, dass den Schaltflächen beim Aufruf über die URL der jeweilige Text, der auf den Schaltflächen zu lesen sein soll, mitgeteilt wird. Es geht also um echte dynamische Schaltflächen, und nicht um Schaltflächen, die lediglich ihr Aussehen wechseln können.

Beim Surfen im Netz stoßen Sie sehr häufig auf Bilder, die eine Art Vorschau darstellen und sich bei Bedarf vergrößern lassen. Wir erstellen in diesem Kapitel zu guter Letzt ein Script zum Erstellen solcher Vorschaugrafiken (thumbnail, wie die nette englische Bezeichnung dafür heißt) für bereits vorhandene Grafiken. Dieses Script können Sie nicht nur dazu nutzen, große Grafiken zu verkleinern, sondern auch um kleine Grafiken zu vergrößern. Erwarten Sie dabei aber keine Wunder, was die Qualität angeht. Dieses Script wird schließlich und endlich dazu genutzt, eine tabellarische Auflistung von Grafikdateien eines Ordners mit Vorschaugrafiken zu versehen.

In dem Bild 13.1 sehen Sie eine Webseite mit Vorschaubild, vergrößert in einem neuen Fenster, in dem Bild 13.2. sehen Sie eine Art Fotogalerie.

Aufgrund der Konventionen des HTTP-Protocols sendet der Browser die Anfrage nach einer Datei zunächst einfach an den Server und wartet dann auf eine Antwort. Hierbei ist es egal, ob es sich um eine Word-, HTML-, Grafik- oder PHP-Datei handelt. Nach der Anfrage lauscht der Browser auf die Antwort des Servers. In dieser Antwort teilt der Server dem Browser im Header mit, welchen Dateityp er senden wird. Setzen Sie nun mit PHP die Header-Information derart, dass angezeigt wird, dass eine Grafikdatei kommen wird, so ist dem Browser die Dateiendung egal und er verarbeitet jede Datei wie eine Grafik.

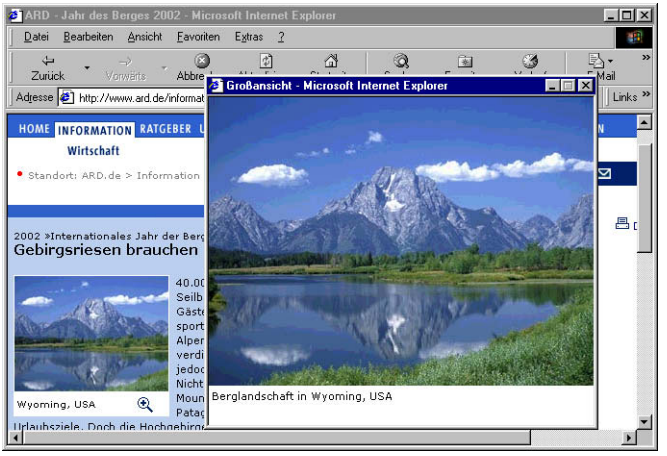

Bild 13.1: Ein Vorschaubild, vergrößert in einem neuen Fenster

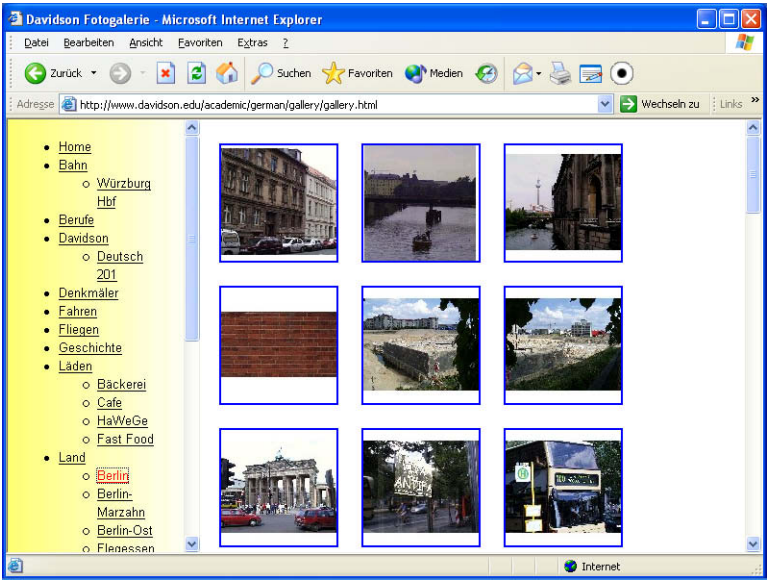

Bild 13.2: Eine Fotogalerie mit angeordneten thumbnails

Wie erstellt man mit PHP Grafiken?

In PHP stehen Ihnen eine ganze Reihe von Befehlen zur Verfügung, mit denen Sie Grafiken erstellen und bearbeiten können. Im Prinzip kann man den Umgang mit Grafiken in drei Schritte unterteilen:

1. Erstellen des Rohbildes mit `imagecreate()`. Als Argumente werden die Höhe und Breite des Bildes in Pixeln übergeben. Sie können auch mit einer fertigen Grafikdatei beginnen, die Sie dann als Basis verwenden. Je nach Grafikformat benötigen Sie hierzu einen etwas anderslautenden Befehl: `imagecreatefromwbmp()`, `imagecreatefrompng()`, `imagecreatefromjpeg()`, `imagecreatefromgif()`. Diese Befehle erwarten jeweils den Pfad zur Datei.
2. Füllen bzw. Zeichnen des Bildes mit verschiedenen Elementen. Sie können die Hintergrundfarbe bestimmen, Kreise, Rechtecke, einzelne Pixel und andere geometrische Figuren zeichnen oder auch vorhandene Grafiken einfügen. Auch Text können Sie auf die Grafik schreiben.
3. Senden des Bildes an den Browser oder Speichern des Bildes in eine Datei.

Sie haben die Wahl zwischen mehreren Grafikformaten, wobei das sehr beliebte GIF-Format bei einigen Providern (abhängig von der Version der GD-Bibliothek) aus rechtlichen Gründen nicht mehr unterstützt wird, während ältere Versionen von PHP noch nicht alle Formate unterstützen. Wie Sie herausfinden können, welche Formate Ihr Provider unterstützt, erfahren Sie im nächsten Abschnitt.

Welche Grafikformate werden unterstützt?

Ab Version 4 von PHP gibt es einen Befehl, der Ihnen die unterstützten Grafikformate ausgibt. Verwenden Sie eine ältere Version von PHP, ist es am einfachsten, die alt bewährte Try and Error- Methode zu verwenden.

Die Grafikformate mit PHP4 ermitteln

Mit dem Befehl `imagetypes()` ermitteln Sie, welche Grafikformate unterstützt werden. Der Befehl erwartet keine Parameter und gibt eine Zahl zurück. Rufen Sie den Befehl mit `echo imagetypes();` auf. Anhand dieser Zahl können Sie ersehen, welche Formate unterstützt werden. Jedem Format ist ein Wert zugeordnet. Die Werte aller unterstützten Formate werden addiert. Entnehmen Sie den nachfolgenden Tabellen die Werte und Beispielberechnungen.

Wert	Format
1	GIF
2	JPG

Tabelle 13.1: Die Werte der unterstützten Grafikformate

Wert	Format
4	PNG
8	WBMP

Tabelle 13.1: Die Werte der unterstützten Grafikformate (Forts.)

Rückgabewert	Berechnung	Formate
15	1+2+4+8	Gif JPG PNG WBMP
14	0+2+4+8	JPG PNG WBMP
11	1+2+0+8	Gif JPG WBMP
7	1+2+4+0	Gif JPG PNG

Tabelle 13.2: Beispiele für die Unterstützung von Grafikformaten

Wenn Sie keine Lust auf Kopfrechnen haben und statt dessen lieber ein paar Zeilen mehr tippen, verwenden Sie einfach das folgende kleine Script (Listing 13.1), um sich die unterstützten Formate ausgeben zu lassen. In dem Bild 13.3 sehen Sie die Ausgabe des Scriptes.

```
if(imagetypes() AND IMG_GIF){echo "Gif wird unterstützt <br>";}
if(imagetypes() AND IMG_JPG){echo "JPG wird unterstützt <br>";}
if(imagetypes() AND IMG_PNG){echo "PNG wird unterstützt <br>";}
if(imagetypes() AND IMG_WBMP){echo "WBMP wird unterstützt <br>";}
```

Listing 13.1: Mit diesem Script können Sie die unterstützten Grafikformate ermitteln.

Das Bild 13.3 zeigt die unterstützten Grafikformate, ermittelt durch `imagetypes()`.

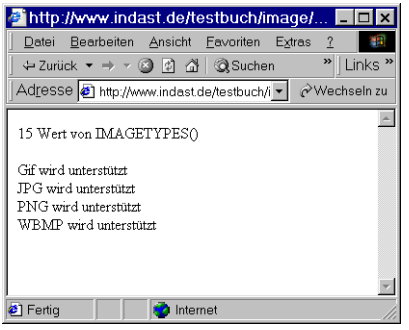

Bild 13.3: Die mit PHP4 ermittelten Grafikformate

Die verschiedenen Grafikformate austesten

Wenn Sie nicht auf PHP4 zurückgreifen können, sondern eine ältere Version verwenden, ist es am einfachsten auszutesten, welche Dateiformate für Grafiken unterstützt werden. Zu diesem Zweck erstellen Sie eine ganz einfache Grafik und geben Sie in den unterschiedlichen Formaten aus. Wenn die Grafik angezeigt wird, haben Sie ein unterstütztes Format entdeckt, ansonsten haben Sie Pech gehabt. Einige Provider lassen bei nicht unterstützten Grafikformaten freundlicherweise eine Fehlermeldung ausgeben.

Das Script zum Testen der Grafikformate ist ganz schnell geschrieben. Speichern Sie dieses Script in eine PHP-Datei und rufen Sie es mit den unterschiedlichen Header-Informationen und den darauffolgendem Befehl für die unterschiedlichen Formate auf. Im abgebildeten Script (Listing 13.2) sind alle benötigten Befehle enthalten. Sie müssen nur die auskommentierten Zeilen wechseln.

Sollten Sie eine Meldung erhalten, dass Ihr Rechner die Datei speichern möchte, erkennt Ihr Browser das Dateiformat nicht als Grafik. Dies heißt nicht unbedingt, dass die PHP-Version das Dateiformat nicht unterstützt.

```
<?php

$image = imagecreate(300,150);
$farbe_body = imagecolorallocate($image,0,255,255);

header("Content-Type: image/gif");
imagegif($image);

//header("Content-Type: image/png");
//imagepng($image);

//header("Content-Type: image/jpeg");
//imagejpeg($image,"",1);

//header("Content-Type: image/vnd.wap.wbmp");
//imagewbmp($image);

?>
```

Listing 13.2: Das Script zum Testen der unterstützten Grafikformate

Bild 13.4 zeigt eine Testgrafik im Browser.

Wenn das Grafikformat nicht unterstützt wird, erhalten Sie eine Warnmeldung, zu sehen in dem Bild 13.5.

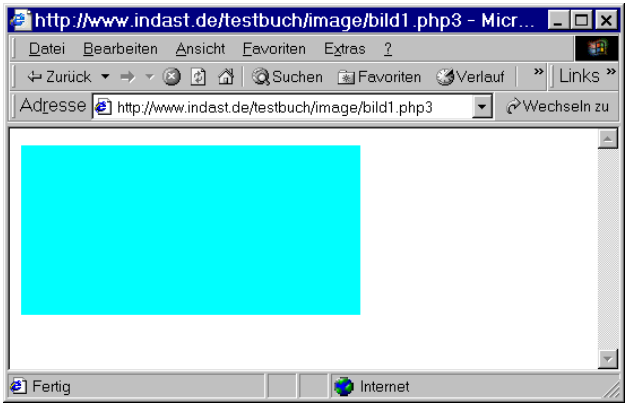

Bild 13.4: Die Testgrafik im Browser, wenn das gewählte Grafikformat unterstützt wird

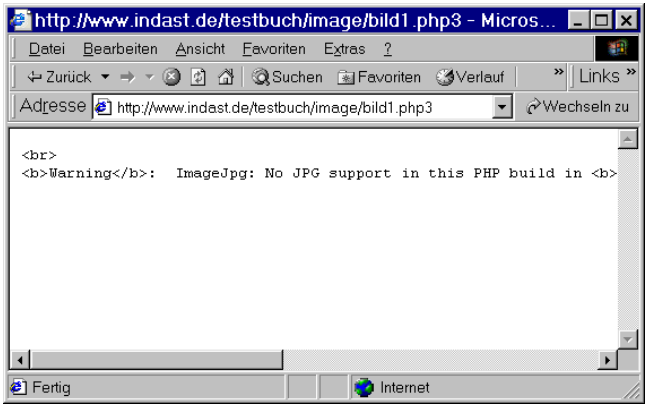

Bild 13.5: Die Warnmeldung im Browser, wenn das gewählte Grafikformat nicht unterstützt wird

Erstellen der ersten Grafik

Um den Umgang mit Grafiken zu demonstrieren, entwerfen wir ein modernes »Kunstwerk« bestehend aus zwei Linien, einem Rechteck und einem Polygon. Sie finden am Ende des gelisteten Scripts die Befehle, um das Bild in den unterschiedlichen Formaten auszugeben. Sie können das Script entsprechend Ihres Bildformats leicht anpassen, indem Sie die Kommentierung der benötigten Zeilen entfernen und umgekehrt vor dem JPG-Format, das wir hier verwenden, Kommentierungszeichen einfügen. Sie sehen das Bild, wenn Sie dieses Script direkt im Browser aufrufen.

Achtung

Das Bild wird nur im Browser angezeigt, wenn die Zeilen für den Header am Ende des Scripts und der folgende Befehl zum Senden des Bildes (z.B. `imagejpeg($image,"",100)`) im Script enthalten sind. Bedenken Sie dies, wenn Sie zwischendurch einen Kontrollblick auf das Bild werfen möchten.

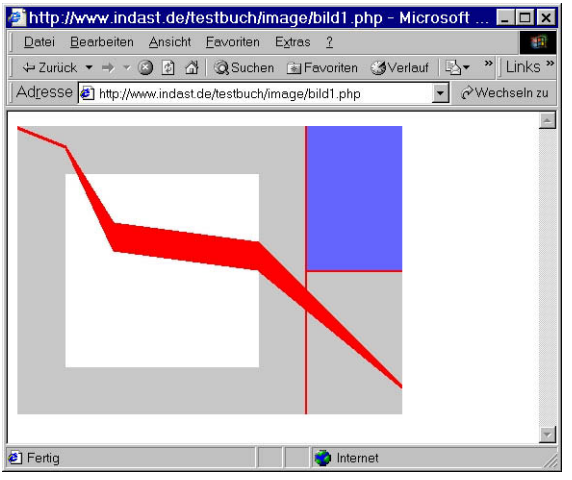

Bild 13.6: Das fertige »Kunstwerk«

Das Listing 13.3 zeigt das Script zum Erstellen der Zeichnung. Aufgrund der Größenangaben und der richtigen Platzierung ist es nicht »ganz ohne«!

```
<?php

$image = imagecreate(400,300);

$farbe_body = imagecolorallocate($image,200,200,200);
$Schwarz = imagecolorallocate($image,0,0,0);
$weis = imagecolorallocate($image,255,255,255);
$red = imagecolorallocate($image,255,0,0);
$blau = imagecolorallocate($image,100,100,255);

imagefill($image, 0, 0, $farbe_body);
imageline($image, 300, 0, 300, 300, $red);
imageline($image, 300, 150, 400, 150, $red);
imagefill($image, 301, 0, $blau);
imagefilledrectangle($image, 50, 50, 250, 250, $weis);
```

```
//1.Punkt
$punkte[] = 0;
$punkte[] = 0;
//2.Punkt
$punkte[] = 50;
$punkte[] = 20;
//3.Punkt
$punkte[] = 100;
$punkte[] = 100;
//4.Punkt
$punkte[] = 250;
$punkte[] = 120;
//5.Punkt
$punkte[] = 400;
$punkte[] = 270;
//6.Punkt
$punkte[] = 400;
$punkte[] = 273;
//7.Punkt
$punkte[] = 250;
$punkte[] = 150;
//8.Punkt
$punkte[] = 100;
$punkte[] = 130;
//9.Punkt
$punkte[] = 50;
$punkte[] = 23;
//10.Punkt
$punkte[]= 0;
$punkte[]= 3;
imagefilledpolygon ($image, $punkte, count($punkte)/2, $red);

//header("Content-Type: image/gif");
//imagegif($image);

//header("Content-Type: image/png");
//imagepng($image);

header("Content-Type: image/jpeg");
imagejpeg($image,"",100);

imagedestroy($image);

?>
```

Listing 13.3: Ein Script zum Erstellen einer Grafik

Erläuterung des Scripts

Zunächst weisen Sie PHP an, eine Rohgrafik zu erstellen, die Sie anschließend bearbeiten und mit Elementen füllen können. Beachten Sie: Diese Rohgrafik existiert nur im Arbeitsspeicher des Rechners. Sie müssen zum Anzeigen oder Speichern der Grafik diese Aktionen explizit angeben.

Als Argumente erwartet der Befehl `imagecreate()` die Breite und Höhe der Grafik in Pixeln. Der Befehl gibt den Handle auf die Grafik zurück. Diesen Handle müssen Sie in den Befehlen zum Bearbeiten und Zeichnen der Grafik immer mit angeben, damit PHP weiß, welche Grafik bearbeitet werden soll.

```
$image = imagecreate(400,300);
```

Hinweis

```
int imagecreate(int x_size, int_y-size)
```

Die Funktion erzeugt ein Rohbild oder eine Art Arbeitsfläche. Die Größe wird als Breite (x-size) und Höhe (y_size) in Pixeln angegeben.

Bevor Sie ein bisschen zeichnen können, müssen Sie bestimmen, welche Farben Sie verwenden möchten. Sie müssen nicht alle Farben in einem Block definieren, sondern können die Definition auch kurz vor dem Einsatz der Farbe vornehmen. Im Moment ist es so, dass PHP die zuerst definierte Farbe automatisch für den Hintergrund der Grafik verwendet. Im Beispiel gehen wir aber auf Nummer Sicher und füllen die Grafik sicherheitshalber nochmals mit der Farbe – wer weiß, ob zukünftige Versionen dieses Verfahren beibehalten.

Die Definition der Farben erfolgt mit dem Befehl `imagecolorallocate()`. Dieser Befehl gibt einen Handle auf die Farbe zurück, den Sie einer Variablen zuweisen, um die Farbe beim Zeichnen einzusetzen. Der Befehl erwartet folgende Argumente:

- Den mit `imagecreate()` erzeugten Handle auf die Grafik, für die die Farbe gesetzt werden soll.
- Die Angabe der Farbe. Der Farbwert wird über die RGB-Farbangabe in den drei Argumenten gesetzt.

```
$farbe_body = imagecolorallocate($image,200,200,200);
$schwarz = imagecolorallocate($image,0,0,0);
$weis = imagecolorallocate($image,255,255,255);
$red = imagecolorallocate($image,255,0,0);
$blau = imagecolorallocate($image,100,100,255);
```

Hinweis

Die RGB-Farbangabe bestimmt die Farbe über den Farbanteil von R=Rot, G=Grün und B=Blau. Der Farbanteil wird jeweils über Zahlen von 0 bis 255 angegeben, wobei 0 für »kein Anteil« steht und 255 für »maximaler 'Farbanteil«. Deswegen ist 0,0,0 = Schwarz – Abwesenheit aller Farben – und 255,255,255 = Weiß – volle Anwesenheit aller Farben. Grün ist 0,255,0 usw.

Wie oben bereits erwähnt, füllen wir den Hintergrund des Bildes aus Sicherheitsgründen mit der gewünschten Farbe. Der Befehl `imagefill()` arbeitet übrigens ähnlich wie die Flutfüllungen in Grafikprogrammen. Er füllt eine Fläche um einen angegebenen Punkt mit der angegebenen Farbe. Die Füllung wird soweit ausgedehnt, bis diese durch andersfarbige Pixel vollständig begrenzt wird. Da bisher noch keine Elemente gezeichnet sind, die eine Fläche abgrenzen können, füllt der Befehl den gesamten Hintergrund.

Hinweis

```
int imagefill(int image, int koordinateX, int koordinateY, int farbe)
```

Die Funktion füllt eine Arbeitsfläche mit eine Farbe. Die Füllung beginnt ab dem Koordinatenpunkt.

Der Befehl erwartet als Argumente: das Handle auf die Grafik, den Startpunkt der Füllung, angegeben durch zwei Werte – X- und Y-Koordinate – und den Handle auf die zu verwendende Farbe.

Achtung

Die Koordinatenangaben sind bei PHP-Grafiken etwas gewöhnungsbedürftig, da – anders als man das aus der Schule bei einem Koordinatenkreuz kennt –, der Nullpunkt (0,0) in der linken oberen Ecke der Grafik angeordnet ist. Die X-Werte werden Pixel für Pixel horizontal nach rechts hochgezählt und die Y-Werte vertikal nach unten.

```
imagefill($image, 0, 0, $farbe_body);
```

Jetzt wird das erste einfache Element auf die Rohgrafik gezaubert: Eine vertikale Linie von 300,0 nach 300,300, wobei der erste Wert der Tupel (Wertepärchen) die X-Koordinate und der zweite die Y-Koordinate angeben. Als weitere Argumente benötigt der Befehl `imageline()` das Handle auf die Grafik und auf die Farbe.

```
imageline($image, 300, 0, 300, 300, $red);
```

Der Trend geht zur »Zweitlinie«; also zeichnen wir schnell noch eine weitere Linie von 300,150 nach 400,150.

```
imageline($image, 300, 150, 400, 150, $red);
```

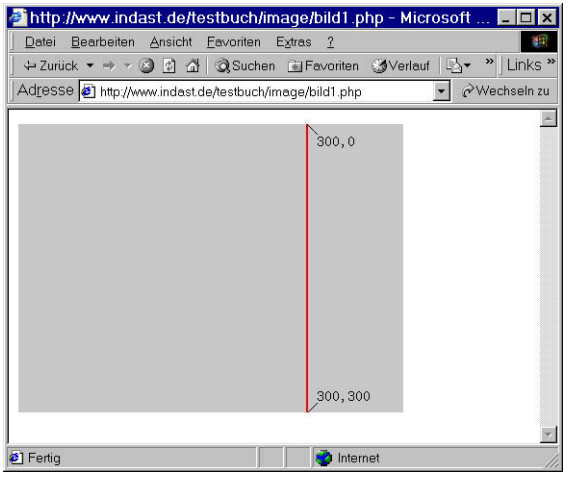

Bild 13.7: Eine vertikale Linie auf der Rohgrafik

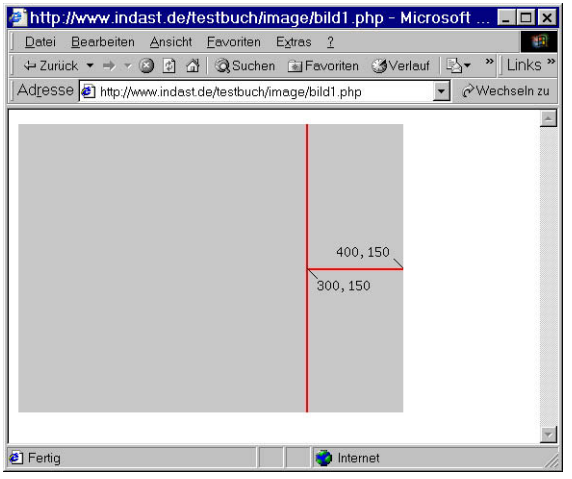

Bild 13.8: Die horizontale Linie

Hinweis

```
int imageline(int image, int koordinateX1, int koordinateY1, int
koordinateX2, int koordinateY2, int farbe)
```

Die Funktion zeichnet eine Linie zwischen die angegebenen Koordinaten.

Mit dem Befehl `imagefill()` soll nun der rechte obere Bereich der Grafik, der durch die Linien entstanden ist, gefärbt werden. Um diesen Bereich mit dem Befehl anzusprechen, brauchen wir irgendeinen Punkt in diesem Bereich, z.B. 301,0. Weitere Argumente sind, wie oben bereits erwähnt, das Handle auf die Grafik und die Farbe.

```
imagefill($image, 301, 0, $blau);
```

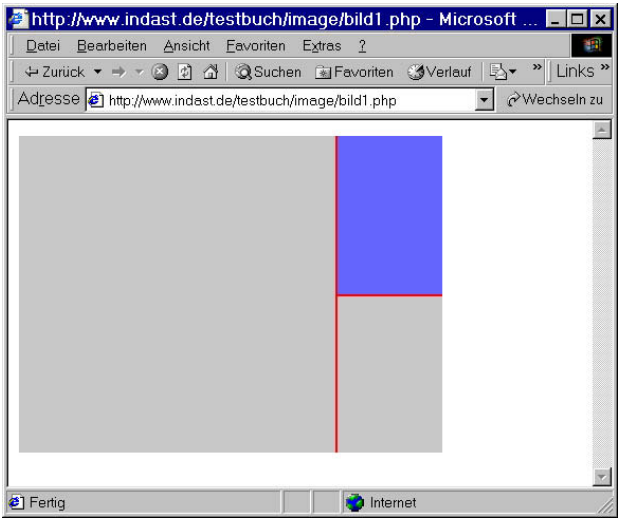

Bild 13.9: Der obere rechte Bereich ist gefüllt worden.

Jetzt erstellen wir ein gefülltes Rechteck. Mit dem Befehl `imagefillrectangle()` wird das Rechteck über die Koordinaten des linken oberen (50,50) und des rechten unteren (250,250) Punkts bestimmt. Wie gehabt benötigt auch dieser Befehl das Handle auf die Grafik und die Farbe.

```
imagefilledrectangle($image, 50, 50, 250, 250, $weis);
```

Hinweis

```
int imagerectangle((int image, int eckpunktX1, int eckpunktY1, int
eckpunkt X2, int eckpunktY2, int farbe)
```

Die Funktion erstellt ein Rechteck mit den angegebenen Eckpunkten. EckpunktX1/eckpunktY1 sind die linke obere Ecke.

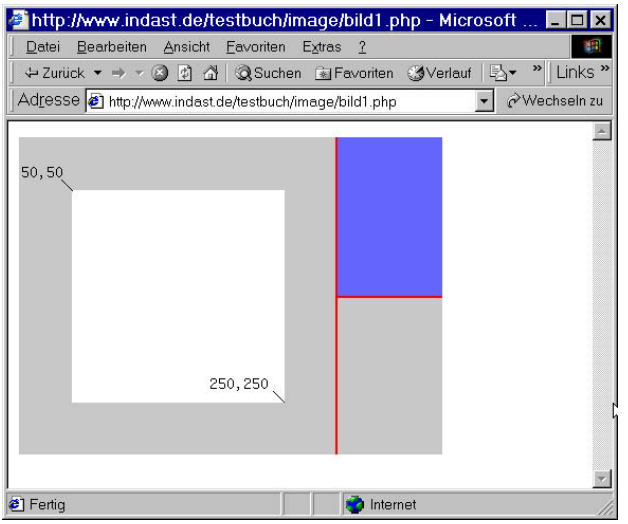

Bild 13.10: Ein weißes Rechteck wird hinzugefügt.

Das komisch geformte, zackige Etwas auf der Grafik haben wir mit dem Befehl `imagefilledpolygon()` erstellt. Mit diesem Befehl zeichnen Sie beliebig gefüllte Vielecke auf das Bild. Neben den üblichen Argumenten, das Handle auf die Grafik und die Farbe, erwartet dieser Befehl als Besonderheit ein Array mit den Koordinaten und die Anzahl der Punkte.

Hinweis

```
int imagefilledpolygon( int image, array Punkte, int anzahlPunkte, int farbe)
```

Die Funktion zeichnet ein Vieleck, wobei die Eckpunkte in einem Array (Punkte) abgelegt werden. In anzahlPunkte legt man die Anzahl der Eckpunkte fest.

Die X- und Y-Werte der Koordinaten je Punkt werden getrennt jeweils in aufeinander folgenden Arrayelementen gespeichert. Entsprechend berechnet sich die Anzahl der Punkte, indem man die Anzahl der Arrayelemente durch zwei teilt, da je Punkt zwei Elemente benötigt werden.

```
//1.Punkt
$punkte[]=0;
$punkte[]=0;
//2.Punkt
$punkte[]=50;
$punkte[]=20;
```

```
//3.Punkt
$punkte[]=100;
$punkte[]=100;
//4.Punkt
$punkte[]=250;
$punkte[]=120;
//5.Punkt
$punkte[]=400;
$punkte[]=270;
//6.Punkt
$punkte[]=400;
$punkte[]=273;
//7.Punkt
$punkte[]=250;
$punkte[]=150;
//8.Punkt
$punkte[]=100;
$punkte[]=130;
//9.Punkt
$punkte[]=50;
$punkte[]=23;
//10.Punkt
$punkte[]=0;
$punkte[]=3;
imagefilledpolygon($image, $punkte, count($punkte)/2, $red);
```

Soweit ist das Bild im Speicher fertig gestellt. Jetzt müssen/können Sie das Bild als Datei speichern oder an den Browser ausgeben. Wenn Sie das Bild an den Browser schicken, müssen Sie zunächst die Header-Information setzen.

Hinweis

```
int imagegif(int image[, string filename])
```

Die Funktion sendet ein Bild (das zuvor erstellt wurde) an den Browser, und zwar im GIF-Format. Mit dem optionalen Parameter wird das Bild in eine Datei geschrieben.

Beachten Sie, dass bei der Ausgabe an den Browser nur Bilddaten gesendet werden dürfen. Sie dürfen keine Textausgaben oder Leerstellen oder Leerzeilen im Script ausgeben, dies würde den Browser überfordern.

Tipp

Sie können das Bild auch mehrmals ausgeben, solange Sie es nicht im Speicher gelöscht haben. Allerdings können Sie nur ein Bild an den Browser senden; alternativ speichern Sie es z.B. zunächst als Datei im GIF- und anschließend im PNG- Format, senden es dann im JPG-Format an den Browser und speichern es abschließend noch im JPG-Format. Sie haben dann drei Dateien erstellt. Um ein Bild aus dem Speicher in eine Datei zu schreiben, verwenden Sie den gleichen Befehl wie zum Senden an den Browser. Geben Sie als zweiten Parameter den Pfad und den Dateinamen an, beispielsweise imagegif($image , »testbild.gif«).

Wenn Sie das Bild speichern möchten, geben Sie als weiteres Argument den Dateinamen in dem Befehl zum Erstellen des Bildes an (`imagegif($image, 'testbild.gif'))`. In diesem Fall können Sie auf die Header-Information verzichten. Eine besondere Stellung nimmt hier übrigens das JPG-Format ein. Es erlaubt drei Argumente; mit dem dritten optionalen Argument geben Sie die Qualität des Bildes mit Hilfe eines Wertes von `0` bis `100` an (`0` schlecht bis `100` gut). Möchten Sie diesen Wert bei der Ausgabe an den Browser definieren, müssen Sie für das zweite Argument – den Dateinamen – eine leere Zeichenkette – also »« – setzen.

```
//header("Content-Type: image/gif");
//imagegif($image);
//header("Content-Type: image/png");
//imagepng($image);

header("Content-Type: image/jpeg");
imagejpeg($image,"",100);
```

Wenn das Bild im Arbeitsspeicher nicht mehr benötigt wird, können Sie den Speicher freigeben mit:

```
imagedestroy($image);
```

Achtung

Wenn Sie Koordinaten angeben, die außerhalb des erzeugten Bildes liegen, ist PHP sehr sensibel. Es wird nicht einmal eine Fehlermeldung ausgegeben. Sie erhalten nur einen allgemeinen HTTP-Fehler.

Hinweis

```
int imagedestroy(int image)
```

Die Funktion löscht den Speicher, der durch das Bild besetzt wurde.

Erstellen einer einfachen Schaltfläche für wechselnde Texte

Mit dem folgenden Script möchten wir demonstrieren, wie bequem und schnell Sie variierenden Text auf ein und dieselbe Schaltfläche schreiben können. Die hier vorgestellte Methode erspart Ihnen die Mühe, für Ihr Webseitenprojekt viele Schaltflächen mit jeweils fester Beschriftung zu erstellen. Das Script, das wir hier schreiben, gibt eine Grafikdatei zurück, die dann den beim Aufruf über die URL jeweils übergebenen Text anzeigt.

Die Schaltfläche selbst besteht aus einem einfachen Rechteck mit schwarzem Rand, wobei sie den Vorteil bietet, dass die Breite und Höhe der Schaltfläche je nach Text und Schriftart angepasst werden können. Die Schriftart und die Randbreiten können mit Hilfe der URL übergeben werden, geschieht dies nicht, verwendet das Script die festgelegten Standardwerte.

Der Aufruf der Schaltfläche erfolgt über einen normalen `<Image>`-Tag, wobei die für die Erstellung der Grafik benötigten Informationen mit Hilfe der URL übergeben werden. Sie schreiben in den `<image>`-Tag der Webseite, für die Sie die Schaltfläche verwenden möchten, dann also – generell ausgedrückt – `<image src= "schalt.php?text=schaltflaechentext">` sofern das Script für die Schaltfläche *schalt.php* heißt und die Variable für den auszugebenden Text `$text`.

Denken Sie daran, dass der Text, der auf der Grafik abgebildet werden soll, vor der Übergabe mithilfe der Funktion `urlencode()` für die Übermittlung per URL vorbereitet werden muss, damit er keine Zeichen enthält, die stören. Ein Aufruf der Schaltfläche mit Hyperlink und Schriftattributen könnte in einem HTML-Bereich z.B. auf die folgende Art und Weise geschehen:

```
<a href='zieladresse'><img border=0 src='schalt1.php?text=<?php echo
urlencode('Der Text der Schaltflächen'); ?>&font=4'></a>
```

Integrieren Sie die Schaltfläche auf diese Art, so wird sie den angehängten Text (hier: »Der Text auf der Schaltfläche«) anzeigen und dabei die Fontart Nummer 4 verwenden.

Beachten Sie das Attribut `border=0`. Die meisten Browser legen automatisch einen Rahmen um eine Grafik, wenn sie als Hyperlink eingesetzt wird. Mit `border=0` wehren Sie sich gegen dieses Verfahren.

In dem Bild 13.11 sehen Sie eine Reihe von mit PHP erzeugten Schaltflächen.

Der Programm-Code für eine variable Schaltfläche:

```
<?php

If(!$text){$text = 'Beschriftung';}else{$text=urldecode($text);}
If(!$randh){$randh = 3;}
If(!$randv){$randv = 6;}
If(!$font){$font = 5;}
```

```
$hochtext = imagefontheight($font);
$breittext = strlen($text) * imagefontwidth($font);
$hoch=$hochtext+$randh*2;
$breit=$breittext+$randv*2;

$image = imagecreate($breit,$hoch);
$farbe_body=imagecolorallocate($image,210,210,210);
$schwarz=imagecolorallocate($image,0,0,0);
$blau=imagecolorallocate($image,100,100,255);
imagefill ($image, 0, 0, $farbe_body);
imagerectangle($image, 0, 0, $breit-1, $hoch-1, $schwarz);
imagestring($image, $font, $randv-1, $randh-1, $text, $blau);

header("Content-Type: image/gif");
imagegif($image);
imagedestroy($image);

?>
```

Listing 13.4: Das Script zum Erstellen einer Grafik. Am Ende wird die Grafik an den Browser geschickt und dann »zerstört«.

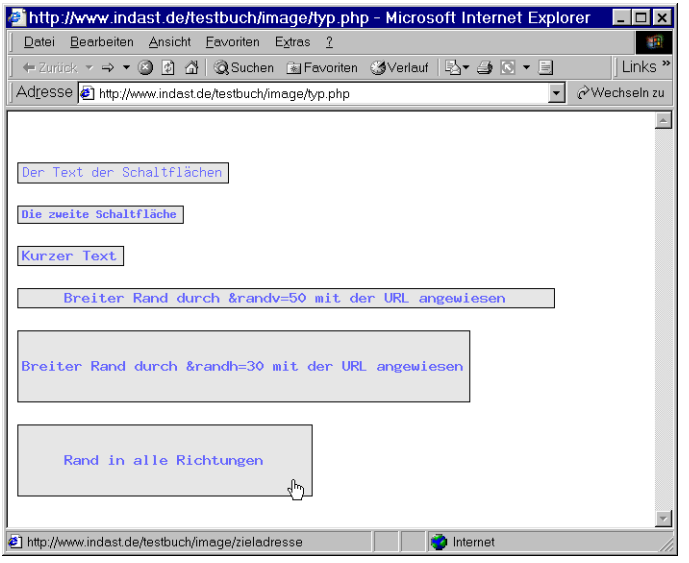

Bild 13.11: Verschiedene Schaltflächen – alle Änderungen sind über Anpassungen an der URL erreicht.

Erläuterung des Scripts zum Erstellen einer Grafik

Zunächst setzen Sie die benötigten Variablen auf Standardwerte, falls keine oder auch unzulässige Werte per URL übergeben wurden. Im Einzelnen steuern die Variablen die folgenden Eigenschaften der Grafik:

- `$text`: Der Text, der auf der Schaltfläche angezeigt werden soll.
- `$randh`: Der horizontale Rand oben und unten um den Text. Dieser Wert muss größer als 1 sein, damit das ein Pixel breite Rechteck noch um den Text gelegt werden kann. Wird ein Wert übergeben, wird er, falls keine ganze Zahl vorliegt, aufgerundet.
- `$randv`: Der vertikale Rand links und rechts vom Text. Auch dieser Wert muss größer 1 sein und wird gegebenenfalls aufgerundet.
- `$font`: Die Art der zu verwendenden Schriftart. PHP liefert fünf interne Schriftarten mit, die mit den Werten 1-5 angesprochen werden. Wird keiner dieser Werte übergeben, verwendet PHP einfach die Standardschriftart.

```
If(!$text){$text = 'Beschriftung';}else{$text=urldecode($text);}
If(!$randh OR $randh<1){$randh = 3;} else {$randh = ceil($randh);}
If(!$randv OR $randv<1){$randv = 6;} else {$randv = ceil($randv);}
If($font!=1 AND $font!=2 AND $font!=3 AND $font!=4 AND $font!=5){$font = 5;}
```

Die Höhe der Schriftarten variiert. Um die Höhe der verwendeten Schriftart zu ermitteln, damit daraus die gesamte Höhe der Grafik berechnet werden kann, steht der Befehl `imagefontheight()` parat. Er gibt die Höhe der Schriftart in Pixeln zurück.

```
$hochtext = imagefontheight($font);
```

Um die Breite des Textes zu berechnen – daraus wird die Gesamtbreite der Grafik ermittelt – multiplizieren wir die Anzahl der Buchstaben, die der Befehl `strlen($text)` zurückgibt, mit der Breite eines einzelnen Buchstabens. Die Breite der Buchstaben in Pixeln gibt der Befehl `imagefontwidth()` zurück.

```
$breittext = strlen($text) * imagefontwidth($font);
```

Hinweis

```
int imagefontheight(int font) / int imagefontwidth(int font)
```

Die Funktionen ermitteln, welche Höhe bzw. Breite der Text in der angegebenen Schriftgröße benötigt. Die Werte werden in Pixeln zurückgegeben.

Die gesamte Höhe und Breite der Grafik berechnet sich also aus der Höhe und Breite des Textes. Wir addieren noch jeweils den Rand hinzu. Der Rand wird mit zwei multipliziert, da er auf beiden Seiten (links und rechts sowie oben und unten) hinzugefügt werden soll.

```
$hoch=$hochtext+$randh*2;
$breit=$breittext+$randv*2;
```

Nachdem die benötigte Breite und Höhe für die Grafik berechnet wurden, kommt `imagecreate` zum Einsatz: die Grafik wird nun mit diesen Werten als Rohbild erstellt.

```
$image = imagecreate($breit,$hoch);
```

Dann folgt die Definition der für die Grafik benötigten Farben. Anschließend wird die Grafik mit der gewünschten Hintergrundfarbe ausgefüllt.

```
$farbe_body=imagecolorallocate($image,210,210,210);
$schwarz=imagecolorallocate($image,0,0,0);
$blau=imagecolorallocate($image,100,100,255);
imagefill ($image, 0, 0, $farbe_body);
```

Den schwarzen Rand um die Schaltfläche erzeugen wir mit einem nicht gefüllten Rechteck. Die Eckpunkte des Rechtecks sind `0,0` und `$breit-1, $hoch-1`. Sie müssen von der Höhe und Breite jeweils eins abziehen, da die Pixel der Grafik bei Null beginnend durchnummeriert werden. Haben Sie eine Grafik mit einer Breite von 300 Pixeln erstellt, sprechen Sie das erste Pixel in X-Richtung mit 0 an, entsprechend ist das letzte Pixel 299 (300-1).

```
imagerectangle($image, 0, 0, $breit-1, $hoch-1, $schwarz);
```

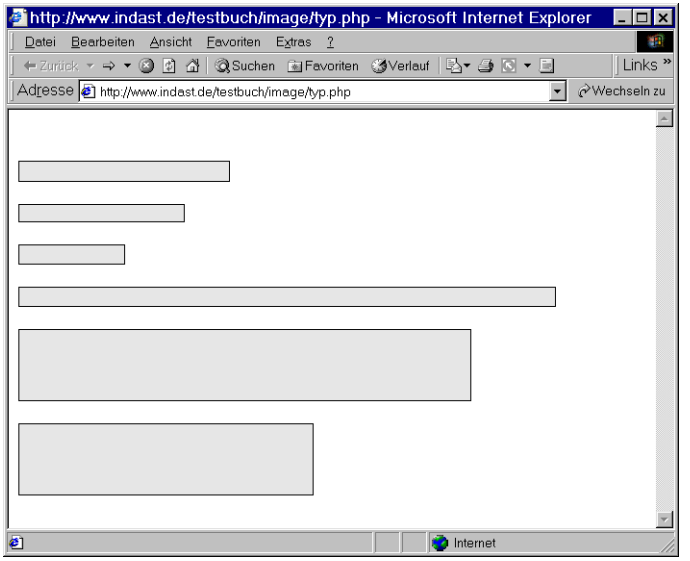

Bild 13.12: Den Schaltflächen fehlt nur noch die Beschriftung.

Die Beschriftung wird mit dem Befehl `imagestring()` auf die Grafik geschrieben. Dieser Befehl erwartet als Argumente neben dem Handle auf die Grafik und der Farbe die Schriftart (`$font`) und den Text (`$text`), der jeweils erzeugt werden soll. Die Position wird über die linke obere Ecke des Schriftbereichs festgelegt. Diese Position geben Sie mit Hilfe der X- und Y-Koordinaten an. Im Beispiel soll die Schrift entsprechend der Randangaben platziert werden. Auch hier ist eins abzuziehen, da die Pixel/Koordinaten bei `0` beginnen.

```
imagestring($image, $font, $randv-1, $randh-1, $text, $blau);
```

Hinweis

```
int imagestring(int image, int font, int koordinateX, int koordinateY,
string text, int farbe)
```

Die Funktion zeichnet eine Beschriftung (text) auf das Bild. Sie geben den Startpunkt (koordinateX, koordinateY), die Schriftgröße (font) und die Farbe an.

Abschließend wird der Header und das Bild an den Browser gesendet. Dann können Sie das Bild zerstören und den Speicher, den das Bild belegt hat, freigeben.

```
header("Content-Type: image/gif");
imagegif($image);
imagedestroy($image);
```

Vorschaugrafiken erstellen

Möchten Sie beispielsweise Ihre Urlaubsfotos in guter Qualität online zur Verfügung stellen, werden Sie keinem Surfer zumuten mögen, dass er abwartet, bis für ein paar Bilder mehrere MByte Daten an seinen Browser geschickt worden sind. Selbst mit schnellen ADSL-Verbindungen kann dabei das Surfen zur Qual werden.

In den bisher vorgestellten Varianten des Uploads von Dateien wurden die Bilder nicht alle auf einen Schlag angezeigt, sondern lediglich eine Auflistung von Links, die zu den einzelnen Bildern führt. Mit diesem Verfahren muss der Surfer nicht darauf warten, bis alle Bilder angezeigt werden, sondern er kann gezielt die Bilder aussuchen, die ihn interessieren.

Bei der datenbankgestützten Methode des Uploads erhält der Surfer über die Beschreibung einen Eindruck von dem Bild, das sich hinter dem Link versteckt, so dass er selbst beurteilen kann, ob es ihm die Übertragungszeit wert ist.

Der vorgestellte Upload, bei dem die Dateien in einem Verzeichnis gespeichert werden, vermittelt dem Surfer keinen Eindruck von den Grafiken, die sich hinter den Links verbergen. Diese Variante ergänzen wir mit dem hier entwickelten Script insofern, als dieses Script dafür sorgt, dass kleine Vorschaugrafiken in der Auswahlliste zu sehen sein werden.

Sie können auch die datenbankgestützte Variante mit Vorschaugrafiken verschönern, allerdings werden wir diese Möglichkeit hier aus Platzgründen nicht vorstellen.

Zunächst schreiben wir ein Script, das automatisch aus einer angegebenen Datei eine Vorschaugrafik erstellt. Sie sehen in Bild 13.13, dass sich dieses Script auch dazu eignet, die Vorschaugrafik größer darzustellen als das Original. Dieses Script können Sie natürlich auch unabhängig von dem Beispiel des Datei-Uploads verwenden.

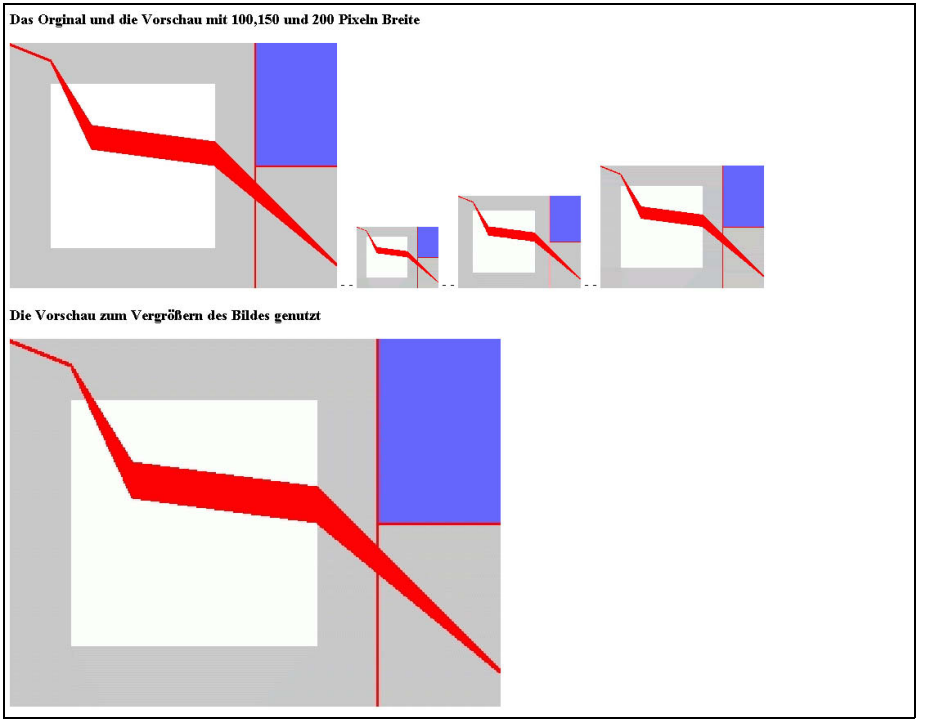

Bild 13.13: Eine Originaldatei mit vielen Vorschaugrafiken

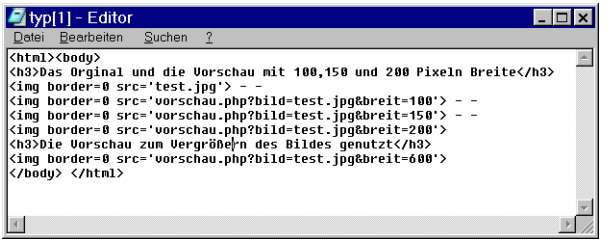

Bild 13.14: Der Quelltext der Seite, die die Vorschaugrafiken in dem Bild 13.13 anzeigt.

Anschließend wird die Upload-Seite, die im *Kapitel 11, Dateien hochladen*, erstellt wurde, derart angepasst, dass alle bisher vorliegenden Bilder mit Hilfe eines neuen Scripts in einer Tabelle mit kleinen Vorschaugrafiken angezeigt werden.

Das Script für Vorschaugrafiken

Das Script erstellt aus einer vorhandenen Grafik eine Kopie und skaliert dieses neue Bild entsprechend der gewünschten Breite, die das neue Bild erhalten soll. Aufgrund der Skalierung variieren die neuen Grafiken in der Höhe, aber immerhin werden unschöne Verzerrungen vermieden.

Um die weiteren Erklärungen zu vereinfachen, nennen wir dieses Script *vorschau.php* und speichern es in der gleichnamigen Datei.

Sie können über die URL die Breite des Bildes bestimmen. Geschieht dies nicht, wird ein im Script gesetzter Standardwert verwendet.

Auch das Originalbild können Sie über die URL angeben. Aber bedenken Sie dann bitte, dass die Pfadangabe zu dieser Datei entweder relativ von dem Script *vorschau.php* (../ORDNER/*bild.gif*) angegeben werden muss, oder eine absolute Angabe in der Form *http://www.ihredomain.de/ordner/bild.jpg* erforderlich ist. Außerdem sollten Sie nicht vergessen, dass der Dateiname bzw. Pfad URL-konform übertragen werden muss (`urlencode()`), wenn Sonderzeichen oder Leerzeichen enthalten sind. Auch hier wird ein Standardbild verwendet, wenn keine Angaben mittels der URL übertragen werden.

In unserem Beispiel senden wie die erstellte Kopie an den Browser zurück. Sie können die Kopie aber auch in einen Ordner speichern. Diese Alternative wurde bereits weiter oben beschrieben. Um sich die Vorschaugrafik einer vorhandenen Grafik anschauen zu können, rufen Sie die Datei *vorschau.php* direkt im Browser auf; dazu geben Sie in die Adressleiste die URL und die gewünschten Parameter ein, z.B.:

http://www.ihredomain.de/vorschau.php?bild=http://www.brain-and-trust.de/spain/Haus_Arcos.jpg&breit=250

oder

http://www.ihredomain.de/vorschau.php?bild=http://www.brain-and-trust.de/spain/Hausvorn.jpg&breit=150

Wie Sie dem Aufrufen des Scripts entnehmen können, erwartet das Script in `$breit` (breit) die gewünschte Breite der Kopie und in `$bild` (bild) den Pfad zur Originaldatei.

Sie können das Script, wie Bild 13.14 zeigt, auch im `<image>`-Tag einer vorhandenen Webseite aufrufen.

Listing 13.5 enthält das Script zum Erstellen von Vorschaugrafiken.

```
<?php

if(!$breit){$breit = 150;} else {$breit = ceil($breit);}
```

```
if(!$bild){$quellbild = 'test.jpg';}
else {$quellbild = urldecode($bild);}

$info = getimagesize($quellbild);
$breitalt = $info[0];
$hochalt = $info[1];

$hoch = ceil($hochalt*$breit/$breitalt);

switch($info[2])
{
case 1:
$bildalt = imagecreatefromgif($quellbild);
break;

case 2:
$bildalt = imagecreatefromjpeg($quellbild);
break;

}

$bildneu = imagecreate($breit , $hoch);
imagecopyresized($bildneu , $bildalt , 0 , 0 , 0 , 0 , $breit ,$hoch , $breitalt
, $hochalt);

header("Content-Type: image/gif");
imagegif($bildneu);
imagedestroy($bildneu);
imagedestroy($bildalt);

?>
```

Listing 13.5: Das Script zum Erstellen von Vorschau-Grafiken

Erläuterung des Scripts

Wir legen eine Standardbreite fest für den Fall, dass keine Angabe über die URL erfolgt und `$breit` dementsprechend noch nicht gesetzt ist. Gleichzeitig wird, falls `$breit` mit der URL angegeben wurde, die Breite auf die nächst höhere ganze Zahl aufgerundet. Dies macht der Befehl `ceil()`.

```
if(!$breit){$breit = 150;} else {$breit = ceil($breit);}
```

Mit der folgenden if-Anweisung testen wir, ob über die URL eine Quelldatei angegeben wurde. Ist dies nicht der Fall, sorgen wir für die Verwendung eines Standardbildes, andernfalls wird die URL-Information decodiert.

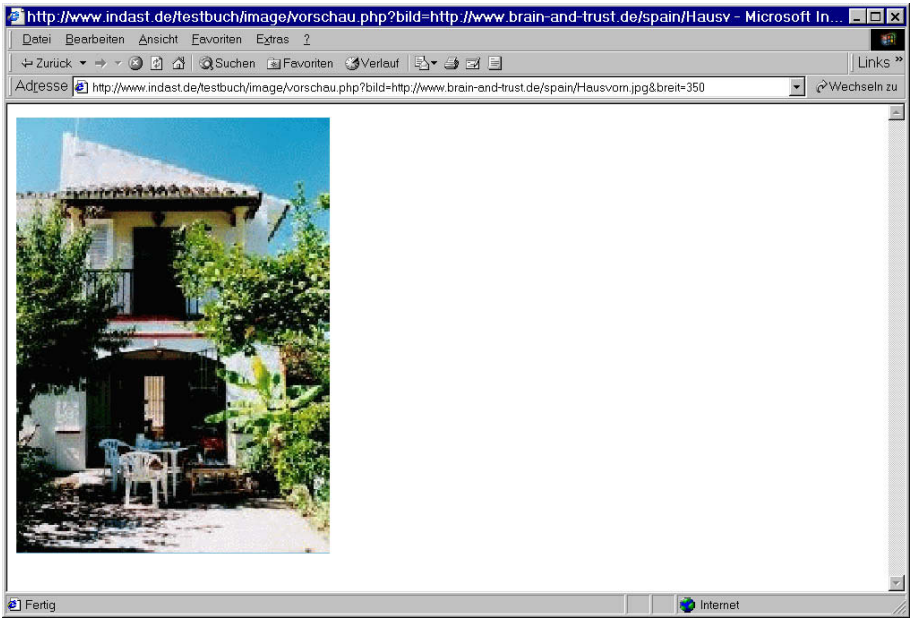

Bild 13.15: Das Script vorschau.php wird direkt im Browser aufgerufen.

```
if(!$bild){$quellbild = 'test.jpg';}
else {$quellbild = urldecode($bild);}
```

Achtung

Dieser Test überprüft nicht, ob diese Datei existiert oder erreichbar ist. Für lokale Dateien können Sie diesen Test schnell mit dem Befehl `file_exists()` hinzufügen. Für Dateien die über `http://...` angesprochen werden, ist dieser Test leider nicht trivial und deswegen verzichten wir hier auf diese Überprüfung, um die Flexibilität des Scripts nicht zu sehr einzuschränken.

Um Informationen über eine Bilddatei zu ermitteln, steht der Befehl `getimagesize()` parat. Er liefert die Höhe, Breite und das Dateiformat des Bildes in einem Array.

```
$info = getimagesize($quellbild);
```

Nachdem im Array `$info` die Informationen über das Originalbild enthalten sind, weisen wir die Höhe und Breite, die im nullten und ersten Element des Arrays abgelegt werden, Variablen zu.

```
$breitalt = $info[0];
$hochalt = $info[1];
```

Hinweis

```
array getimagesize(string filename [, array bildinfo])
```

Die Funktion gibt Informationen über das Bild aus.

Nachdem die Maße des alten Bildes bekannt sind, lässt sich die Höhe des neuen Bildes anhand dieser Werte und der gewünschten Breite berechnen. Das Ergebnis wird mit `ceil()` auf die nächst höhere ganze Zahl aufgerundet, da man kein Bild mit einer Höhe von 198,765 Pixeln erstellen kann.

```
$hoch = ceil($hochalt*$breit/$breitalt);
```

Soweit liegen jetzt die benötigten Informationen vor:

- der Pfad zu Quelldatei in `$quellbild`
- die Höhe der Originalgrafik in `$hochalt`
- die Breite der Originalgrafik in `$breitalt`
- die Höhe der neuen Grafik in `$hoch`
- die Breite der neuen Grafik in `$breite`

Nun lesen wir die Originaldatei wieder ein, damit sie anschließend kopiert werden kann.

Da der Befehl, die Informationen einer vorhandenen Bilddatei einzulesen, sich je nach Dateityp unterscheidet, verwenden wir eine `switch`-Anweisung. Diese `switch`-Anweisung testet `$info[2]`, da zuvor der Befehl `getimagesize()` das Dateiformat der Originalgrafik hier abgelegt hat.

Die Informationen über das Dateiformat werden über Zahlen kodiert: 1 steht für *gif* zwei für *jpg*. Dieser Kodierung entsprechend verwenden wir in den `cases` die Befehle zum Einlesen der vorhandenen Grafiken je Dateiformat. Das Handle auf die Originalgrafik steht anschließend mit `$bildalt` bereit.

```
switch($info[2])
{
case 1:
$bildalt = imagecreatefromgif($quellbild);
break;
case 2:
$bildalt = imagecreatefromjpeg($quellbild);
break;
}
```

Auf Basis der berechneten Maße wird ein neues Rohbild angelegt.

```
$bildneu = imagecreate($breit , $hoch);
```

Hinweis

```
int imagecreatefromgif(string filename)
```

Die Funktion liest die Informationen eines vorhandenen Bildes ein, um ein neues Bild zu erstellen.

Das alte Bild wird in das neue Rohbild kopiert. Der Befehl `imagecopyresized()` erwartet eine Reihe von Argumenten, die Sie in der erforderlichen Reihenfolge in der unten stehenden Auflistung finden. Es werden mehr Angaben erwartet als man zunächst vermuten würde, da es für das Beispiel reichen würde zu sagen: Kopiere das alte Bild in ein neues Bild mit den Maßen X und Y. Der Befehl ist aber mächtiger und braucht deswegen ein paar mehr Angaben, da er auch Ausschnitte einer Grafik an beliebige Stellen der Zielgrafik kopieren kann. In unserem Beispiel sind die Bereiche allerdings identisch mit den kompletten Grafiken.

- Handle auf die Zielgrafik – im Beispiel `$bildneu`
- Handle auf die Originaldatei – `$bildalt`
- X-Koordinate der linken oberen Ecke des Bereichs, in den kopiert werden soll (Zielgrafik) – `0`, da die gesamte neue Rohgrafik gefüllt werden soll und nicht nur ein Ausschnitt
- Y-Koordinate der linken oberen Ecke des Bereichs, in den kopiert werden soll (Zielgrafik) – `0`, da die gesamte neue Rohgrafik gefüllt werden soll und nicht nur ein Ausschnitt
- X-Koordinate der linken oberen Ecke des Bereichs, der kopiert werden soll (Originalgrafik) – `0`, da die gesamte Grafik kopiert werden soll und nicht nur ein Ausschnitt
- Y-Koordinate der linken oberen Ecke des Bereichs, der kopiert werden soll (Originalgrafik) – `0`, da die gesamte Grafik kopiert werden soll und nicht nur ein Ausschnitt
- Breite des Bereichs, in den kopiert werden soll, in Pixeln (Zielgrafik) – `$breit`, damit die neue Grafik voll ausgenutzt wird
- Höhe des Bereichs, in den kopiert werden soll, in Pixeln (Zielgrafik) – `$hoch`, damit die neue Grafik voll ausgenutzt wird
- Breite des Bereichs, der kopiert werden soll, in Pixeln (Originalgrafik) – `$breitalt`, damit die gesamte Breite der Originalgrafik erfasst wird
- Höhe des Bereichs, der kopiert werden soll, in Pixeln (Originalgrafik) – `$hochalt`, damit die gesamte Höhe der Originalgrafik erfasst wird

Mit diesen Argumenten versehen sieht der Befehl dann so aus:

```
imagecopyresized($bildneu , $bildalt , 0 , 0 , 0 , 0 , $breit ,$hoch, $breitalt,
$hochalt);
```

Abschließend schicken wir den `Header` und das neue Bild an den Browser und dann werden beide Bilder – alt wie neu – im Speicher gelöscht.

```
header("Content-Type: image/gif");
imagegif($bildneu);
imagedestroy($bildneu);
imagedestroy($bildalt);
```

Hinweis

```
int imagecopyresized(int zielimage, int quellimage, int zx, int zy, int
qx, in qy, int brziel, int hochziel, int brquell, int hochquell)
```

Die Funktion kopiert einen Teil eines Bildes in ein anderes Bild. zx und zy sind die Koordinaten für das Ziel, qx und qy für den Bereich, der kopiert werden soll. Dann werden die Breite und Höhe des Zielbereichs und die Breite und Höhe des Ursprungsbildes angegeben.

Eine Tabelle mit Vorschaugrafiken anzeigen

In diesem Abschnitt erstellen wir eine Tabelle mit Vorschaugrafiken, sodass eine kleine Bildgalerie dabei herauskommt. Die Vorschaugrafiken zeigen die Grafikdateien an, die gesammelt in einem Ordner liegen. Wir gehen im aktuellen Beispiel davon aus, dass dieses Script und das zuvor erstellte Script zum Anzeigen der Vorschaugrafiken (*vorschau.php*) in einem Verzeichnis liegen. In einem Unterverzeichnis mit Namen DATEIEN befinden sich die Bilddateien, die als Vorschau angezeigt werden sollen.

Dieses Script eignet sich beispielsweise dafür, Dateien, die nach der im *Kapitel 11, Dateien hochladen*, beschriebenen Methode hochgeladen wurden, mit Vorschaugrafiken zu versehen.

Die oben erstellte Datei *vorschau.php* brauchen wir, weil sie aufgerufen wird, um die Vorschau der im Unterverzeichnis liegenden Dateien anzuzeigen. In dem Script, das wir jetzt erstellen, ermitteln wir lediglich die Anzahl der Dateien des Unterverzeichnisses sowie die Dateinamen.

Kern des Scripts ist die Tabelle, in deren Zellen die Datei *vorschau.php* jeweils für eine Bilddatei des Unterverzeichnisses aufgerufen wird, also so oft, wie Grafiken im Unterverzeichnis gespeichert sind. Diese Vorschaugrafiken werden jeweils mit einem Link versehen, der die Originalgrafik in voller Größe in einem neuen Fenster des Browsers aufruft.

Als kleines »Schmankerl« nebenbei erstellen wir vier Links, mit denen der Surfer sich die Anzahl der Spalten der Tabelle und die Breite der Tabelle selbst aussuchen kann. Bild 13.16 zeigt die Vorschaugrafiken, unterhalb der Grafiken sehen Sie die vier Links.

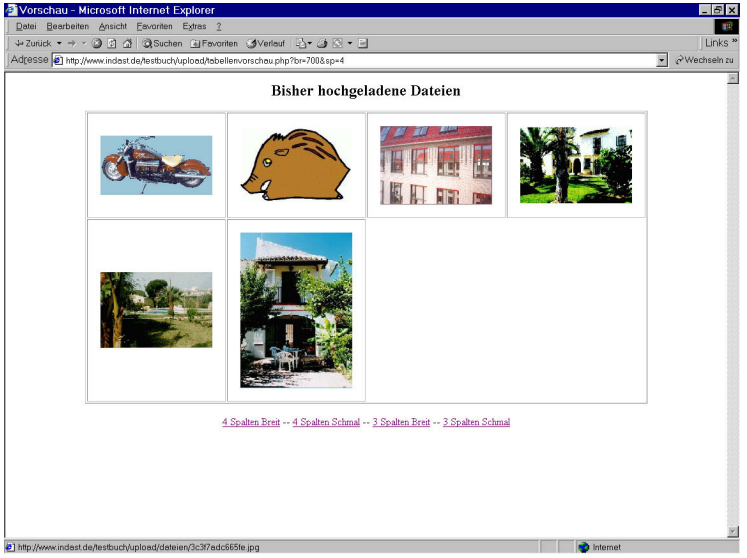

Bild 13.16: Die Tabelle mit den Vorschaugrafiken in der breiten vierspaltigen Variante

Kurze Einführung in Funktionen

In diesem Script soll erstmalig eine *Funktion* verwendet werden. Funktionen werden wir an dieser Stelle nicht in ihrem vollen Leistungsumfang erklären können. Dennoch versuchen wir in diesem Abschnitt, Sie im Schnelldurchlauf mit den Grundlagen vertraut zu machen. In der Erklärung des Scripts zum Erstellen der Vorschautabelle finden Sie dann weitere Hinweise.

Die Erfahrung hat gezeigt, dass ein gewisses Maß an Sicherheit im Umgang mit der Programmiersprache notwendig ist, bevor man die Vorteile von Funktionen richtig nutzen kann. Aus diesem Grunde haben wir es bewusst vorgezogen, die bisherigen Scripte ohne Funktionen zu erstellen, auch wenn es an einigen Stellen sinnvoll gewesen wäre und/oder sich dadurch mehr Funktionalitäten ergeben hätten.

Funktionen kann man salopp beschreiben als eine Zusammenfassung von Programmzeilen, die mit einem vergebenen Namen an jeder Stelle des Scripts aufgerufen werden können. Die in einer Funktion zusammengefassten Befehle werden nur ausgeführt, wenn Sie die Funktion über deren Namen im Script auch aufrufen. Unterlassen Sie diesen Aufruf, stehen die Programmzeilen der Funktion nur als Ballast in der Datei.

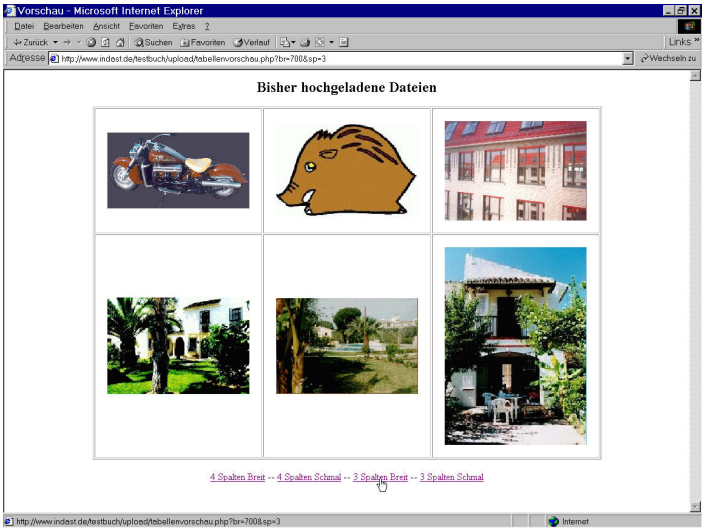

Bild 13.17: Die Tabelle mit den Vorschaugrafiken in der schmalen dreispaltigen Variante

Man kann Funktionen mit Eingabewerten füttern, die man bei jedem Aufruf neu setzen kann. Funktionen geben, sofern man sie dazu anweist, nur einen Wert zurück. Da es sich bei dem Rückgabewert aber auch um ein Array handeln kann, lassen sich mit diesem Trick mehrere Werte in einem Rutsch zurückgegeben.

Wird in der Funktion ein `echo`-Befehl eingesetzt, wird die Ausgabe an der Stelle eingesetzt, an der die Funktion aufgerufen wurde.

Hinweis

Vorab ein bisschen Theorie: Innerhalb von Funktionen können Sie nicht ohne Weiteres auf Variablen zugreifen oder diese manipulieren, die Sie im normalen Script einsetzen. Sie können also z.B. einer Variablen mit dem Namen `$eumel` im Script den Wert 7 zugewiesen haben. Wenn Sie in einer Funktion `$eumel=$eumel+10` rechnen, erhalten Sie in der Funktion den Wert 10, da auf die Variable `$eumel`, die außerhalb der Funktion gesetzt wurde, nicht zugegriffen werden kann und damit `$eumel` in der Funktion vor der Berechnung nicht gesetzt war. So wird auch der Wert der Variablen `$eumel`, die außerhalb der Funktion gesetzt wurde, durch die Berechnung innerhalb der Funktion nicht geändert, `$eumel` ist außerhalb der Funktion weiterhin 7.

Diese Eigenschaft können Sie umgehen, wenn Sie Variablen in der Funktion ausdrücklich als *global* deklarieren. Diese globalen Variablen können Sie in der Funktion dann verändern.

Eine Funktion deklarieren und aufrufen

Sie beginnen eine Funktion mit `function` gefolgt von dem Namen der Funktion und einer runden Klammer. In den runden Klammern können Sie die Namen der Variablen festlegen, die die Werte aufnehmen, die bei jedem Aufruf der Funktion mit übergeben werden sollen. Diese Variablen stehen in der Funktion dann zur Verfügung. Alle Anweisungen/Befehle der Funktion werden in geschweifte Klammern gesetzt. Die Syntax sieht dann so aus:

```
function beispiel($willi, $susie)
{
$test=$willi+$Susie;
return $test;
}
```

Sie rufen eine Funktion mit ihrem Namen auf. In den Klammern übergeben Sie die Werte an die Funktion. Die eben erstellte Funktion `beispiel` rufen Sie also mit `echo` auf und übergeben den Wert 7 und 14:

```
echo beispiel(7,14);
```

Die Funktion weist jetzt nur für den Ablauf der Funktion `$willi` den Wert `7` und `$susie` den Wert `14` zu. `7` stand beim Aufruf der Funktion an erster Stelle in der Klammer, deswegen `$willi=7` und `14` an der zweiten Stelle, deswegen `$susie=14`.

Mit diesen Werten wird die Berechnung in der Funktion durchgeführt, sodass `$test` den Wert `21` erhält. Anschließend gibt die Funktion den Wert von `$test` zurück, da dies mit dem Befehl `return` angewiesen wird (der Befehl `return` wird nur im Zusammenhang mit Funktionen eingesetzt).

Da der Wert `21` von der Funktion zurückgegeben wurde, wird im Script mit dem `echo`-Befehl auch `21` ausgegeben.

Globale Variablen

Um die Bedeutung von globalen Variablen zu verdeutlichen, wird die obige kleine Funktion `bespiel` ein wenig ergänzt und erneut aufgerufen:

```
function beispiel($willi, $susie)
{
$test=$willi+$Susie+$egon;
return $test;
}
$egon=100;
echo beispiel(7,14)
```

Auch in diesem Fall wird mit `echo` der Wert 21 ausgeben, obwohl `$egon` der Wert 100 zugewiesen wurde. Die Funktion kennt `$egon` aber nicht, sodass die Berechnung wieder 21 ergibt.

Anders sieht es aus, wenn `$egon` als global erklärt wird. Im folgenden kleinen Script ist `$egon` der Funktion deswegen bekannt und somit gibt die Funktion 121 zurück. Der `echo`-Befehl wird also 121 ausgeben.

```
function beispiel($willi, $susie)
{
Global $egon;
$test=$willi+$Susie+$egon;
return $test;
}
$egon=100;
echo beispiel(7,14)
```

Das Script für die Tabelle mit Vorschaugrafiken

Das Script sorgt dafür, dass die Grafikdateien eines Verzeichnisses in einer Tabelle angeordnet angezeigt werden. Sie können mithilfe von zwei Parametern, die Sie über die URL übergeben, die Spaltenzahl und Breite beeinflussen. Sollten diese Parameter nicht gesetzt sein, werden im Script definierte Standardwerte verwendet.

Um die Vorschaugrafiken anzuzeigen, brauchen wir die oben erstellte Datei *vorschau.php*. Der Aufruf dieser Datei übergibt per URL alle von diesem Script benötigten Informationen zum Erstellen und Anzeigen der einzelnen Vorschaugrafiken.

Beachten Sie bitte, dass die Aufrufe im gelisteten Beispiel unter der Voraussetzung erfolgen, dass das hier erstellte Script und die Datei *vorschau.php* im gleichen Verzeichnis liegen. Sollte dies nicht der Fall sein, müssen Sie die Pfadangaben natürlich entsprechend anpassen.

Der Datei *vorschau.php* werden die Namen der Grafikdateien, für die Vorschauen zu erstellen sind, mit Hilfe der URL im Aufruf übergeben. Der Pfad zu diesen Dateien ist hier ein Unterverzeichnis des aktuellen Ordners mit dem Namen DATEIEN.

Das Script verwendet ein Scriptfragment, das wir bereits im *Kapitel 11, Dateien hochladen*, geschrieben und erläutert haben. Dieses Fragment haben wir auf Grafikdateien angepasst. Die aus diesem Fragment bereits bekannten Befehle werden wir an dieser Stelle nur kurz erläutern.

Weiter oben haben wir Sie kurz mit Funktionen vertraut gemacht. Hier soll nun eine Funktion praktisch angewendet werden. Wir definieren also eine Funktion, die für die Ausgabe der einzelnen Tabellenzellen sorgt. Diese Funktion erklären wir gleich zu Beginn, also wenn sie geschrieben, und nicht erst, wenn sie aufgerufen wird.

Hinweis

Wir haben in diesem Script eine Funktion verwendet, um am Beispiel relevante Aspekte eines Einsatzes (von Funktionen) erklären zu können. Ob dies in einem ähnlichen Fall in der »echten« Praxis sinnvoll wäre, bleibt ein Stück weit Geschmacksache, ist aber auch fraglich – insbesondere, da die Funktion im Script nur einmal aufgerufen wird.

Listing 13.6 zeigt das Script, in dem die Darstellung der Vorschaugrafiken in Tabellen entwickelt wird.

```
<html><head><title>Vorschau</title></head><body>
<div align=center>

<?php

function vorschaubild($bild,$breite)
{
Global $uverz;

$ausg="<td align=center valign=center>";
$ausg.="<a href='$uverz/$bild' target=_blank><img border=0
src='vorschau.php?breit=$breite&bild=$uverz/$bild'></a>";
$ausg.="</td>";
return $ausg;
}

if(!$br){$breite=550;} else{$breite=$br;}
if(!$sp){$gesamtspalten=3;}else{$gesamtspalten=$sp;}
$uverz="dateien";
$bildbreite=$breite/$gesamtspalten;

echo "<h2>Bisher hochgeladene Dateien</h2>";

$verzeichnis=opendir($uverz);

$spalte=1;
echo "<table cellpadding=10 border=1>";

while ($file = readdir($verzeichnis))
{
$test=@getimagesize($uverz."/".$file);

if($test[2]==1 OR $test[2]==2)
{
if($spalte==1){echo "<tr>";}
```

```
echo vorschaubild($file,$bildbreite);
if($spalte<$gesamtspalten){$spalte++;}
else{$spalte=1; echo "</tr>";}
}}

closedir($verzeichnis);
echo "</table>";
echo "<br>";

echo "<a href=$PHP_SELF?br=700&sp=4>4 Spalten 700px</a> -- ";
echo "<a href=$PHP_SELF?br=400&sp=4>4 Spalten Schmal</a> -- ";
echo "<a href=$PHP_SELF?br=700&sp=3>3 Spalten Breit</a> -- ";
echo "<a href=$PHP_SELF?br=400&sp=3>3 Spalten Schmal</a>";
?>

</div></body></html>
```

Listing 13.6: Das Script zum Anzeigen der Vorschaugrafiken in Tabellen

Erklärung des Scripts zum Anzeigen der Vorschaugrafiken in tabellarischer Anordnung

Nachdem die HTML-Angaben für den Kopf der Datei geschrieben und eine Zentrierung der gesamten Ausgabe mittels eines `<div>`-Tags eingeleitet wurde, erstellen wir im PHP-Bereich die Funktion `vorschaubild`.

Diese Funktion erwartet zwei Argumente: Der Name der Grafikdatei, für die eine Vorschau erstellt werden soll, wird in der Funktion der Variablen `$bild` zugewiesen. Die Breite (in Pixeln), die die Vorschaugrafik erhalten soll, steht in der Variablen `$breite`.

```
function vorschaubild($bild,$breite)
{
```

Die Variable `$uverz` wird dem Script zugänglich gemacht – ist also *global* verfügbar. Dieser Schritt ist notwendig, da der Pfad zur Grafikdatei für den Hyperlink benötigt wird, der die Originalgrafik in einem neuen Browserfenster aufruft. Dieser Pfad steht in `$uverz`. Deswegen heißt es:

```
Global $uverz;
```

In der Variablen `$ausg` setzen wir den HTML-Text zusammen, der benötigt wird, um die Tabellenzelle samt Inhalt richtig anzuzeigen. Am Ende der Funktion wird mit dieser Variablen der gesamte HTML-Text der Zelle zurückgegeben.

Zunächst schreiben wir das öffnende `<td>`-Tag mit der Anweisung, den Inhalt vertikal als auch horizontal zu zentrieren, in die Variable.

```
$ausg="<td align=center valign=center>";
```

Anschließend wird an die Variable `$ausg` der HTML-Text für das Hyperlink zum Aufruf der Originalgrafik angehängt und mit `href='$uverz/$bild'` die gewünschte Datei angesprochen, da vereinbarungsgemäß in `$uverz` der Pfad zu den Originalgrafikdateien gespeichert und `$uverz` global gesetzt ist. In `$bild` steht – über den Funktionsaufruf festgelegt – der Dateiname. Es folgt im `<image>`-Tag der Aufruf der Datei *vorschau.php*, um die Vorschaugrafik der Originalgrafikdatei anzuzeigen. Mit diesem Aufruf wird die gewünschte Breite der Vorschaugrafik übergeben, die mit dem Funktionsaufruf in `$breite` gespeichert ist (`?breit=$breite`).

Achtung

Die in der Funktion verwendete Variable `$breite` für eine Vorschaugrafik kollidiert nicht mit der im Script gesetzten Variablen `$breite` für die Gesamtbreite, da diese Variable nicht global ist.

Die Originaldatei wird mit `&bild=$uverz/$bild` an die URL angehängt, denn in `$uverz` steht der Pfad und in `$bild` steht der Dateiname.

```
$ausg.="<a href='$uverz/$bild' target=_blank><img border=0
src='vorschau.php?breit=$breite&bild=$uverz/$bild'></a>";
$ausg.="</td>";
```

Hinweis

Liegen dieses Script und *vorschau.php* nicht in einem Verzeichnis, müssen Sie sowohl den Pfad zu *vorschau.php* im `<image>`-Tag anpassen als auch den Pfad in `&bild=$uverz/$bild`. Sie können hier nicht `$uverz` verwenden, sondern müssen den Pfad von *vorschau.php* zum Verzeichnis mit den Grafiken angeben.

Nachdem der HTML-Text in `$ausg` geschrieben ist, wird er mit `return` zurückgegeben und die Funktion mit der schließenden Klammer beendet.

```
return $ausg;
}
```

Der eigentliche Scriptteil beginnt damit, die benötigten Variablen zu ermitteln. Wird in `$br` beim Aufruf der Webseite eine gewünschte Breite in Pixeln für die Tabelle übergeben, wird dieser Wert der Variablen `$breite` zugewiesen, falls nicht, wird der Standardwert gesetzt. Diese Breite entspricht nicht exakt der Breite der später angezeigten Tabelle, da die Tabelle noch Platz um die Grafiken hinzuaddiert, der bei der Ermittlung der Breite der einzelnen Bilder nicht berücksichtigt wird.

Wenn eine Spaltenzahl in `$sp` an das Script übergeben wird, wird diese verwendet, ansonsten werden 3 Spalten erzeugt. Die Spaltenzahl steht nach der if-Anweisung in der Variablen `$gesamtspalten`.

Den Pfad zu dem Verzeichnis, das die Originalgrafikdateien beheimatet, weisen wir `$uverz` zu.

```
if(!$br){$breite=550;} else {$breite=$br;}
if(!$sp){$gesamtspalten=3;}else{$gesamtspalten=$sp;}
$uverz="dateien";
```

Anschließend wird die Breite eines Bildes berechnet und der Variablen `$bildbreite` zugewiesen.

```
$bildbreite=$breite/$gesamtspalten;
```

Dann legen wir eine Überschrift fest.

```
echo "<h2>Bisher hochgeladene Dateien</h2>";
```

Nun brauchen wir das Handle auf das Verzeichnis, das die Originalgrafikdateien enthält, damit die vorhandenen Dateinamen später ausgelesen werden können.

```
$verzeichnis=opendir($uverz);
```

Da eine variable Spaltenanzahl der Tabelle möglich ist, setzen wir eine Variable `$spalte`, um in dieser Variablen die aktuelle Spalte der Tabelle mitzählen zu können. Wir beginnen die Spalten bei 1 zu zählen.

```
$spalte=1;
```

Das einleitende `<table>`-Tag wird ausgegeben. Mit `cellpadding` sorgen wir für etwas Raum zwischen den Vorschaugrafiken und dem Tabellenrahmen.

```
echo "<table cellpadding=10 border=1>";
```

Der Befehl `readdir()` liest alle Einträge eines Verzeichnisses aus, wobei jeweils nur ein Eintrag zurückgegeben, aber bei jedem erneuten Aufruf automatisch der nächste Eintrag des Verzeichnisses zurückgegeben wird. Ist `readdir()` am Ende der Einträge angelangt, gibt der Befehl `false` zurück, sodass die while-Schleife beendet wird. Die Zuweisung des Verzeichniseintrags zu der Variablen `$file` kann deswegen gleichzeitig als Abbruchbedingung verwendet werden.

```
while ($file = readdir($verzeichnis))
{
```

Um nur Grafikdateien in die Auflistung mit aufzunehmen, für die auch Vorschaugrafiken erstellt werden können, lassen wir mit `getimagesize()` den Grafikdateityp ermitteln. Nach der Zuweisung an `$test` befindet sich die Kennziffer für den Dateityp in `$test[2]`. Erneut

setzen wir das @-Zeichen vor dem Befehl ein, das Fehlermeldungen verhindert, wenn ein Ordner oder eine Datei mit unbekanntem Dateityp angesprochen wird.

```
$test=@getimagesize($uverz."/".$file);
```

Die Datei *vorschau.php* kann entsprechend des Scripts nur für GIF- und JPG-Grafiken Vorschaubilder erstellen. Diese Dateitypen haben die Kennziffern 1 und 2. Die if-Bedingung prüft diese Kennziffern, damit die nachfolgenden Befehle keine leeren Grafiken anzeigen.

```
if($test[2]==1 OR $test[2]==2)
{
```

Nur vor der ersten Spalte benötigen wir das Tag für eine neue Zeile:

```
if($spalte==1){echo "<tr>";}
```

Nun rufen wir die oben geschriebene Funktion, die den HTML-Text für eine Zelle erstellt, auf und lassen den Rückgabewert ausgeben. Mit `$file` wird der Dateiname übergeben, der in der Funktion durch die Übergabe in der Variablen `$bild` zur Verfügung steht. Außerdem setzen wir mit `$bildbreite` die Breite eines einzelnen Bildes ein, die nur (lokal) in der Funktion als `$breite` genutzt wird.

```
echo vorschaubild($file,$bildbreite);
```

Jetzt werden die Spalten gezählt. Nach jedem Bild muss der Spaltenzähler `$spalte` um eins hochgezählt werden (`$spalte++`). Wenn die letzte Spalte der Tabelle abgearbeitet ist (`$spalte` ist gleich `$gesamtspalten`, somit wird der else-Zweig ausgeführt), wird der Spaltenzähler `$spalte` wieder auf 1 gesetzt und das schließende `</tr>` ausgegeben, um das Ende der Zeile in der Tabelle anzuzeigen.

```
if($spalte<$gesamtspalten){$spalte++;}
else{$spalte=1; echo "</tr>";}
```

Danach folgen die schließenden Klammern für die if-Anweisung (`($test[2]==1 OR $test[2]==2)`) und die while-Schleife.

```
}}
```

Den Handle auf das Verzeichnis können wir nun schließen, da er seinen Dienst getan hat.

```
closedir($verzeichnis);
```

Das schließende `</table>`-Tag beendet die Tabelle.

```
echo "</table>";
```

Nach einem Zeilenumbruch werden die Hyperlinks ausgegeben, die diese tabellarische Auflistung unterschiedlich breit und mit drei oder vier Spalten aufrufen.

```
echo "<br>";

echo "<a href=$PHP_SELF?br=700&sp=4>4 Spalten 700px</a> -- ";
echo "<a href=$PHP_SELF?br=400&sp=4>4 Spalten Schmal</a> -- ";
echo "<a href=$PHP_SELF?br=700&sp=3>3 Spalten Breit</a> -- ";
echo "<a href=$PHP_SELF?br=400&sp=3>3 Spalten Schmal</a>";
```

Kapitel 14
Webseiten schützen

Sie kennen geschützte Webseiten sicherlich von Ihren Streifzügen als Surfer durch das Internet. Auf jeder Seite, die was auf sich hält, finden Sie einen Link mit der Aufschrift LOGIN, der Sie auf eine Webseite bringt, auf der Sie Ihren Benutzernamen und Ihr Kennwort – sofern vorhanden – eingeben können, um Zutritt zu den Webseiten zu erhalten, die nur für registrierte Benutzer gedacht sind.

In diesem Kapitel werden wir die benötigten Scripte für einen Zugriffsschutz erklären, den Sie für beliebig viele Webseiten einsetzen können. Um den Kreis der User zu verwalten, erstellen wir eine neue Webseite, auf der Sie neue User anlegen, vorhandene löschen und bearbeiten können.

In den vorhergehenden Kapiteln wurden bereits Webseiten erstellt, die schützenswert sind. Sie können z.B. die Uploadseiten nur einem eingeschränkten Kreis an auserwählten Usern zugänglich machen, damit Dateien nicht von Gott und der Welt hochgeladen werden können. Auch die Bilder Ihres Urlaubs sind auf einer geschützten Seite mitunter besser aufgehoben; teilen Sie Ihren Bekannten und Verwandten einen Benutzernamen und ein Kennwort mit, damit Ihre Schnappschüsse nicht in falsche Hände fallen.

Insbesondere der Schutz von Administrationsseiten ist zu empfehlen; lesen Sie dazu den Abschnitt *Besondere Rechte für den Administrator*.

Achtung

Bei den hier gezeigten Scripten greifen wir nicht auf die *Session-Funktionen* zurück, da sie in PHP3 noch nicht etabliert sind. Aus diesem Grund sind die Scripte mitunter etwas umfangreicher, als es mit der *Session-Funktion* notwendig gewesen wäre. Allerdings werden im Beispiel einige Features integriert, die auch mit den *Session-Funktionen* nicht einfach zu erstellen gewesen wären.

Die Zugriffskontrolle gewährleistet, dass sich jeder User nur genau einmal anmelden kann. Es ist nicht möglich, dass ein User sich in zwei Browserfenstern – weder auf einem Rechner noch auf mehreren Rechnern – mehrfach anmeldet. Allerdings kann man diese Einschränkung, die aus Sicherheitsgründen sinnvoll und gewollt ist, teilweise umgehen, indem man den Hyperlink einer Seite mit Hilfe des Kontextmenüs des Links in einem neuen Fenster anzeigen lässt. Für das neue Browserfenster wird dann die bestehende Anmeldung wieder verwendet. Jedoch unterscheidet sich das Verhalten von Browser zu Browser: Einige Browser erfordern das Aufrufen neuer Fenster wie oben beschrieben, um

zwei Seiten mit gleicher Anmeldung zu sehen, andere übernehmen beim Öffnen eines neuen Browserfensters einfach die Anmeldung des ersten Browserfensters.

Wie soll der Zugriffsschutz funktionieren?

In einer Tabelle mit dem Namen *user* speichern wir die Usernamen und Passwörter aller Benutzer. Auf der *Login-Seite* gibt der Surfer seinen Benutzernamen und sein Kennwort in einem Formular ein. Nachdem das Formular abgeschickt wurde, werden diese Angaben mit den gespeicherten Datensätzen der Tabelle *user* verglichen. Sind die Angaben des Surfers in einem Datensatz enthalten, haben Sie es mit einen berechtigten User zu tun.

Weiterhin wird dadurch getestet, ob bereits ein Surfer mit diesem Usernamen angemeldet ist. Dieser Test wird anhand einer Sitzungs-ID durchgeführt, die als Cookie auf dem Rechner des Users und in der Tabelle gespeichert wird. Wenn bereits eine Sitzungs-ID in der Tabelle vorliegt, wird dementsprechend davon ausgegangen, dass ein User mit diesem Benutzernamen eingeloggt ist, es sei denn, der letzte Zugriff auf eine geschützte Seite ist bereits einige Minuten her. In diesem Fall kann davon ausgegangen werden, dass der User sich nicht ausgeloggt hat und daher wird eine neue Anmeldung zugelassen.

Damit der User seine Angaben nicht auf jeder Seite erneut eingeben muss, werden zwei Cookies gesetzt, die anzeigen, dass der User sich bereits erfolgreich angemeldet hat: Das Cookie `IDuser` mit der ID-Nummer des Datensatzes des Users in der Tabelle *user* und `sit` mit der Sitzungs-ID.

Ruft ein Surfer eine geschützte Webseite auf, wird nachgeschaut, ob diese Cookies vorhanden sind. Wenn ja, ist der User prinzipiell berechtigt und man muss prüfen, ob die SitzungsID mit der in der Datenbank für diesen User gespeicherten übereinstimmt und ob seit dem letzten Aufruf einer geschützten Seite nicht zu viel Zeit vergangen ist. Werden diese Tests bestanden, wird die Seite angezeigt, andernfalls ist er nicht berechtigt und der Surfer wird zum Formular zum Anmelden umgeleitet.

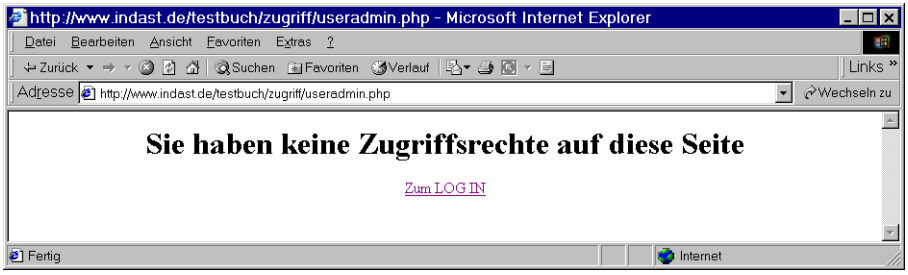

Bild 14.1: Der Zugriff auf eine Webseite wurde verweigert.

Die Zutaten

Um dieses Verfahren zu realisieren, brauchen Sie also:

- Eine Tabelle *user* mit den Usernamen und Passwörtern
- Eine Webseite für den *Login* der User und logischerweise eine Webseite für den *Logout*
- Ein Script, das Sie in jede zu schützende Webseite einfügen, um die Zugriffsrechte zu testen

Die User in der gleichnamigen Tabelle müssen Sie als Administrator verwalten. Als Administrator legen Sie neue User an, ändern die Passwörter und Benutzernamen und löschen vorhandene User. Dafür notwendig sind:

- Eine Webseite, auf der Sie einzelne User auswählen können, um die Verwaltungsaufgaben durchzuführen.
- Aus Gründen der Bequemlichkeit außerdem: Eine Datei mit den Zugangsdaten zur Datenbank, sowie einige Funktionen, u.a. die für den Zugriffsschutz. Dieses Verfahren erspart Tipparbeit.

Besondere Rechte für den Administrator

In Ihrem gesamten Webprojekt gibt es höchstwahrscheinlich Seiten, die nur für Sie als Administrator zugänglich sein sollen. Eine solche Seite ist die Webseite zum Verwalten der User, die wir in diesem Kapitel erstellen.

Weitere Seiten könnten die Verwaltungsseiten der Datei-Uploads (*Kapitel 11 und 12*) oder des Gästebuchs (*Kapitel 10*) sein. Damit Sie diese Seiten auch mit dem hier erstellten Zugriffsschutz versehen können, teilen wir die User in zwei Gruppen (normale User und Administratoren). Sie können dann beim Implementieren des Zugriffschutzes in die einzelnen Webseiten festlegen, ob alle bekannten User auf die Seite zugreifen dürfen oder nur Administratoren. Um festzulegen, welcher dieser beiden Gruppen ein User angehört, wird in der Tabelle *user* ein weiteres Feld hinzugefügt, das diese Information aufnimmt.

Sicherheit

Das hier vorgestellte Verfahren zum Schutz Ihrer Seiten stellt bereits einen guten Schutz der betreffenden Webseiten dar. Es ist zwar noch kein hundertprozentiger Schutz, aber wo finden Sie den im Internet?

Auf jeden Fall wird es so nicht möglich sein, Ihre geschützten Webseiten ohne erheblichen Aufwand zu knacken. Ein einfaches Betrachten der Informationen, die in den Cookies auf dem Rechner des Surfers gespeichert werden, wird z.B. – wie es bei vielen Webseiten immer noch der Fall ist – keine verwertbaren Informationen zum Knacken der Webseite verraten.

Gegen die Weitergabe von Benutzernamen und Kennwörtern durch die User selber ist aber der beste Schutzmechanismus hilflos.

Die Sicherheit der hier vorgestellten Lösung lässt sich noch durch weitere Features erhöhen, die wir in diesem Rahmen aber nicht weiter erklären können. Wir geben an dieser Stelle nur ein paar allgemeine Hinweise:

- Verwenden Sie SSL-Verbindungen, um ein »Abhören« der übertragenen Zugangsdaten Ihrer User zu vermeiden
- Begrenzen Sie die Zahl der Anmeldeversuche je User. Wenn diese Anzahl überschritten ist, sperren Sie den Account
- Begrenzen Sie die Zeit der Inaktivität auf eine kurze Zeitspanne. Ist diese Dauer überschritten, wird der User zum erneuten Anmelden aufgefordert
- Fragen Sie nach dem Zufallsprinzip nur bestimmte Zeichen des Passwortes ab: Mal das erste und dritte Zeichen, ein anderes Mal das sechste und vierte.
- Erzwingen Sie eine Neuanmeldung, wenn sich die IP-Adresse des Users ändert.

Wenn Sie diese zusätzlichen Features integrieren, sind Sie schon dicht an dem Verfahren, das Banken für browserbasiertes Online-Banking verwenden.

Bedenken Sie aber in jedem Fall, dass nur Seiten geschützt werden, in die Sie das Script zum Schutz eingebaut haben. Es ist nicht ausreichend, die Startseite zu schützen, auf der alle Hyperlinks zu Ihren weiteren Seiten liegen. In diesem Fall kann der böswillige Surfer zwar nicht per Hyperlink auf Ihre weiteren Seiten zugreifen, aber mit etwas Mühe oder einem kleinen Script kann er sich die URLs der weiteren Seiten besorgen und diese ungeschützten Seiten einfach durch Eingabe der URL in die Adressleiste des Browsers aufrufen. Je nach Konfiguration des Webservers reicht es mitunter auch aus, das Verzeichnis ohne Dateinamen in den Browser einzugeben, um eine Liste aller enthaltenen Dateien zu erhalten.

In diesem Script wird das Kennwort des Users mit `md5` verschlüsselt in der Datenbank gespeichert, sodass auch Sie als Administrator das Kennwort, nachdem Sie es eingegeben haben, nicht mehr unverschlüsselt einsehen können. Dieses Verfahren bietet einen guten Schutz, falls Ihre Datenbank einmal geknackt werden sollte, da der Einbrecher die Kennwörter auch nicht entschlüsseln kann. Allerdings erfordert dieses Verfahren einige Besonderheiten beim Ändern der Kennwörter, die wir mit dem Script erklären.

Die Tabelle erstellen

Die Tabelle, in der die Informationen der User gespeichert werden, ist am einfachsten mit *phpMyAdmin* zu erstellen. Legen Sie eine neue Tabelle mit sieben Feldern an. Im Beispiel nennen wir die Tabelle *user*. Entnehmen Sie die benötigten Felder der Tabelle 14.1. Das Bild 14.2 zeigt das Formular zum Erstellen der Tabelle in *phpMyAdmin*.

Feld	Beschreibung
ID	eindeutige Identifizierung des Datensatzes durch Autozähler
username	Feld des Typs CHAR, in dem der Usernamen/Benutzernamen gespeichert wird. Damit bei Abfragen die Groß/Kleinschreibung unterschieden wird, wird das Attribut BINARY gesetzt. Weiterhin wird es UNIQUE gekennzeichnet, damit es keine zwei User mit gleichlautenden Benutzernamen geben kann. Die Datenbank verweigert bei gleichlautenden Benutzernamen das Anlegen oder Ändern des Datensatzes. Als Standardwert wird NEUERUSER gesetzt, um das Anlegen neuer User zu vereinfachen.
kwort	Feld des Typs CHAR, in dem das Kennwort des Users gespeichert wird. Auch hier wird das Attribut BINARY aus den gleichen Gründen wie zuvor gesetzt und ein Standardwert (kennwort), um das Anlegen neuer User zu vereinfachen.
admin	Feld des Typs TINYINTEGER mit einer Länge von zwei, wobei negative Werte zugelassen werden. Der Standardwert wird -1, womit die Zugehörigkeit zu den Standardusern angezeigt wird, für Administratoren wird später der Wert 1 verwendet.
beschreibung	Ein Textfeld (CHAR), in dem Sie einen kleinen Text über den User ablegen können, damit Sie sich besser an ihn oder seine »Verfehlungen« erinnern können.
kennung	speichert die Kennung, die den User je Sitzung identifiziert
zeit	Zeitpunkt des letzten Aufrufs einer geschützten Seite durch den angemeldeten User

Tabelle 14.1: Die Felder der Tabelle user

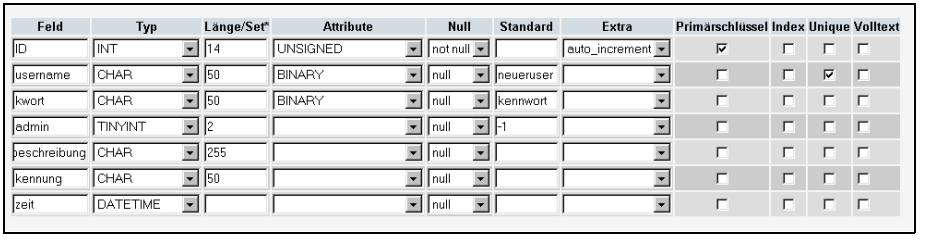

Bild 14.2: Das Formular zum Anlegen der Tabelle USER *in* PHPMYADMIN

Alternativ zur Arbeit mit *phpMyAdmin* können Sie auch den folgenden SQL-String an die Datenbank schicken:

```
CREATE TABLE `user` (
`ID` INT(14) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
`username` CHAR(50) BINARY DEFAULT 'neueruser',
`kwort` CHAR(50) BINARY DEFAULT 'kennwort',
`admin` TINYINT(2) DEFAULT '-1',
`beschreibung` CHAR(255),
`kennung` CHAR(50),
`zeit` DATETIME;
UNIQUE (`username`)
);
```

Eine Webseite zum Verwalten der User

Eine kurze Definition vorweg: Ein Zugriffsschutz macht nur Sinn, wenn man bestimmen kann, wer berechtigt ist, Seiten anzuschauen. Diese ausgewählten Personen nennen wir im Folgenden User. Ein User identifiziert sich durch seinen Benutzernamen und sein Kennwort. Um Webseiten zu schützen, benötigen Sie also eine Liste der User, die Sie in der zuvor erstellten Tabelle *user* speichern.

In diesem Abschnitt schreiben wir das Script, mit dem Sie die Liste der User bearbeiten können. Dieses Script wird im Beispiel in der Datei *useradmin.php* gespeichert.

Im Einzelnen gehören zu einer Userverwaltung folgende Aufgaben:

- Anlegen neuer User
- Ändern des Usernamens
- Ändern des Passworts
- Ändern der Gruppenzugehörigkeit
- Ändern des beschreibenden Textes eines Users
- Löschen eines vorhandenen Users

Diese Aufgaben erfordern – bis auf das Anlegen eines neuen Users – , dass Sie zunächst den User auswählen können, dessen Angaben Sie bearbeiten oder den Sie löschen möchten.

In Listing 14.1 sehen Sie das gesamte Script der Seite. In den nachfolgenden Abschnitten werden die einzelnen Funktionsbereiche (Anlegen eines Users, vorhandene User-Informationen ändern, Löschen eines Users) erläutert.

Bild 14.3: Die Webseite für die Userverwaltung mit einem zum Bearbeiten ausgewählten User

```
<html><head><title>Userverwaltung</title></head><body><div align=center>
<?php

require('connect.php');
$tabellenname='user';

//Anlegen eines neuen Users
if($sent==1)
{
$sql="INSERT INTO $tabellenname () VALUES () ";
mysql_query($sql,$link);
if(mysql_insert_id()>0)
{
$ID=mysql_insert_id();
$meldung="Der neue User wurde angelegt<br>";
```

```
}
else
{
$meldung="Es konnte kein neuer User angelegt werden. Bearbeiten Sie zuvor neu
angelegte User, und ändern Sie den Usernamen, bevor Sie einen weiteren neuen
User anlegen.<br>";}
}

//Speichern der bearbeiteten Userinformationen
if($sent==2)
{
if($kwort!='' AND (strlen($kwort)<6 OR strlen($kwort)>12))
{
$meldung.="Das Kennwort hat nicht die richtige Länge<br>";
unset($kwort);
}
$sql="UPDATE $tabellenname SET ";
if($kwort!='')
{
$kwort2=md5($kwort);
$sql.=" kwort='$kwort2', ";
}
$sql.=" username='$username', ";
$sql.=" admin='$admin', ";
$sql.=" beschreibung='$beschreibung' ";
$sql.=" WHERE ID='$ID' ";
$test=mysql_query($sql,$link);
if($kwort!='' AND !$test){$meldung.="Die Änderungen konnten nicht gespeichert
werden, das alte Kennwort ist noch gültig<br>";}
if($kwort!='' AND $test){$meldung.="Das neue Kennwort - $kwort - ist jetzt
gültig<br>";}
}

//Löschen eines Users
if($sent==3)
{
$sql="DELETE FROM $tabellenname WHERE ID=$ID ";
mysql_query($sql,$link);
unset($ID);
}

echo "<h2><font color=blue>$meldung</font></h2>";

//Auslesen der vorhandenen User und Auswahlformular anzeigen
$sql="SELECT * FROM $tabellenname ORDER BY username ";
```

```
$result=mysql_query($sql,$link);
if(mysql_num_rows($result)>0)
{
echo "<h2>Bitte den User zum Bearbeiten auswählen</h2>";
echo "<form action='$PHP_SELF' method='post'>";
echo "<select name='ID'>";
for($i=0;$i<mysql_num_rows($result);$i++)
{
$ID1=mysql_result($result,$i,'ID');
$uname=mysql_result($result,$i,'username');
echo "<option value=$ID1";
if($ID==$ID1){echo " selected ";}
echo ">$uname";
}
echo "</select><br><br>";
echo "<input type=submit>";
echo "</form>";
}

//Das Formular zum Bearbeiten
if($ID)
{
$sql="SELECT * FROM $tabellenname WHERE ID=$ID ";
$result=mysql_query($sql,$link);
if(mysql_num_rows($result)==1)
{
$ID1=mysql_result($result,0,'ID');
$username1=mysql_result($result,0,'username');
$kwort1=mysql_result($result,0,'kwort');
$admin1=mysql_result($result,0,'admin');
$beschreibung1=mysql_result($result,0,'beschreibung');
echo "<h2>Bitte den User $username1 bearbeiten</h2>";
echo "<form action='$PHP_SELF' method='post'>";
echo "<input type=hidden name=sent value=2>";
echo "<input type=hidden name=ID value=$ID1>";
echo "<p>Benutzername</p>";
echo "<input type=text name=username value='$username1'>";
echo "<p>Kennwort  <br><font color=red>Bitte leer lassen, wenn es nicht geändert
werden soll!</font></p>";
if($kwort1=='kennwort'){echo "<p><font color=red size=+1>Sie müssen das Kennwort
noch angeben, damit der User sich anmelden kann.</font></p>";}
echo "<input type=text name=kwort value=''>";
echo "<p>Beschreibung</p>";
echo "<input type=text name=beschreibung value='$beschreibung1'>";
echo "<p>Administrator</p>";
```

```
echo "<input type=radio name=admin value=-1 ";
if($admin1==-1){echo " checked ";}
echo ">NEIN --- ---";
echo "<input type=radio name=admin value=1 ";
if($admin1==1){echo " checked ";}
echo ">JA";
echo "<br><br><input type=submit>";
echo "</form>";

echo "<br><a href='$PHP_SELF?sent=3&ID=$ID1'><font size=+1>Den angezeigten User
löschen</font></a>";

}}

echo "<br><a href='$PHP_SELF?sent=1'><font size=+1>Einen neuen User anlegen</
font></a>";

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

Listing 14.1: Das gesamte Script der Userverwaltung der Datei useradmin.php

Die Verbindung zum Datenbankserver und das Auswählen der Datenbank haben wir auch für die Userverwaltung in eine externe Datei (*connect.php*) ausgelagert, die wir mit `require()` einbinden. In der Datei sind nur drei Zeilen enthalten, wie das Listing 14.1 zeigt. In der weiteren Beschreibung werden wir dieser Datei zwei Funktionen hinzufügen.

Der Tabellenname wird in diesem Fall nicht hier definiert sondern in der jeweils aufrufenden Datei.

In der Variablen `$idle` legen wir die Zeit in Minuten fest, die ein User ohne nochmaliges Anmelden zwischen zwei Aufrufen geschützter Webseiten verstreichen lassen darf. Diese Angabe wird hier zentral gesetzt, da Sie diesen Wert dann anpassen können, ohne durch alle Scripte blättern zu müssen.

```
<?php
$link = mysql_connect("localhost", "username", "passwort");
mysql_select_db("phpbuch", $link);
$idle=10;
?>
```

Listing 14.2: Die Verbindung zur Datenbank und $idle werden zentral in der Datei connect.php definiert.

Neue User anlegen

Für das Anlegen eines neuen Users reicht ein einfacher Link, der das Script nochmals aufruft. Mit dem Aufruf wird `sent=1` übergeben, um anzuzeigen, dass ein neuer User angelegt werden soll. Den Link finden Sie so ziemlich am Ende des Scripts.

```
echo "<br><a href='$PHP_SELF?sent=1'><font size=+1>Einen neuen User anlegen</
font></a>";
```

Wenn beim Aufrufen des Scripts – wie es durch den erstellten Link geschieht – `$sent==1` ist, wird der Anweisungsblock ausgeführt, der einen neuen Datensatz in der Tabelle *user* anlegt. Dieser Datensatz wird mit den Standardwerten, die wir bei der Definition der Tabelle gesetzt haben, gefüllt. Deswegen sind im SQL-String die Klammern, die die Feldnamen und die zugewiesenen Werte aufnehmen, leer.

```
if($sent==1)
{
$sql="INSERT INTO $tabellenname () VALUES () ";
mysql_query($sql,$link);
```

War das Anlegen des neuen Datensatzes erfolgreich, gibt der Befehl `mysql_insert_id()` die in das Autozählerfeld *ID* automatisch eingetragene ID-Nummer zurück. Da das Autozählerfeld bei 1 zu zählen beginnt, muss dieser Wert im Erfolgsfall größer `0` sein. Wir schreiben eine Erfolgsmeldung in die Variable `$meldung` und weisen der Variablen `$ID` den Wert von `mysql_insert_id()` zu. Dadurch, dass `$ID` die ID-Nummer des Datensatzes zugewiesen wird, wird direkt nach dem Anlegen eines neuen Users das Formular zum Bearbeiten mit den Informationen des neuen Users angezeigt. Dieses Formular ist somit immer sichtbar, wenn `$ID` gesetzt ist.

Für den Fall, dass der neue Datensatz nicht angelegt werden konnte, schreiben wir eine Fehlermeldung in `$meldung`.

```
if(mysql_insert_id()>0)
{
$ID=mysql_insert_id();
$meldung="Der neue User wurde angelegt<br>";
}
else
{
$meldung="Es konnte kein neuer User angelegt werden. Bearbeiten Sie zuvor neu
angelegte User, und ändern Sie den Usernamen, bevor Sie einen weiteren neuen
User anlegen.<br>";}
}
```

Da der neue Datensatz mit den Standardwerten der Tabelle gefüllt ist, können Sie keine zwei neuen User nacheinander anlegen. Im Klartext: Sie müssen zunächst den Benutzernamen des neuen Users ändern, um einen weiteren neuen User anlegen zu dürfen, da das Feld *username* der Tabelle nicht zwei identische Einträge zulässt.

Das Anlegen des neuen Users sollten Sie im Script vor dem Auslesen und Anzeigen des Formulars zum Bearbeiten eines ausgewählten Users durchführen, damit dieser User, nachdem er erfolgreich angelegt wurde, durch die oben vorgenommene Zuweisung des Autozählers an `$ID` automatisch zum Bearbeiten ausgewählt wird.

Die Meldungen, die Sie in die Variable `$meldung` schreiben, werden im Script nach dem Bereich, in dem das Anlegen, Bearbeiten sowie Löschen von Datensätzen durchgeführt wird, aber vor der Ausgabe der Formulare, mit `echo` ausgegeben.

```
echo "<h2><font color=blue>$meldung</font></h2>";
```

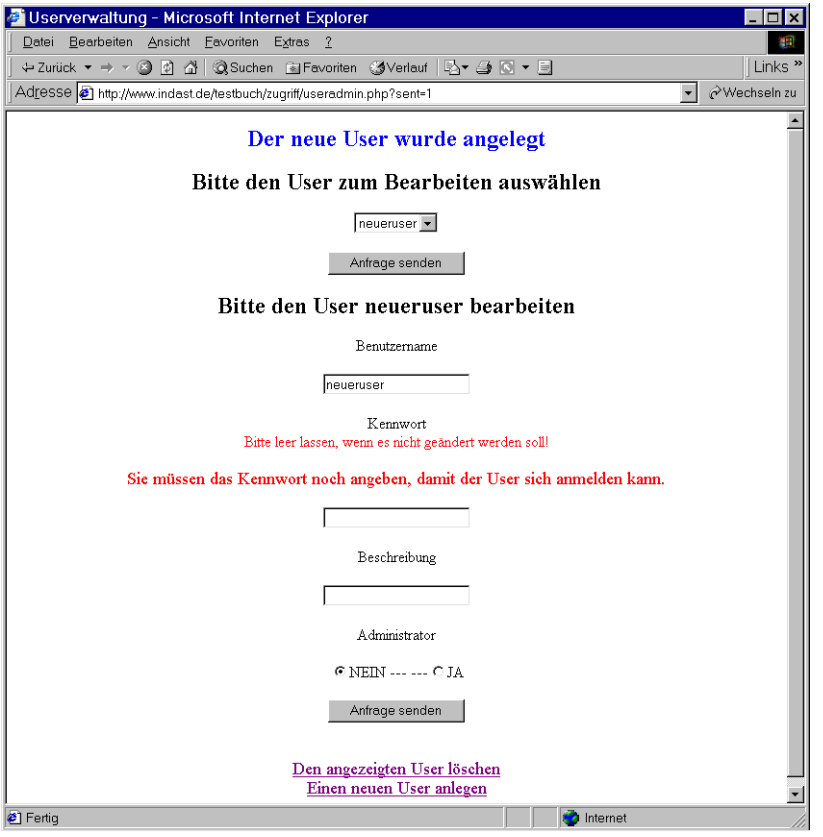

Bild 14.4: Ein neuer User wurde angelegt. Das Formular zum Bearbeiten des neuen Users wird automatisch eingeblendet. In den Feldern des Formulars finden Sie die in der Datenbank definierten Standardwerte wieder.

Die Userangaben bearbeiten

Um die Informationen – Benutzername, Kennwort, Gruppe (Admin oder nicht Admin) und den beschreibenden Text – eines Users zu ändern, sind drei Schritte notwendig:

- Auswählen des Users in einem Formular, dessen Angaben Sie ändern möchten.
- Wenn ein User ausgewählt ist, sollen die bisherigen Informationen in einem weiteren Formular angezeigt werden, in dem Sie die Angaben ändern können.
- Nachdem Sie das zweite Formular abgeschickt haben, werden die Änderungen in die Datenbank geschrieben.

In diese drei Schritte untergliedert erläutern wir nun die entsprechenden Scriptfragmente.

Auswählen des Users

Zunächst lassen wir alle vorhandenen User aus der Tabelle *user* auslesen.

```
$sql="SELECT * FROM $tabellenname ORDER BY username ";
$result=mysql_query($sql,$link);
```

Wenn mindestens ein User vorhanden ist, wird das Formular zur Auswahl des Users, dessen Angaben Sie bearbeiten möchten, angezeigt.

```
if(mysql_num_rows($result)>0)
{
echo "<h2>Bitte den User zum Bearbeiten auswählen</h2>";
echo "<form action='$PHP_SELF' method='post'>";
```

In einem `select`-Feld werden alle User aufgelistet, Die `<option>`-Tags erzeugen wir mithilfe einer for-Schleife, die durch jeden Datensatz der Ergebnisliste läuft.

```
echo "<select name='ID'>";
for($i=0;$i<mysql_num_rows($result);$i++)
{
```

Als Wert der Optionen tragen wir die ID-Nummer des Datensatzes aus der Ergebnisliste, der zuvor der Variablen `$ID1` zugewiesen wurde, ein. Angezeigt wird der Benutzername im Feld USERNAME.

Sollte bereits durch einen vorherigen Aufruf der Seite ein User ausgewählt sein, steht die ID-Nummer in `$ID`. Ist dies der Fall, wird über die if-Anweisung dieser zuvor schon ausgewählte Datensatz/User im `select`-Feld mit `selected` im `<option>`-Tag vorausgewählt.

```
$ID1=mysql_result($result,$i,'ID');
$uname=mysql_result($result,$i,'username');
echo "<option value=$ID1";
if($ID==$ID1){echo " selected ";}
echo ">$uname";
}
```

```
echo "</select><br><br>";
echo "<input type=submit>";
echo "</form>";
}
```

Abschließend wird die Submit-Schaltfläche angezeigt und das Formular beendet.

Bild 14.5: Wenn bereits ein User ausgewählt wurde, ist er im Auswahlfeld vorausgewählt.

Das Formular zum Bearbeiten eines ausgewählten Users

Das Formular wird immer angezeigt, wenn `$ID` gesetzt ist. `$ID` ist gesetzt, wenn Sie im zuvor beschriebenen Formular einen User ausgewählt, einen neuen User angelegt oder dieses Formular abgeschickt haben.

Dadurch, dass der gleiche User in diesem Formular erneut angezeigt wird, wenn Sie Änderungen abgeschickt haben, können Sie anschließend kontrollieren, ob die Änderungen zu Ihrer Zufriedenheit ausgeführt wurden. Eine Sonderrolle nimmt hier das Kennwort ein – dazu in der folgenden Erklärung mehr.

```
if($ID)
{
```

Zunächst werden die Informationen des Datensatzes mit der ID-Nummer, die in `$ID` gespeichert ist, ausgelesen.

```
$sql="SELECT * FROM $tabellenname WHERE ID=$ID ";
$result=mysql_query($sql,$link);
```

Es darf nur einen Datensatz geben, da die ID eindeutig ist. Wenn dies nicht der Fall ist, liegt ein Fehler vor und das Formular wird nicht angezeigt.

```
if(mysql_num_rows($result)==1)
{
```

Die Informationen des Datensatzes aus der Ergebnisliste schreiben wir in Variablen. Die Variablen haben die gleichen Namen wie die Datenfelder, sind aber mit einer 1 versehen, um versehentliches Überschreiben beim Abschicken und Verarbeiten zu vermeiden.

```
$ID1=mysql_result($result,0,'ID');
$username1=mysql_result($result,0,'username');
$kwort1=mysql_result($result,0,'kwort');
$admin1=mysql_result($result,0,'admin');
$beschreibung1=mysql_result($result,0,'beschreibung');
```

Anschließend wird das Formular ausgegeben. In versteckten Feldern setzen wir `sent=2`, um nach dem Abschicken des Formulars anzuzeigen, dass ein Datensatz geändert und die ID-Nummer des Datensatzes übergeben wurde.

```
echo "<h2>Bitte den User $username1 bearbeiten</h2>";
echo "<form action='$PHP_SELF' method='post'>";
echo "<input type=hidden name=sent value=2>";
echo "<input type=hidden name=ID value=$ID1>";
```

In den Feldern des Formulars sorgen wir mit den aus der Datenbank gewonnenen Informationen mithilfe des `value`-Attributs für eine Vorbelegung. Beim `radio`-Feld kommt eine if-Anweisung zum Einsatz, um das Attribut `checked` in das richtige Feld zu setzen.

```
echo "<p>Benutzername</p>";
echo "<input type=text name=username value='$username1'>";
echo "<p>Kennwort  <br><font color=red>Bitte leer lassen, wenn es nicht geändert
werden soll!</font></p>";
if($kwort1=='kennwort'){echo "<p><font color=red size=+1>Sie müssen das Kennwort
noch angeben, damit der User sich anmelden kann.</font></p>";}
echo "<input type=text name=kwort value=''>";
echo "<p>Beschreibung</p>";
echo "<input type=text name=beschreibung value='$beschreibung1'>";
echo "<p>Administrator</p>";
echo "<input type=radio name=admin value=-1 ";
if($admin1==-1){echo " checked ";}
echo ">NEIN --- ---";
echo "<input type=radio name=admin value=1 ";
if($admin1==1){echo " checked ";}
echo ">JA";
echo "<br><br><input type=submit>";
echo "</form>";
```

Hinweise zum Kennwort

Die oben erwähnte Sonderrolle des Kennworts beginnt hier im Formular. Das Kennwort wird nicht vorbelegt, da in der Datenbank der `md5`-verschlüsselte Wert steht. Mit diesem kryptischen Wert können Sie als Administrator nichts anfangen.

Hinweis

```
string md5(string text)
```

Die Funktion verschlüsselt den Text (text) nach dem md5-Algorithmus.

Deswegen wird das Kennwort später beim Speichern der Änderungen in der Datenbank nur geändert, wenn Sie einen Wert in das Feld KWORT eingeben. Lassen Sie – wie der Hinweistext im Formular es auch besagt – dieses Feld leer, wenn das Kennwort nicht geändert werden soll.

Eine weitere Besonderheit des Kennwortes: Wenn Sie einen neuen Datensatz einrichten, wird der Standardwert in das Feld KWORT der Datenbank geschrieben. Dieser Wert ist nicht `md5`-verschlüsselt, sodass regelrecht KENNWORT gespeichert wird. Mit diesem Kennwort kann sich aber niemand anmelden, da wir im Anmeldescript, das wir weiter unten erstellen, davon ausgehen, dass das Kennwort im Feld KWORT verschlüsselt ist. Aus diesem Grunde müssen Sie für jeden neuen User ein Kennwort neu anlegen, das dann `md5`-verschlüsselt gespeichert wird – was aus Sicherheitsgründen auch sinnvoll ist.

Die Änderungen der Userinformationen in der Datenbank speichern

Die Änderungen an den User-Informationen werden beim erneuten Aufruf der Seite nach dem Abschicken des zweiten Formulars gespeichert. Dieses Scriptfragment sollte vor dem Auslesen der User-Informationen stehen, damit immer die aktuellen Informationen angezeigt werden. Das Speichern findet nur statt, wenn `$sent==2` ist, so wie es zuvor im Formular gesetzt wurde.

```
if($sent==2)
{
```

Zunächst prüfen wir, ob ein Eintrag im Feld KWORT vorliegt und ob dessen Länge außerhalb der zulässigen Grenzen (6 und 12) ist. Für den Fall, dass dies so ist, schreiben wir eine Fehlermeldung in `$meldung`, und löschen das Kennwort in `$kwort`, damit das fehlerhafte Kennwort nicht in die Datenbank geschrieben wird. Die restlichen Änderungen werden aber durchgeführt. An dieser Stelle können Sie weitere Einschränkungen des Kennworts hinzufügen. Sie könnten z.B. mit Hilfe eines *regulären Ausdrucks* die erlaubten Zeichen beschränken. (Im nächsten Kapitel finden Sie eine ausführliche Beschreibung eines regulären Ausdrucks, der eine E-Mail-Adresse testet. Die Sache ist etwas verzwickt, aber sehr wirkungsvoll!)

```
if($kwort!='' AND (strlen($kwort)<6 OR strlen($kwort)>12))
{
$meldung.="Das Kennwort hat nicht die richtige Länge<br>";
unset($kwort);
}
```

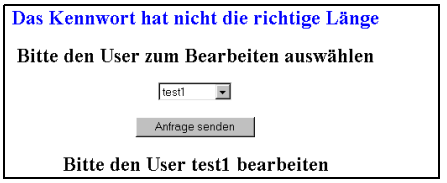

Bild 14.6: Der Versuch, ein Kennwort zu ändern, ist fehlgeschlagen.

Nach diesem Test wird der `SQL-UPDATE`-String erstellt. Das Kennwort wird, falls vorhanden, mit dem Befehl `md5()` verschlüsselt und der benötigte Teil des SQL-Strings zum Speichern des Kennworts an `$sql` angehängt. Dieser Teil des SQL-Strings wird im Umkehrschluss also weggelassen, wenn `$kwort` leer ist. Auf diese Art wird das bereits gespeicherte Kennwort nicht überschrieben, wenn `$kwert` leer ist.

Dies ist der Fall, wenn das Kennwort den Test zuvor nicht bestanden hat oder wenn Sie das Kennwortfeld im Formular leergelassen haben.

```
$sql="UPDATE $tabellenname SET ";
if($kwort!='')
{
$kwort2=md5($kwort);
$sql.=" kwort='$kwort2', ";
}
```

Anschließend vervollständigen wir den SQL-String und senden ihn an die Datenbank.

```
$sql.=" username='$username', ";
$sql.=" admin='$admin', ";
$sql.=" beschreibung='$beschreibung' ";
$sql.=" WHERE ID='$ID' ";
$test=mysql_query($sql,$link);
```

Jetzt generieren wir noch zwei Meldungen in Bezug auf Kennwortänderungen. `$test` ist `true`, wenn der SQL-String erfolgreich von der Datenbank verarbeitet werden konnte. Somit wurde das Kennwort geändert, wenn `$kwort` nicht leer ist und der SQL-String ausgeführt werden konnte.

```
if($kwort!='' AND $test){$meldung.="Das neue Kennwort - $kwort - ist jetzt
gültig<br>";}
```

Wenn `$kwort` nicht leer ist und `$test` `false`, wurde versucht, das Kennwort zu ändern, aber irgendetwas ist schiefgegangen, sodass das alte Kennwort noch gültig ist.

```
if($kwort!='' AND !$test){$meldung.="Die Änderungen konnten nicht gespeichert
werden, das alte Kennwort ist noch gültig<br>";}
 }
```

In dieser Meldung ist das neue Kennwort das letzte Mal für Sie im Klartext sichtbar, merken Sie es sich gut bzw. geben Sie es an den User weiter. Anschließend können Sie es nicht mehr hervorzaubern – nur noch verschlüsselt, was aber weder Ihnen noch dem User weiterhilft.

Bild 14.7: Ein erfolgreicher Versuch, das Kennwort zu ändern. In der Meldung sehen Sie das Kennwort zum letzten Mal im Klartext.

Einen User löschen

Für das Löschen eines Users reicht ein einfacher Link, der die Seite erneut aufruft, die ID-Nummer des Datensatzes übergibt und mit `sent=3` anzeigt, dass der Datensatz gelöscht werden soll.

```
echo "<br><a href='$PHP_SELF?sent=3&ID=$ID1'><font size=+1>Den angezeigten User
löschen</font></a>";
```

Dieser Link muss in den Scriptbereich eingebaut werden, in dem `$ID` gesetzt ist, damit bekannt ist, welcher User gelöscht werden soll.

Das Löschen des Datensatzes ist ein kurzes SQL-Statement, das anweist, welcher Datensatz – der mit `ID=$ID` – gelöscht werden soll. Anschließend löschen wir `$ID`, damit nicht versucht wird, das Formular zum Bearbeiten dieses Datensatzes anzuzeigen. Dies würde mit der in `$ID` gespeicherten ID-Nummer keinen Sinn mehr machen, da dieser Datensatz soeben entfernt wurde.

```
if($sent==3)
{
$sql="DELETE FROM $tabellenname WHERE ID=$ID ";
mysql_query($sql,$link);
unset($ID);
}
```

Der Login der User

Damit ein Zugriffsschutz funktioniert, müssen die User sich anmelden können. Diese Anmelde-Webseite erstellen wir in diesem Abschnitt. Wir speichern dieses Script im Beispiel als *login.php*.

Damit die User sich nicht jedes Mal neu anmelden müssen, setzen wir nach erfolgreichem Anmelden zwei Cookies, die anzeigen, dass der User eingeloggt ist. Beim Abmelden – ausloggen – werden diese Cookies gelöscht. Wir verwenden also Cookies, die mit dem Schließen des Browsers gelöscht – also nur kurzfristig vom Browser gespeichert werden. In einem Cookie speichern wir die ID-Nummer des Users, in dem zweiten die Sitzungskennung.

In den Scripten wird häufiger der Benutzername des Users verwendet, deswegen erstellen wir zunächst eine Funktion, die den Benutzernamen ausliest.

Die Funktion benutzer()

Um dem User anzuzeigen, mit welchem Benutzernamen er eingeloggt ist, schreiben wir eine kleine Funktion, die mit Hilfe der im Cookie gespeicherten ID-Nummer den Benutzernamen zurückgibt. Fügen Sie die Funktion am besten in die Datei ein, die die Verbindung zur Datenbank herstellt, da die Funktion die Datenbankverbindung benötigt. Sie können dann auf diese Funktion in allen Webseiten zugreifen, in die Sie diese Datei – im Beispiel *connect.php* – einbinden.

```
function benutzer()
{
Global $link, $IDuser,$sit;
$tabellenname = 'user';
$sql="SELECT username FROM $tabellenname WHERE ID='$IDuser' AND kennung = $sit";
$result=mysql_query($sql,$link);
if(mysql_num_rows($result)==1)
{
return mysql_result($result,0,'username');
}}
```

Listing 14.3: Die Funktion benutzer() liefert den Benutzernamen zu einem angemeldeten Benutzer.

Die Funktion heißt `benutzer` und erwartet keine Parameter. Sie rufen sie also einfach mit `echo benutzer()` auf, um den Benutzernamen des eingeloggten Users anzuzeigen.

Zunächst werden die für die Funktion benötigten Variablen *Global* gesetzt. Dies sind `$link`, um eine Verbindung zur Datenbank herzustellen, und `$IDuser`, um auf die ID-Nummer des Users, die in dem Cookie dieses Namens gespeichert wird, zuzugreifen, sowie `$sit`, um die im gleichnamigen Cookie gespeicherte Sitzungskennung zu prüfen. Anschließend wird das Feld USERNAME für den über `$IDuser` bestimmten User aus der Datenbank ausgelesen. Das Ergebnis wird mit `return` zurückgegeben.

Die Login-Seite

Jede Benutzerverwaltung benötigt eine Seite zum Einloggen. Nachdem der Surfer die Informationen des Anmelde-Formulars ausgefüllt und abgeschickt hat, werden diese Angaben mit den Informationen der Datenbank verglichen. Sind beide Angaben in Ordnung, sollen einige Test durchgeführt werden, die mit der Sitzungsverwaltung und Zeitbegrenzung zusammenhängen:

- Ist der User bereits eingeloggt?
- Ist bereits jemand anderes mit diesem Usernamen eingeloggt?

Wenn alle Tests bestanden sind, setzen wir zwei Cookies: eins mit dem Namen `IDuser`, das die ID-Nummer des Datensatzes aufnimmt und eins mit dem Namen `sit`, das eine eindeutige Sitzungs-ID enthält. Beide Cookies zeigen an, dass der User sich erfolgreich eingeloggt hat. Wird ein Test nicht bestanden, erscheinen entsprechende Fehlermeldungen.

Die Erklärungen haben wir wegen der Komplexität des Scripts, das diese Aufgaben übernimmt, direkt zwischen die Programmzeilen geschrieben.

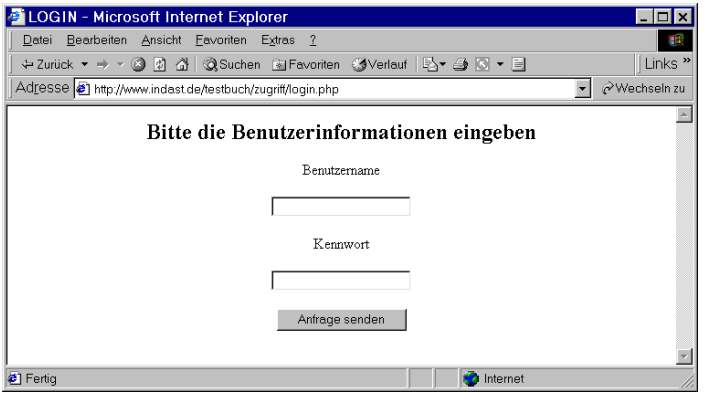

Bild 14.8: Das Formular zum Anmelden als registrierter User

```
<?php
require('connect.php');
$tabellenanme='user';
```

Die Verbindung zur Datenbank wird hergestellt und der Tabellenname in eine Variable geschrieben.

```
if($sent==1){
$sql="SELECT kwort, ID, kennung, (UNIX_TIMESTAMP()-UNIX_TIMESTAMP(zeit)) as
time, zeit FROM $tabellenname WHERE username='$username' ";
$result=mysql_query($sql,$link);
```

Wenn das Formular abgeschickt wurde, werden die Informationen des Users mit dem eingegebenen Usernamen aus der Datenbank gelesen. Mit `(UNIX_TIMESTAMP()-UNIX_TIMESTAMP(zeit)) as time` wird die Zeitdifferenz zwischen der aktuellen Zeit und dem letzten Aufruf einer geschützten Webseite durch den User in Sekunden berechnet und mit dem Feldnamen TIME an die Ergebnisliste übergeben. Die aktuelle UNIX-Zeit in Sekunden gibt `UNIX_TIMESTAMP()` zurück und `UNIX_TIMESTAMP(zeit)` ermittelt die UNIX-Zeit in Sekunden zum Zeitpunkt des letzten Aufrufs einer Seite, da im Feld ZEIT dieser Zeitpunkt festgehalten wird. Das Speichern der letzten Zugriffszeit in das Feld ZEIT geschieht im Script in der Funktion `recht()`, die den Zugriffsschutz in eine Webseite implementiert.

```
if(mysql_num_rows($result)==1){
```

Ist der Username in der Datenbank vorhanden, werden die folgenden Tests durchgeführt:

- ```
  if(md5($kwort)!=mysql_result($result,0,'kwort'))
  {$meldung.="Bitte kontrollieren Sie Ihr Kennwort";}
  else{
  ```

  Wurde das Kennwort richtig eingegeben? Wenn nicht, wird die Fehlermeldung erzeugt, ansonsten wird mit den Tests fortgefahren.

- ```
  if(mysql_result($result,0,'kennung')==$sit AND
  mysql_result($result,0,'time')/60<$idle)
  {$meldung.="Sie sind bereits eingeloggt";unset($username);}
  else{
  ```

  Ist der User bereits angemeldet? In diesem Fall ist die Sitzungs-ID des Cookies `sit` und der Datenbank identisch. Weiterhin wird geprüft, ob der letzte Seitenaufruf noch nicht zu lange her ist. Ist der User noch nicht angemeldet, wird weitergetestet.

- ```
  if(mysql_result($result,0,'kennung')!='' AND
  mysql_result($result,0,'time')/60<$idle)
  {$meldung.="Es ist bereits jemand mit Ihrem Benutzernamen eingeloggt";}
  else{
  ```

  Ist bereits ein User mit einer anderen Sitzungs-ID mit diesem Benutzernamen eingeloggt? In diesem Fall ist das Datenbankfeld KENNUNG nicht leer, da beim Ausloggen die Kennung gelöscht wird. Sollte noch eine Kennung vorhanden sein, wird geschaut, ob diese Sitzung noch aktiv ist – der User also in der letzten Zeit Seiten aufgerufen hat –, da es möglich ist, dass bei der vorherigen Anmeldung vergessen wurde, sich auszuloggen, sodass die Kennung nicht gelöscht wurde.
  ```

Ist auch dieser Test bestanden, steht dem Einloggen nichts mehr im Wege. Zunächst wird eine eindeutige ID für die Sitzung generiert und die ID-Nummer des Datensatzes einer Variablen zugewiesen.

```
$sitzung=uniqid('');
$IDuser=mysql_result($result,0,'ID');
```

Anschließend werden die aktuelle Zeit und die Sitzungs-ID in die Datenbank eingetragen und dann als Cookie gespeichert.

```
$sql="UPDATE $tabellenname SET ";
$sql.=" zeit= now(), ";
$sql.=" kennung='$sitzung' ";
$sql.=" WHERE ID='$IDuser' ";
$test=mysql_query($sql,$link);
setcookie('sit',$sitzung);
setcookie('IDuser',$IDuser);
$meldung="Sie wurden eingeloggt als $username";
unset($username);
```

Alle schließenden Klammern der vorhergehenden Tests.

```
}}}
```

Wenn kein Datensatz mit dem Benutzernamen gefunden werden konnte, wird eine Meldung erzeugt.

```
}else{$meldung.="Bitte kontrollieren Sie Ihren Benutzernamen";}
```

Schließen der Klammer für `if($sent==1)`.

```
}
```

Das Formular zum Einloggen wird ausgegeben.

```
echo "<html><head><title>LOGIN</title></head><body><div align=center>";
echo "<h2><font color=red>$meldung</font></h2>";
echo "<h2>Bitte die Benutzerinformationen eingeben</h2>";
echo "<form action='$PHP_SELF' method='post'>";
echo "<input type=hidden name=sent value=1>";
echo "<p>Benutzername</p>";
echo "<input type=text name=username value='$username'>";
echo "<p>Kennwort</p>";
echo "<input type=password name=kwort>";
echo "<br><br><input type=submit>";
echo "</form>";
?>
</div></body></html>
```

Listing 14.4: Das Script zum Einloggen

Das Script zum Testen der Zugriffsrechte

Für die Überprüfung, ob ein User das Recht hat, eine Seite aufzurufen, definieren wir eine Funktion, sodass es reicht, die Funktion in die Datei *connect.php* zu schreiben. Anschließend können Sie die Funktion in jeder Datei, in der Sie *connect.php* eingebunden haben, aufrufen.

Wenn ein User kein Recht auf den Zugriff hat, wird die Abarbeitung des Scripts mit `die()` beendet und es werden je nach Grund des Abbruchs unterschiedliche Meldungen ausgegeben.

Als Parameter erwartet die Funktion `recht()` einen Wert, der anzeigt, ob die zu schützende Seite von allen registrierten Usern angesehen werden darf oder nur von Administratoren. Für Administratoren muss dieser Wert 1 sein, ansonsten ist er beliebig, z.B. Null, aber nicht leer.

Auch für dieses Script werden die Erklärungen direkt in den Code integriert.

```
function recht($admin)
{
```

Der Funktion wird der Name `recht` gegeben und der übergebene Wert der Variablen `$admin` zugewiesen.

```
Global $link,$IDuser,$sit,$idle;
$pfad="http://www.ihredomain.de/login.php";
$tabellenname='user';
```

Die benötigten Variablen werden als *global* gekennzeichnet. Zwei lokale Variablen werden definiert, um Tipparbeit zu sparen und die Anpassung zu vereinfachen. Zum einen ist dies der absolute Pfad zur Login-Webseite und zum anderen der Tabellenname, in dem die User verwaltet werden.

Wenn einer der beiden Cookies nicht gesetzt ist, wird der Fehler mit der Nummer 1 angezeigt. Die entsprechenden Fehlermeldungen werden später ausgegeben. Sind diese Cookies gesetzt, wird mit den Tests fortgefahren.

```
if(!$IDuser OR !$sit){$fehler=1;}
else{
```

Wenn keine Datenbankverbindung vorliegt, wird der Fehler 2 angezeigt, ansonsten folgen weitere Tests.

```
if(!$link){$fehler=2;}else{
```

Die benötigten Informationen aus der Tabelle der User werden ausgelesen.

```
$sql="SELECT admin, zeit, kennung, (UNIX_TIMESTAMP()-UNIX_TIMESTAMP(zeit)) as
time FROM $tabellenname WHERE ID=$IDuser AND kennung='$sit'";
$result=mysql_query($sql,$link);
```

Mit `(UNIX_TIMESTAMP()-UNIX_TIMESTAMP(zeit)) as time` wird wie beim Login die seit dem letzten Aufruf einer geschützten Webseite vergangene Zeit ermittelt. Über die where-Bedingung gewährleisten wir, dass nur dann ein Datensatz gefunden wird, wenn die im Cookie `IDuser` gespeicherte ID-Nummer des Datensatzes und die im Cookie `sit` abgelegte Sitzungs-ID in einem Datensatz der Tabelle gefunden werden. Somit ist sichergestellt, dass es sich, wenn ein Datensatz gefunden wird, um einen registrierten User mit der aktuellen Sitzungs-ID handelt.

```
if(mysql_num_rows($result)==1){
```

Wenn kein Datensatz vorhanden ist, ist der User nicht berechtigt, die Seite anzuschauen und deswegen weisen wir weiter unten im else-Zweig eine Fehlermeldung an. Wird hingegen ein Datensatz gefunden, kann mit den weiteren Tests fortgefahren werden.

```
if($admin==1 AND mysql_result($result,0,'admin')!=1){$fehler=1;}
```

Wenn die Seite den Administratoren-Status erfordert, der User diesen aber nicht inne hat, wird Fehler 1 angewiesen.

```
if(mysql_result($result,0,'time')/60>$idle){$fehler=3;}
```

Falls der User zu lange inaktiv war, der letzte Seitenaufruf einer geschützten Seite also länger als in `$idle` festgelegt zurückliegt, wird Fehler 3 bestellt.

```
}//mysql_num_rows($result)==1
else{$fehler=1;}
```

Wurde kein Datensatz zurückgegeben, wird im else-Zweig die Fehlermeldung 1 geordert. Anschließend müssen die schließenden Klammer der obigen geschachtelten else-Zweige geschrieben werden.

```
}//else !$link
}//else !$IDuser OR !$sit
```

Wenn ein Fehler gefunden wurde, werden die entsprechenden Meldungen in der Switch-Anweisung je nach Fehlernummer ausgegeben, wobei mit `die()` das Anzeigen der Seite verhindert wird.

```
if($fehler)
{
switch($fehler)
{
case 1:
die("<html><body><div align=center><h1>Sie haben keine Zugriffsrechte auf diese
Seite</h1><a href='$pfad'>Zum LOG IN</a></div></body></html>");
break;
case 2:
die("<html><body><div align=center><h1>Ein Datenbankfehler  verhindert die
Authentifizierung, bitte versuchen Sie es sp&auml;ter noch einmal.</h1></div></
body></html>");
```

```
break;
case 3:
die("<html><body><div align=center><h1>Ihre SitzunggsID ist aufgrund zu langer
inaktivität nicht mehr gültig, bitte melden Sie sich erneut an</h1><a
href='$pfad'>Zum LOG IN</a></div></body></html>");
break;
}
}
```

Waren alle Tests erfolgreich, so liegt kein Fehler vor und die Zeit des letzten Zugriffs auf eine geschützte Webseite kann im Feld ZEIT der Tabelle aktualisiert werden.

```
else
{
$sql="UPDATE $tabellenname SET ";
$sql.=" zeit= now() ";
$sql.=" WHERE ID='$IDuser' ";
mysql_query($sql,$link);
```

Weiterhin wird der Rückgabewert der Funktion ermittelt und mit `return` zurückgegeben. Der Rückgabewert ist, wenn der User Administratorenrechte hat `true`, ansonsten `false`. Lesen Sie dazu den folgenden Exkurs.

```
if(mysql_result($result,0,'admin')==1){return true;}
else{return false;}
}
mysql_free_result($result);
}//Ende Funktion
```

Listing 14.5: So testen Sie die Zugriffsrechte.

Administrator oder kein Administrator

Wenn ein User Zugriffsrechte auf die Seite hat, gibt die Funktion `recht()` TRUE zurück, sofern der User ein Administrator ist, ansonsten FALSE.

Dieser Rückgabewert bedeutet nicht, dass ein User die Seite anschauen darf oder nicht. Wenn ein User die Seite nicht aufrufen darf, wird die Abarbeitung des Scripts mit `die()` in der Funktion `recht()` beendet. Dieser Rückgabewert kennzeichnet, ob der berechtigte User ein Administrator ist oder nicht.

Weisen Sie diesen Rückgabewert mit dem Aufruf der Funktion einer Variablen zu, können Sie eine Webseite mit zusätzlichen Elementen für Administratoren füllen, die der normale User nicht zu Gesicht bekommt oder umgekehrt.

Der Aufruf der Funktion wäre dann `$admin=recht(0);` für eine Seite, die auch Nicht-Administratoren sehen dürfen. Bei anderen Seiten macht die Unterscheidung keinen Sinn.

Eine Textausgabe nur für Administratoren blenden Sie folgendermaßen ein:

```
if($admin){echo "Hallo Administrator";}
```

Entsprechend eine Meldung für normale User:

```
if(!$admin){echo "Hallo normaler User";}
```

Die Seite für das Logout

Wenn Sie einen Zugriffsschutz auf Userbasis haben, müssen die User sich nicht nur einloggen, sondern auch wieder ausloggen können. Also kommen Sie um eine Webseite zum Abmelden nicht umhin. Dieses Script speichern wir im Beispiel in der Datei *logout.php*.

Bild 14.9: Die Webseite zum Abmelden, wenn Sie angemeldet sind

Das Sript für den Logout, erneut mit den integrierten Erkärungen:

```
<?php
require('connect.php');
recht(0);
$tabellenname='user';
```

Die Verbindung zur Datenbank wird hergestellt und die Funktion `recht()`, die in *connect.php* gespeichert ist, zum Test des Zugriffsrechts aufgerufen. Der Aufruf an dieser Stelle hat den Vorteil, dass nicht registrierte oder bereits inaktive User das Script nicht aufrufen können, sodass im Folgenden die Tests einfacher ausfallen können. Weiterhin wird der Name der User-Tabelle in eine Variable geschrieben.

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

Wenn der User das weiter unten erstellte Formular, mit dem er bestätigt, dass er sich abmelden möchte, abgeschickt hat, ist `$sent==1`.

Das Abmeldeprozedere beginnt damit, dass die Sitzungs-ID aus dem Feld KENNUNG des Datensatzes des Users gelöscht und die Zeit, die seit dem letzten Zugriff auf eine geschützte Seite vergangen ist, im Feld ZEIT auf `0` gesetzt wird.

```
$sql="UPDATE $tabellenname SET ";
$sql.=" zeit= 0, ";
$sql.=" kennung='' ";
$sql.=" WHERE ID='$IDuser' ";
mysql_query($sql,$link);
```

Anschließend werden die Cookies gelöscht, indem das Verfallsdatum in die Vergangenheit gesetzt wird.

```
setcookie('sit', '', time()-1000);
setcookie('IDuser', $IDuser, time()-1000);
```

`$IDuser` wird gelöscht, damit nach der Abmeldung nicht das Formular angezeigt wird, sondern die Meldung, dass die Abmeldung erfolgreich war.

```
unset($IDuser);
}
```

Erst jetzt wird der Kopf der Seite ausgegeben, da vor dem Befehl zum Setzen eines Cookies keine Ausgabe an den Browser erfolgen darf.

```
echo "<html><head><title>LOGOUT</title></head><body><div align=center>";
```

Wenn `$IDuser` gesetzt ist, wird das Formular zur Bestätigung, dass der User sich wirklich ausloggen möchte, angezeigt.

```
if($IDuser)
{
echo "<h2><font color=red>$meldung</font></h2>";
echo "<h2>Sie sind angemeldet als ".benutzer()."<br>";
echo "Abmelden?</h2>";
echo "<form action='$PHP_SELF' method='post'>";
echo "<input type=hidden name=sent value=1>";
echo "<br><br><input type=submit>";
echo "</form>";
}
```

Ist `$IDuser` nicht mehr gesetzt, was nach dem erfolgreichen Abmelden der Fall ist, wird eine Erfolgsmeldung ausgegeben.

```
else
{
echo "Sie sind abgemeldet";
}
```

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

Listing 14.6: Das Script für den Logout

Den Zugriffsschutz einsetzen

Nachdem Sie die bis dato erklärten Dateien und Tabellen angelegt haben, sind nun noch die folgenden vorbereitenden Schritte notwendig, um den Zugriffsschutz in Ihre Webseite einzubauen:

- Kopieren Sie die Dateien für den Login und Logout in die oberste Verzeichnishierarchie Ihres Webprojektes. Es muss die oberste Hierarchie sein, damit die Cookies, die beim Anmelden gesetzt werden, für alle untergeordneten Verzeichnisse gelten.
- Die Dateien *connect.php* mit den Anmeldeinformationen und den Funktionen `recht()` und `benutzer()`, und die Datei *useradmin.php* für die Verwaltung der User können Sie in ein beliebiges Verzeichnis legen.
- Passen Sie anschließend alle Pfade und Datenbankangaben an. Die Datenbankangaben finden Sie in der Datei *connect.php*. Alle Pfadangaben, die Sie anpassen müssen, haben wir in den Beschreibungen der Scripte erwähnt. Mit der Variablen `$idle` in *connect.php* setzen Sie die Leerlaufzeit, die zwischen dem Aufruf zweier geschützter Seiten maximal verstreichen darf, ohne dass der User aufgefordert wird, sich erneut anzumelden.
- Erstellen Sie anschließend mit der Datei *useradmin.php* einen User-Account für sich und weisen diesem User den Status eines Administrators zu. Dieser Schritt ist notwendig, da wir vermuten, dass Sie die Seite *useradmin.php* mit Zugriffsschutz versehen möchten. Haben Sie zuvor noch keinen User angelegt, der Administratorrechte hat, können Sie die Seite anschließend nicht mehr aufrufen.

Nach diesen Vorbereitungen können Sie sich um den Schutz der einzelnen Seiten kümmern. Fügen Sie an oberster Stelle der Dateien den folgenden Code ein. Dieser Code muss ganz oben stehen, damit das Anzeigen der Seiten bei nicht vorhandenem Zugriffsrecht mit der Funktion `recht()` verhindert werden kann. Passen Sie den Pfad zur Datei *connect.php* an. Denken Sie daran, dass diese Befehle im PHP-Bereich aufgerufen werden müssen. Umschließen Sie sie gegebenenfalls mit `<php ... ?>`.

Für Administratorwebseiten fügen Sie ein:

```
require('connect.php');
$recht=recht(1);
```

Für sonstige geschützte Seiten fügen Sie ein:

```
require('connect.php');
$recht=recht(0);
```

Kapitel 15
Tipps und Tricks

In diesem Kapitel werden wir einige Themen behandeln, die in den vorhergehenden Kapiteln noch nicht zur Sprache gekommen sind, die Ihnen aber das Arbeiten mit PHP erleichtern können.

Zunächst diskutieren wir einige typische Fehlermeldungen, die mitunter kaum auf die Fehlerquelle hinweisen und die Fehlersuche deswegen besonders aufwändig machen.

Anschließend stellen wir in knapper Form einige nützliche Funktionen von PHP vor, die bisher nicht aufgenommen werden konnten, da sie zu den in den Kapiteln durchgespielten Beispielen nicht gepasst hätten (oder die Beispiele zu umfangreich gemacht hätten!).

Tipps zum Finden von Fehlern

Mit den nachfolgenden Fehlermeldungen werden Sie sicherlich mehr als einmal in Ihrem Leben als Programmierer konfrontiert. Wenn Sie allerdings das erste Mal über diese Meldungen stolpern, sagen die Meldungen wenig über die eigentliche Fehlerquelle aus. Die hier angeführten Beispiele decken nicht alle Fehlermeldungen von PHP ab und sie sind – zugegebenermaßen – auch nicht zwingend die Ursachen für die Fehlermeldungen. Allerdings sind sie relativ häufig der Grund für eine der wenig erfreulichen, aber kaum zu vermeidenden Error-Meldungen.

Fehlende Semikolon

```
Parse error: parse error, expecting `','' or `';'' in /homepages/htdocs/
testbuch/database/bearbeiten.php on line 104
```

Diese Fehlermeldung wird Ihnen sicherlich häufig begegnen; sie weist u.a. auf ein fehlendes Semikolon hin. Allerdings ist die Zeilenangabe oft um eins zu hoch, schauen Sie einfach ein Zeile nach oben, um zu entdecken, wo das Semikolon fehlt. In dem Bild 15.1 sehen Sie das Script, das die Fehlermeldung produziert hat.

Geschwungene Klammern um Anweisungsblöcke

Die nachfolgenden Fehlermeldungen sind fast identisch, nur die Zeilennummern unterscheiden sich. Anhand der Zeilennummern können Sie aber auf zwei unterschiedliche Fehlerursachen schließen.

```
98  echo "</td><td align=right>";
99  echo "<a href='$PHP_SELF?sent=3&ID=";
100 echo mysql_result($result,$i, 'ID');
101 echo "'>Bearbeiten</a>";
102 echo "</td></tr>";
103 echo "</table>"
104 }
105 }
106 else
107 {
108 echo "<h2>Es liegen keine Eintr&auml;ge in der Datenbank vor</h2>";
109 }
110 ?>
```

Bild 15.1: In der Zeile 104 kann kein Semikolon fehlen, aber eine Zeile zuvor.

Fehlende schließende Klammer oder eine öffnende Klammer zuviel

```
Parse error: parse error in /homepages/htdocs/testbuch/database/bearbeiten.php
on line 112
```

Diese Fehlermeldung ist besonders garstig, da die letzte Zeile als Hinweis mit angegeben wird. Im Beispiel ist dies tatsächlich die letzte Zeile (112) im Script. Dort werden Sie voraussichtlich keinen Fehler finden; statt dessen dürfen Sie sich auf die Suche durch das ganze Script begeben, um eine fehlende schließende Klammer eines Anweisungsblocks ausfindig zu machen.

Eine schließende Klammer zuviel

```
Parse error: parse error in /homepages/htdocs/testbuch/database/bearbeiten.php
on line 67
```

Diese Meldung weist häufig darauf hin, dass es im Script eine überflüssige schließende Klammer eines Anweisungsblocks gibt. Vorausgesetzt, die Zeilennummer ist nicht die letzte Zeile (wie im obigen Beispiel), finden Sie die Klammer, über die PHP gestolpert ist, in der genannten Zeile (67), aber bei verschachtelten Strukturen muss es wiederum nicht die schließende Klammer in dieser Zeile sein, die für die Fehlermeldung verantwortlich ist. Dann handelt es sich lediglich um die Zeile, in der die Klammersetzung auf jeden Fall falsch ist. Die überflüssige schließende Klammer kann auch zuvor im Script stehen. PHP kann diese falsch gesetzte schließende Klammer aber nicht genauer identifizieren, da durch unterschiedliche Anordnung der Klammern bewusst logisch verschiedene Blockbildungen veranlasst werden.

Unterschiedliche Programmabläufe durch differierende Klammersetzung

Zwei logisch unterschiedliche Programmabläufe des gleichen Scripts, in dem nur die Klammersetzung verändert wurde, finden Sie in den folgenden beiden Listings. Diese beiden Scripte sind übrigens fehlerfrei, d.h. Sie geben keine Fehlermeldungen aus, aber eines der beiden Scripte macht nicht das, was der Programmierer beabsichtigt hat.

```
<?
$test=-2;
if($test>=0)
{
echo " Test ist positiv oder null ";
}
if($test==0)
{
echo "Test ist 0";
}
else
{
echo " Test ist negativ ";
}
?>
```

Listing 15.1: Beispiel 1 für unterschiedliche Programmabläufe durch Ändern der Klammersetzung der Anweisungsblöcke

```
<?
$test=-2;
if($test>=0)
{
echo " Test ist positiv oder null";

if($test==0)
{
echo " Test ist 0";
}}
else
{
echo "Test ist negativ";
}
?>
```

Listing 15.2: Beispiel 2 für unterschiedliche Programmabläufe durch Ändern der Klammersetzung der Anweisungsblöcke

Im ersten Beispiel (Listing 15.1) wird korrekt ausgegeben:

```
Test ist negativ
```

Auch im zweiten Beispiel (Listing 15.2) wird korrekt ausgegeben:

```
Test ist negativ
```

Wenn Sie jetzt `$test` eine positive Zahl zuweisen z.B. `$test=2`, werden Sie feststellen, dass das zweite Beispiel richtig arbeitet und Folgendes ausgibt:

```
Test ist positiv oder null
```

während das erste Beispiel sich nicht zu einer klaren Aussage durchringen konnte.

```
Test ist positiv oder null
Test ist negativ
```

Unbekannte Funktionen

```
Fatal error: Call to undefined function: benutzer() in /homepages/htdocs/
testbuch/database/bearbeiten.php on line 110
```

Diese Fehlerausgabe weist auf eine nicht bekannte Funktion `benutzer()` in Ihrem Script hin. Prüfen Sie zunächst die Schreibweise des Funktionsnamens im Funktionsaufruf.

Ein häufig auftretendes Problem ist, dass man vergessen hat, die Datei, in der die Funktion definiert wird, mit `include()` oder `require()` einzubinden. Prüfen Sie auch diese Fehlerquelle.

Datenbankfehler

```
Warning: Supplied argument is not a valid MySQL result resource in /homepages/
htdocs/testbuch/database/bearbeiten.php on line 69
```

Diese Fehlermeldung kann Ihnen über den Weg laufen, wenn keine Verbindung zur Datenbank hergestellt wurde oder werden konnte. In Zeile 69 steht z.B. folgende Scriptzeile, die zwar einwandfrei ist, aber die Datenbankverbindung benötigt:

```
$result=mysql_query($sql, $link);
```

Auch die nachfolgend gezeigte Zeile wird einen derartigen Fehler erzeugen, da auf das Ergebnis einer Datenbankabfrage zugegriffen wird, die nicht durchgeführt werden konnte:

```
$anzahl=mysql_num_rows($result);
```

Leider ist die Ursache dieser Fehlermeldung nicht leicht einzukreisen. Haben Sie zuvor einen `SELECT`-SQL-String an die Datenbank gesendet, kann auch dieser fehlerhaft sein. Da es aufwändig ist, die MySQL-Fehler in PHP genauer ausgeben zu lassen, verwenden Sie einfach folgendes Verfahren:

- Lassen Sie sich den SQL-String mit einem `echo`-Befehl im Browser ausgeben.
- Kopieren Sie den im Browser ausgegebenen SQL-String in die Zwischenablage.
- Fügen Sie den Inhalt der Zwischenablage in *phpMyAdmin* in das Formularfeld ein, in dem Sie SQL-Statements an die Datenbank schicken können.
- Anschließend erhalten Sie einen genaueren Hinweis auf die Fehlerquelle.

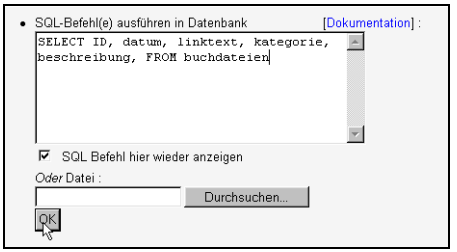

Bild 15.2: Den kopierten SQL-String mit phpMyAdmin an die Datenbank schicken

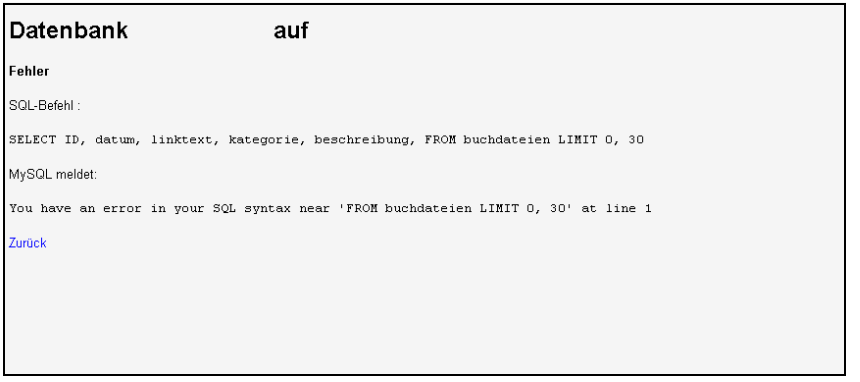

Bild 15.3: Die Fehlermeldung, die MySQL zurückgibt. Richtig: Der Fehler ist in der Nähe von FROM: Nach »beschreibung« darf kein Komma folgen.

Testen eines Kennworts

Ein neues Kennwort können Sie auf erlaubte Zeichen und die erlaubte Länge mithilfe eines regulären Ausdrucks prüfen. Die Arbeit mit regulären Ausdrücken kann etwas verzwickt sein, und deswegen beginnen wir mit einem relativ einfachen Ausdruck, der leicht nachvollziehbar ist. Der hier gezeigte Test beschränkt das Kennwort auf eine Länge von 6 bis 12 Zeichen. Es sind nur Buchstaben und Ziffern zulässig. Das gezeigte Scripfragment (Listing 15.3) prüft ein Kennwort, das in der Variablen `$kwort` vorliegt, z.B. von einem gleichnamigen Formularfeld übergeben wird.

```
<?php
echo "Das neue Kennwort $kwort<br>";
$muster"^[a-zA-Z0-9]{6,12}$";
if(ereg($muster, $kwort))
{
echo "Das Kennwort ist in Ordnung";
}
```

```
else
{
echo "Geben Sie ein korrekte Kennwort ein";
}
?>
```

Listing 15.3: So können Sie testen, ob ein Kennwort richtig ist.

Erklärung des regulären Ausdrucks

In der eckigen Klammer werden die erlaubten Zeichen bestimmt (`[a-zA-Z0-9]`). Die geschweifte Klammer zeigt an, dass diese erlaubten Zeichen mindestens 6 Mal, maximal 12 Mal vorkommen müssen (`{6,12}`). Das Dach (`^`) besagt, dass vor dem Ausdruck keine weiteren Zeichen vorhanden sein dürfen und das $-Zeichen, dass nach dem Ausdruck keine Zeichen folgen dürfen. Somit ist der gesamte Ausdruck auf die 6 bis 12 Zeichen beschränkt (`^[a-zA-Z0-9]{6,12}$`).

Testen einer E-Mail-Adresse – aber richtig

Ein häufiger Einsatz von regulären Ausdrücken ist der Test einer E-Mail-Adresse, die in einem Formular angegeben wurde. In vielen Programmierbüchern, Webseiten oder Foren, die sich diesem Thema widmen, finden Sie lediglich Tests, die auf das @-Zeichen achten. Mitunter wird auch noch geprüft, ob zwei Punkte aufeinander folgen, aber ein umfassenderer Test ist uns bisher nicht begegnet – was nicht heißen soll, dass es keine besseren Tests gibt, aber wir haben bisher keine entdeckt. Der reguläre Ausdruck (Muster), der annähernd alle syntaktischen Regeln einer E-Mail-Adresse erfasst, ist ziemlich kompliziert, und kann in der Regel in entsprechenden Büchern oder Webseiten nicht hinreichend erklärt werden. Wir möchten uns und Ihnen an dieser Stelle diesen Platz gönnen.

Das Script zum Testen

Mit einem Muster lässt sich hervorragend überprüfen, inwieweit eine E-Mail-Adresse korrekt in ein Formularfeld eingegeben wurde. Je nach dem, ob eine Übereinstimmung gefunden wird oder nicht, können Sie dann eine entsprechende Meldung ausgeben. Die Funktion `ereg()` wird dabei als Prüfbedingung eines if-Befehls verwendet. Vorausgesetzt, es gibt ein Script, in dem ein Formularfeld zur Eingabe einer E-Mail den Namen `email` hat (der dann in PHP als Variable genommen wird), sieht das dann so aus:

```
<?php

echo "Ihre Nachricht $message <br>";
$muster"^[_a-zA-Z0-9-](\.{0,1}[_a-zA-Z0-9-])*@([a-zA-Z0-9-]{2,}\.){0,}[a-zA-Z0-
9-]{3,}(\.[a-zA-Z]{2,4}){1,2}$";
if(ereg($muster, $email))
{
echo "danke";
```

```
}
else
{
echo "geben Sie eine korrekte E-Mail-Adresse ein";
}

?>
```

Listing 15.4: Überprüfung der korrekten Eingabe einer E-Mail-Adresse mit einem regulären Ausdruck.

Erklärung des Musters

Das Muster ist etwas länger geworden, da – wie gesagt – die Aufgabe, eine E-Mail-Adresse zu testen, nicht unbedingt trivial ist. Auch mit diesem umfangreichen und relativ genauen Test sind Sie aber nicht vor ungültigen E-Mail-Adressen geschützt, da z.B. mit regulären Ausdrücken nicht getestet werden kann, ob eine syntaktisch richtig eingegebene Domain wirklich existiert. Auch wird die Eingabe nicht auf die beschränkte Anzahl von Top-Level-Domains getestet, sondern es werden nur die Anzahl und zulässigen Zeichen der Top-Level-Domains untersucht. Es ist sogar unmöglich zu testen, ob der Mail-Name vor dem @-Zeichen existiert. Während Sie Domain-Namen und Top-Level-Domains mit aufwändigen Scripten testen könnten – also die Existenz bei den entsprechenden Servern, die die Domain-Namen verwalten, abfragen können – gibt es keine Auskunftsquelle für die Existenz eines Mail-Namens innerhalb einer Domain.

Wir versuchen, das Muster in einzelnen Teilen zu erklären. Beginnen werden wir mit dem Mail-Namen vor dem @-Zeichen (`^[_a-zA-Z0-9-](\.{0,1}[_a-zA-Z0-9-])*@`).

Der Mailname darf Buchstaben, Ziffern, Unterstriche (_), Punkte (.) und Bindestriche (-) enthalten. Die möglichen Zeichen werden mit [_a-zA-Z0-9-] auf diese erlaubte Zeichenmenge begrenzt. Der Punkt ist zunächst ausgelassen, da er eine Sonderrolle spielt. Dieser Ausdruck ([_a-zA-Z0-9-]) wird in dem Muster mehrfach eingesetzt.

Mit `^[_a-zA-Z0-9-]` wird erreicht, dass der Mail-Name mit einem zulässigen Zeichen, aber nicht mit einem Punkt anfängt. Eines der erlaubten Zeichen ist erforderlich. Das »Dach« (^) zeigt an, dass dieses Zeichen am Anfang zu finden sein muss. Ein kurzer Mail-Name kann an dieser Stelle schon zu Ende sein, so dass alle folgenden Zeichen vor dem @-Zeichen optional sind. Der Test der nachfolgenden optionalen Zeichen erfolgt in der runden Klammer. Mit dem `*`-Zeichen am Ende der Klammer wird angezeigt, dass die durch die Tests in der Klammer zugelassenen Zeichen beliebig häufig auftreten dürfen – also auch keinmal.

Die runde Klammer `(\.{0,1}[_a-zA-Z0-9-])` erlaubt einen oder keinen Punkt (`\.{0,1}`) gefolgt von einem erlaubten Zeichen. Zulässig ist z.B. »G« oder ».G« aber nicht »..G« oder ».« (nur ein Punkt). In Kombination mit dem folgenden Sternchen können sich diese Angaben beliebig häufig wiederholen, sodass sichergestellt ist, dass nach dem Punkt immer ein erlaubtes Zeichen erscheint. Das @-Zeichen kann somit nicht auf einen Punkt folgen. Entnehmen Sie zur Veranschaulichung ein paar Beispiele der Tabelle 15.1.

Zeichenkette	Erlaubt	Erklärung
g	JA	`[_a-zA-Z0-9-]` lässt das Zeichen zu, der Punkt ist optional
.	NEIN	`\.{0,1}` lässt den Punkt zu, aber mit `[_a-zA-Z0-9-]` wird ein weiteres Zeichen zwingend gefordert
.t	JA	`\.{0,1}` lässt den Punkt zu, und `[_a-zA-Z0-9-]` das weitere Zeichen
..t	NEIN	`\.{0,1}` lässt nur einen Punkt zu – keine zwei Punkte – und `[_a-zA-Z0-9-]` erlaubt das weitere Zeichen. Auch durch das * wird der Ausdruck nicht zulässig, da ein einzelner Punkt unzulässig ist (s.o.).
gg	JA	`[_a-zA-Z0-9-]` lässt das Zeichen zu, was durch das * zweimal wiederholt wird
.t.g	JA	`\.{0,1}` lässt den Punkt zu, und `[_a-zA-Z0-9-]` das weitere Zeichen. Das * erlaubt die Wiederholung.
.ttt.kkk	JA	.t wird durch `\.{0,1}` und `[_a-zA-Z0-9-]` zugelassen. Bei der Wiederholung durch das * wird zweimal das t (tt) zugelassen und mit .kkk wiederholt sich das Ganze.

Tabelle 15.1: Beispiel für den regulären Ausdruck `(\.{0,1}[_a-zA-Z0-9-])*`

Jetzt wird der Teil `(@([a-zA-Z0-9-]{2,}\.){0,}[a-zA-Z0-9-]{3,}(\.[a-zA-Z]{2,4}){1,2}$)` nach dem @-Zeichen erklärt.

Nach dem @-Zeichen dürfen wieder bis auf den Unterstrich (_) alle oben erlaubten Zeichen verwendet werden. Auch hier spielt der Punkt eine Sonderrolle, sodass die erlaubten Zeichen mit `[a-zA-Z0-9-]` getestet werden können.

Jeder Domain-Name besteht prinzipiell aus drei Elementen: die Subdomain, Domain-Name und Top-Level-Domain. Alle drei Angaben werden durch Punkte getrennt. Die Sache wird leider etwas erschwert, da in der Subdomain wiederum Unterdomains angelegt werden können, die erneut durch Punkte getrennt werden. Auch die Top-Level-Domain sind nicht so einfach gestrickt, wie die bekannten Top-Level-Domains (.com, .de, .info, .gov etc.) es vermuten lassen. Denken Sie z.B. an co.uk, hier ist wieder ein Punkt enthalten. Eine komplizierte E-Mail-Adresse könnte hinter dem @-Zeichen also auch folgendermaßen aussehen:

```
@unterdomain.unterdomain.DomainName.co.uk
```

Um die Sache systematisch anzugehen, beginnen wir die Erklärung diesmal am Ende der Zeichenkette.

Die Top-Level-Domains bestehen nur aus Buchstaben, deren Anzahl zwischen 2 und 4 schwankt. Diesem Sachverhalt tragen Sie mit `[a-zA-Z]{2,4}` Rechnung. Vor der Top-Level-Domain muss ein Punkt sein, deswegen wird `\.` vorangestellt. Um den Sonderfall co.uk oder Ähnliche abzudecken, darf dieser Ausdruck sich zweimal wiederholen – muss aber mindestens einmal vorkommen –, deswegen wird er in runde Klammern gesetzt und mit `{1,2}` die mögliche Wiederholung angezeigt. Das folgende `$`-Zeichen erzwingt, dass diese Angabe am Ende der zu testenden Zeichenkette steht – also keine weiteren Zeichen folgen: `(\.[a-zA-Z]{2,4}){1,2}$`

Der eigentlich Domain-Name muss mindestens drei der erlaubten Zeichen enthalten und steht vor der Top-Level-Domain. Dies wird durch `[a-zA-Z0-9-]{3,}` abgedeckt. Es sind hier beliebig viele der erlaubten Zeichen zulässig, aber es müssen mindestens drei sein.

Die Namen der Subdomains können die gleichen Zeichen enthalten, wie der eigentliche Domainname, allerdings reichen hier zwei Zeichen aus (`[a-zA-Z0-9-]{2,}`). Jeder Subdomainname wird mit einem Punkt separiert, deswegen muss nach den Namen ein Punkt folgen (`[a-zA-Z0-9-]{2,}\.`). Da Subdomainnamen nicht vorgeschrieben sind, sondern eher die ganz seltene Ausnahme darstellen, kann der Name komplett entfallen. Andererseits kann die Hierarchiestruktur theoretisch beliebig tief untergliedert werden, sodass beliebig viele Subdomainnamen zugelassen werden müssen (`([a-zA-Z0-9-]{2,}\.){0,}`).

Wenn Sie nun den »Wald vor lauter Bäumen« nicht mehr sehen (Sie aber neugierig geworden sind), dann testen Sie den regulären Ausdruck. Sie schreiben das folgende kleine Script einfach ab und rufen die Datei im Browser auf. Geben Sie verschiedene E-Mail-Adressen zum Testen ein. Klicken Sie auf die Schaltfläche, um das Formular abzuschicken und das Ergebnis angezeigt zu bekommen.

```
<html><body>

<form action = "<?php echo $PHP_SELF; ?> " method=Post>
<p> bitte die zu testende E-Mail-Adresse eingeben</p>
<input type=Text name=email value="<?php echo $PHP_SELF; ?> ">
<br>
<input type=submit>
<br>
<br>
</form>

<?php $muster = "^[_a-zA-Z0-9-](\.{0,1}[_a-zA-Z0-9-])*@([a-zA-Z0-9-
]{2,}\.){0,}[a-zA-Z0-9-]{3,}(\.[a-zA-Z]{2,4}){1,2}$";
if(ereg($muster, $email)
{
echo "die E-Mail-Adresse ist gültig";
}
else{ echo "die E-Mail-Adresse ist nicht gültig";
}
?>

</body></hmtl>
```

Informationen, die Ihnen PHP und der Webserver flüstern

PHP in Zusammenarbeit mit dem Webserver verrät Ihnen einiges über den Surfer, der Ihre Scripte aufruft sowie über das Script, das gerade ausgeführt wird. Diese Werte stehen in Variablen zur Verfügung. Eine dieser Variablen haben wir in den Beispielen bereits häufig eingesetzt: `$PHP_SELF`. In `$PHP_SELF` steht der Pfad zur aktuellen Datei. Weitere Variablen können Sie der Tabelle 15.2 entnehmen.

Variable	Beschreibung des Werts
$HTTP_USER_AGENT	Angabe über den vom Client verwendeten Browser
$REQUEST_METHOD	Art der Anfrage GET oder POST
$REMOTE_ADDR	IP-Adresse des Clients
$QUERY_STRING	Bei GET-Anfragen werden die an die URL angehängten Daten in dieser Variablen gespeichert. Sie finden hier alles, was nach dem ? an die URL gehängt wird. Ist die aufgerufene Adresse http://www.brain-and-trust.de/index.php?sent=1&name=egon, steht in $QUERY_STRING der Wert sent=1&name=egon. In diesem Fall ist aber auch in `$sent` der Wert 1 und in `$name` der Wert egon gespeichert. Interessant wird es, wenn Sie einen Aufruf wie den folgenden verarbeiten möchten: http://www.brain-and-trust.de?dasistabersupper. In diesem Fall steht nur in $QUERY_STRING dasistabersupper, Sie finden diesen Wert in keiner weiteren Variablen.
$REQUEST_URI	Die gesamte Anfrage inklusive $QUERY_STRING, z.B. http://www.brain-and-trust.de/index.php?sent=1&name=egon
$HTTP_REFERER	Adresse der Seite, von der die Seite aufgerufen wurde. Auf welcher Seite (z.B. http://www.brain-and-trust.de/index.php) wurde ein Hyperlink angeklickt, um auf diese Seite zu kommen. Dieser Wert wird vom Browser gesendet und wird leider nicht von allen Browsern gesetzt.

Tabelle 15.2: Eine Auswahl der Variablen, die der Server Ihnen zur Verfügung stellt

Informationen über den Browser des Clients

Mit der Anfrage nach einer Seite sendet der Browser des Clients, wie Sie der Tabelle 15.2 entnehmen können, die Information über den Browser. Mit der Funktion `get_browser()` wird diese Information mit den Einträgen einer Datei verglichen, die alle unterstützten

Funktionen aller Browser enthält. Diese Datei pflegt der Serveradministrator für Sie. Wird der Browsertyp gefunden, gibt `get_browser()` alle unterstützten Funktionen des betreffenden Browsertyps aus. Beachten Sie, dass hierbei nicht berücksichtigt wird/werden kann, welche der Funktionen der Surfer deaktiviert hat. Wenn der Browser in der Liste nicht gefunden werden kann, oder Ihr Serveradministrator die Liste nicht pflegt bzw. den Pfad zu der Liste nicht in der Datei *php.ini* angegeben hat, erhalten Sie keine Rückmeldung (false).

Anhang
Befehlsreferenz

In dieser Referenz werden die in den Kapiteln dieses Buches erwähnten und/oder verwendeten Befehle von PHP aufgelistet und nochmals kurz erläutert. Wir haben uns für eine alphabetische Sortierung entschieden, da es keine komplette Befehlsreferenz ist und die Einordnung in Kategorien daher nicht zweckmäßig erschien. Lesen Sie die Referenz bitte folgendermaßen

```
int[1] fputs[2](int[3] filehandler[4], string[3] datenX[5][, int[3] length[6]]
[1]= Datentyp    [2]= Befehl    [3]= Datentyp    [4]= arg1    [5]= arg2    [6]= arg3
```

Die Abkürzung »int« steht für *Integer*. Die eckige Klammer bedeutet, dass der/die Parameter optional sind. Um den Befehl bzw. die Syntax deutlich zu machen, wird in der Beschreibung oftmals noch einmal die Bezeichnung für das Argument aufgenommen und dann meistens in eine runde Klammern gesetzt. Es heißt also in der Erklärung beispielsweise:

Mit `fputs` werden Daten (datenX) an die aktuelle Position des Dateizeigers in eine Datei geschrieben. Als datenX wurde oben eines der Argumente bezeichnet.

Die Referenz folgt keinem strikten Muster, mitunter finden Sie lediglich eine Kurzbeschreibung, aber da, wo es uns angebracht erschien, auch ein oder zwei kleine Eingabebeispiele oder ein kurzes Scriptfragment.

(string)

string (`string`) $variable

Mit dem Befehl`(string)` wird der Inhalt einer Variablen als Zeichenkette angesehen. Mit dieser Funktion stellen Sie also sicher, dass die Variable ($variable) tatsächlich als Zeichenkette vorliegt. Der Befehl selbst steht in Klammern.

```
$var1= (string) $var2
```

Genauso verfahren Sie mit (double), (integer), (array), (object)

abs

mixed `abs`(mixed zahl)

Mit `abs` kann man sich den Absolutwert einer Zahl (zahl) zurückgeben lassen.

Beispiel:

```
echo abs(9);
echo abs(-10);
```

Ausgabe:

```
9
10
```

addslashes

string `addlashes`(string zeichenkette)

Die Funktion maskiert alle Vorkommen von bestimmen Zeichen in einer Zeichenkette (zeichenkette) mit einem Backslash. Der Einsatz dieser Funktion ist vor allem im Zusammenhang mit Datenbanken relevant. Geschützt werden: einfache Anführungszeichen, doppelte Anführungszeichen, Backslash.

array

array `array`(...)

Mit der Funktion wird ein Array erzeugt. Ein Array ist eine besondere Form einer Variablen, die mehr als einen Wert speichern kann. (Eine ausführliche Erklärung zu Arrays finden Sie im *Kapitel 3, PHP – Die Grundlagen*.)

array_merge

array `array_merge`(array array1, array array2 [, array3 ...])

Der Befehl vereinigt zwei (oder mehr) Arrays zu einem neuen Array. Das neue Array enthält alle Elemente (als Kopien) der beiden ursprünglichen Arrays. Die alten Arrays bleiben unverändert. So erzeugen Sie ein neues Array:

```
$arrayneu = array_merge(arrayalt1, arrayalt2);
```

ceil

int `ceil`(float zahl)

Mit diesem Befehl wird eine Zahl immer aufgerundet, auch eine Zahl wie 1.3. Die Ausgabe wäre: 2.

Siehe auch round(), floor()

checkdate

int `checkdate`(int Monat, int Tag, int Jahr)

Mit `checkdate` lässt sich prüfen, ob eine Datumsangabe korrekt ist. Die Funktion gibt `true` oder `false` zurück.

Eingabebeispiel:

```
$date1 = checkdate(1, 1, 2000,);
if(date1 == true) {
echo "das Datum stimmt";
}
```

chr

string chr(int ascii);

Der Befehl gibt das ASCII-Zeichen zu einer angegebenen Nummer aus. Schreiben Sie beispielsweise:

```
echo chr(120);
```

Sie erhalten als Ausgabe:

```
x
```

closedir

void closedir (int Handle)

Die Funktion schließt ein Verzeichnis.

siehe auch opendir()

copy

int copy(string Quelle, string Ziel)

Mit copy wird eine Quelldatei (Quelle) zu einer Zieldatei (Ziel) kopiert. Im Erfolgsfall gibt die Funktion true zurück.

count

```
int count(mixed arrayvariable)
```

Die Funktion zählt die Menge der Elemente innerhalb eines Arrays und gibt die Anzahl zurück.

Beispiel

```
$testarray=array("Jan", "Susie", "Daniel", "Willie");
echo count($testarray);
```

Ausgabe:

```
4
```

In der for-Schleife werden die Elemente durchlaufen und zurückgegeben:

```
$testarray=array("Jan", "Susie", "Daniel", "Willie");
for($i=0;$i<count($testarray); $i++)
{echo $testarray[$i] . "<br>";
}
```

Ausgabe:

```
Jan
Susie
Daniel
Willie
```

current

mixed `current`(array array)

Die Funktion gibt den Wert des aktuellen Elements aus.

```
array[hosen]=1;
array[hemden]=2;
echo current(array);
next(array);
echo "<br>";
echo current(array);
```

Ausgabe:

```
1
2
```

Siehe auch next(), reset(), prev()

date

string `date`(string format [, int timestamp])

Die Funktion formatiert eine angegebene Zeit bzw. ein angegebenes Datum. Die Zeitangabe wird im Parameter `timestamp` übergeben. Wenn dieser Parameter leer bleibt, nimmt die Funktion die aktuelle Zeit (Datum und Uhrzeit). Mit dem ersten Parameter (format) wird festgelegt, welche Informationen über die Zeitangabe ausgegeben werden sollen. Zahlreiche Platzhalter sind möglich, z.B. »d« für den Tag des Monats.

Ein Beispiel für die aktuelle Zeit

```
echo date("d M Y");
```

Ausgabe: das aktuelle Datum im Format: 10 Oct 2001. (Die komplette Tabelle mit den Platzhaltern finden Sie im Kapitel 3, *PHP – Die Grundlagen*.)

Mit `timestamp` verwenden Sie `time()` und schreiben die Zeit zunächst in eine Variable.

```
$timestamp = time();
```

Soll ein bestimmtes Datum in den *UNIX-timestamp* umgewandelt werden, setzen Sie den Befehl `mktime()` ein und schreiben die Werte in der Reihenfolge

Stunde, Minute, Sekunde, Monat, Tag, Jahr

in die Klammer:

```
$timestamp = mktime(14, 30,20,9,11,2001);
```

Ausgabe:

```
11.9.2001, 14:30:20 Uhr
```

siehe auch mktime(), time()

die

void `die`(string message)

Mit der Funktion lässt sich ein laufendes Script beenden und eine Nachricht (message) an den Browser senden. Der folgende Programmcode wird nicht weiter ausgeführt.

echo

string `echo`(string arg1, string [argn])

Der Befehl gibt alle Strings aus, in der Regel im Browser. Da es sich bei `echo` um ein Sprachkonstrukt handelt, ist die Klammer, die sonst immer geschrieben werden muss, optional.

empty

int `empty` (mixed variable)

Die Funktion gibt `true` zurück, wenn die Variable gar nicht initialisiert wurde, leer ist oder gleich Null.

ereg_replace

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). Bei der Suche wird für suchennach ein regulärer Ausdruck als Suchmuster erwartet.

siehe str_replace()

explode

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.

Beispiel:

```
$reihe="eins*zwei*drei*vier";
$element=explode('*',$reihe);
for($i=0; $i<count($element);$i++)
{echo "Element $i im Array \$element ist: " . $element[$i]."<br>";}
```

Ausgabe:

```
Element 0 im Array $element ist: eins
Element 1 im Array $element ist: zwei
Element 2 im Array $element ist: drei
Element 3 im Array $element ist: vier
```

fgets

sring `fgets`(int file, int length)

Die Funktion liest aus einer Datei eine Zeile mit der Länge (length) aus. Der Dateihandler (file) muss ein gültiger Handle für eine geöffnete Datei sein. Achtung: Ist die Zeile länger als der angegebene Wert, wird der Rest ignoriert.

Beispiel:

```
$file = fopen("eintrag.txt", r);
$file1 = fgets($file, 100);
$close($file);
```

siehe auch fopen(), rewind(), fputs(), close()

file_exists

bool `file_exists`(string filename)

Mit der Funktion lässt sich überprüfen, ob eine bestimmte Datei auf dem Server existiert. Wenn die Datei existiert wird `true` ansonsten `false` zurückgegeben.

filesize

int `filesize`(string dateiname)

Die Funktion gibt die Größe einer Datei in Byte zurück.

Beispiel:

```
$datei = 'zeichen.txt';
$size = filesize($datei);
echo $size;
```

floor

int `floor`(float zahl)

Diese Funktion rundet eine Zahl immer ab, ist also das Gegenstück zu ceil.

siehe auch ceil(), round()

fopen

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

Mit `fopen` wird eine Datei (dateiname) oder URL geöffnet. Sofern sie noch nicht existiert, wird die Datei mit `fopen` je nach modus auch erzeugt.

Der Einsatz sieht folgendermaßen aus:

```
$dateihandler = fopen("Datei.txt", "Modus");
```

Der Modus bestimmt, auf welche Weise die Datei geöffnet wird. Für den Modus werden Abkürzungen benutzt.

Modus	Bedeutung
r	erlaubt das Auslesen einer Datei und positioniert den Dateizeiger auf den Anfang der Datei.
w	schreibt in eine Datei und legt sie an, sofern sie noch nicht existiert. Der Dateizeiger wird auf den Anfang der Datei positioniert.
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. Der Dateizeiger wird auf den Anfang der Datei positioniert.
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.

siehe auch fgets(), fputs(), rewind(), fclose()

foreach

`foreach`(string arrayname, as string tempvar)

Die Funktion dient dazu, ein Array auszulesen, sodass das gesamte Array auf einmal ausgelesen wird. Der Trick: jeder Wert des Arrays (arrayname) wird temporär einer Variablen (tempvar) zugewiesen, der Name wird durch `as` angegeben.

Beispiel:

```
$testarray=array("Jan", "Susie", "Daniel", "Willie");
foreach($testarray as $ausgabe)
{echo $ausgabe . "<br>";}
```

Als Ausgabe erhalten Sie

```
Jan
Susie
Daniel
Willie
```

fputs

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

Mit `fputs` werden Daten (datenX) 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.

Eine Einsatzmöglichkeit kann folgendermaßen aussehen:

```
$filehandler=fopen("kommentare.txt","a");
$ausgabe=$name."~".$betreff."~".$message;
fputs($filehandler,$ausgabe);
fclose($filehandler);
```

In eine Datei namens *kommentare.txt* werden Daten geschrieben. Der Dateizeiger (Cursor) steht (durch `"a"`) am Ende der Datei und hängt die Daten (die von Formularfeldern übertragen werden und in `$ausgabe` gespeichert sind) an. Die Tilde wird im Beispiel als Trennzeichen zwischen den Daten verwendet.

siehe auch fopen(), rewind(), fgets(), fclose()

getimagesize

array `getimagesize`(string filename [, array bildinfo])

Der Befehl gibt Informationen über das Bild aus. Das Ergebnis steckt in einem Array. Man erhält: Breite, Höhe, Grafiktyp.

Der Grafiktyp wird in Form von Kennziffern ausgegeben und zwar:

1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF

```
$info=getimagesize('bild.gif');
echo "Das Bild ist $info[0] Pixel breit<br>";
echo "Das Bild ist $info[1] Pixel hoch<br>";
echo "Das Bild ist vom Typ: $info[2]";
```

Ausgabe:

```
Das Bild ist 100 Pixel breit<br>
Das Bild ist 150 Pixel hoch<br>
Das Bild ist vom Typ: 1
```

gettype

string `gettype`(mixed Variable)

Die Funktion stellt den Datentyp einer Variablen fest und gibt ihn aus.

Beispiel:

```
$einezahl = 11.7;
echo gettype($einezahl);
```

Ausgabe:

```
double
```

header

int `header`(string zeichenkette)

Die Funktion sendet eine Zeile eines HTTP-Headers. `Header` müssen vor allen anderen Angaben jedweder Art erstellt und geschickt werden.

siehe auch setcookie()

imagecolorallocate

int `imagecolorallocate`(int bild, int rot, int grün, int blau)

Mit `imagecolorallocate()` richtet man eine Farbe ein, die für Zeichenoperationen auf einer Fläche (bild) benutzt werden soll. Die gewünschte Farbe wird als RGB-Wert (rot, grün, blau) angegeben. Sie müssen die Funktion für jede Farbe aufrufen, die Sie in der Grafik verwenden möchten. Jeder mit `imagecolorallocate ()` angelegten Farbe eines Bildes (bild) wird eine Nummer (beginnend bei 0) zugewiesen. Die Funktion gibt Ihnen diese Nummer zurück. Weisen Sie diese Nummer einer Variablen zu, um die Farbe in den Zeichenbefehlen verwenden zu können. Die erste erstellte Farbe bildet die Hintergrundfarbe des Bildes (bild). Das folgende Beispiel zeichnet eine rote Diagonale auf weißem Hintergrund.

```
$bild=imagecreate(200,200);
$weiss=imagecolorallocate($bild, 255, 255, 255);
$rot imagecolorallocate($bild, 255, 0, 0);
imageline($bild, 0, 0, 199, 199, $rot);
```

imagecopyresized

int `imagecopyresized`(int zielimage, int quellimage, int zx, int zy, int qx, in qy, int brziel, int hochziel, int brquell, int hochquell)

Mit dem Befehl wird ein Teil eines Bildes in ein anderes Bild kopiert. zx und zy sind die Koordinaten für das Ziel, qx und qy für den Bereich, der kopiert werden soll. Dann werden die Breite und Höhe des Zielbereichs und die Breite und Höhe des Ursprungsbildes angegeben.

imagecreate

int `imagecreate`(int x-size, int y-size)

Mit diesem Befehl erzeugen Sie eine Fläche (Rohbild) für ein neues Bild mit der Breite x-size und der Höhe y-size. Die Funktion gibt einen Zeiger zurück, mit dem danach weitergearbeitet wird. In der Regel wird auch eine Farbe (nach RGB-Schema) zugewiesen.

imagecreatefromgif

int `imagecreatefromgif`(string filename)

Der Befehl liest die Informationen eines vorhandenen Bildes ein, um ein neues Bild im Speicher zu erstellen. Der Befehl funktioniert entsprechend mit den anderen unterstützten Grafikformaten. Er gibt wie `imagecreate()` ein Handle auf das Bild im Speicher zurück.

imagedestroy

int `imagedestroy`(int bild)

Mit dem Befehl wird der Speicher gelöscht, der durch das Bild (bild) besetzt wurde

imagefill

int `imagefill`(int bild, int koordinateX, int koordinateY, int farbe)

Der Befehl füllt eine Arbeitsfläche (bild) mit einer Farbe. Die Füllung beginnt ab dem Koordinatenpunkt. Die Farbe wird soweit ausgedehnt, bis `imagefill` auf eine andere Farbe als die des Startpunktes trifft. Die verwendete Farbe muss zuvor mit `imagecoloral-locate` erstellt worden sein.

imagefilledpolygon

int `imagefilledpolygon`(int bild, array Punkte, int anzahlPunkte, int farbe)

Hiermit kann man ein Vieleck zeichnen, wobei die Eckpunkte in einem Array (Punkte) abgelegt werden. In anzahlPunkte legt man die Anzahl der Eckpunkte fest. Jeder Punkt belegt zwei aufeinanderfolgende Elemente des Arrays (Punkte): das erste für die X-Koordinate, das zweite Element für die Y-Koordinate.

imagefontheight

int `imagefontheight`(int font)/int imagefontwidth(int font)

Die Befehle ermitteln, welche Höhe bzw. Breite der Text in der angegebenen Schriftart benötigt. Der Wert wird in Pixeln zurückgegeben.

imagegif

int `imagegif`(int bild[, string filename])

Mit diesem Befehl wird ein Bild (bild), das zuvor erstellt wurde an den Browser gesendet, und zwar im GIF-Format. Mit dem optionalen Parameter wird das Bild in eine Datei geschrieben. Der Befehl funktioniert entsprechend mit den anderen unterstützen Grafikformaten.

imageline

int `imageline`(int bild, int koordinateX1, int koordinateY1, int koordinateX2, int koordinateY2, int farbe)

Der Befehl zeichnet eine Linie auf dem Rohbild (bild) zwischen die angegebenen Koordinaten mit der angegebenen Farbe (farbe).

imagerectangle

int imagerectangle(int bild, int eckpunktX1, int eckpunktY1, int eckpunkt X2, int eckpunktY2, int farbe)

Mit dem Befehl lässt sich auf einer Zeichenfläche (bild) ein Rechteck mit den angegebenen Eckpunkten erstellen. eckpunktX1/eckpunktY1 sind die linke obere Ecke, entsprechend eckpunktX2/eckpunktY2 die rechte untere Ecke. Das Rechteck wird nicht ausgefüllt. Die Rahmenfarbe legen Sie mit farbe fest.

imagestring

int `imagestring`(int bild, int font, int koordinateX, int koordinateY, string text, int farbe)

Auf diese Weise zeichnen Sie eine Beschriftung (text) auf das Bild. Sie geben den Startpunkt (koordinateX, koordinateY), die Schriftgröße (font) und die Farbe an. Als Werte für die Schriftgröße können Sie 1, 2, 3, 4, 5 verwenden. Bei den Koordinaten des Startpunkts handelt es sich um die linke obere Ecke des Textes.

include

void `include`(string dateiname)

Die Funktion liest die angegebene Datei ein und wertet sie aus.

is_file

bool `is_file`(string dateiname)

Mit dieser Funktion können Sie testen, ob eine Datei (dateiname) existiert und ob es sich um eine Datei handelt.

Beispiel:

```
$datei = "spruch1.txt";
if(is_file($datei)){
echo "klar, das ist eine Datei";
}
```

is_uploaded_file

bool `is_uploaded_file`(string tempdateiname)

Mit der Funktion können Sie überprüfen, ob eine Datei (tempdateiname) per `Post`-Methode hochgeladen wurde. In tempdateiname muss der temporäre Dateiname stehen, und nicht der tatsächliche.

isset

int `isset`(mixed variable)

Die Funktion überprüft, ob eine Variable (variable) oder ein Array existiert.

Beispiel:

```
$name = "Daniel";
if(isset($name)) {
```

```
echo "hallo Daniel";
}
```

ltrim

string `ltrim`(string zeichenkette)

Der Befehl entfernt Leerstellen am Anfang einer Zeichenkette (zeichenkette).

mail

bool `mail`(string Empfänger, string betreff, string nachricht [,zusätzlicher header])

Die Funktion zum Versenden von E-Mails im Text- oder HTML-Format. Im optionalen Parameter kann man mit `from` eine Absenderadresse übergeben.

Eingabebeispiele:

```
mail(an@whoever.de, "testmail", "na, das klappt doch", "from:von@whoever.de");
```

Geschickter ist die Festlegung von Variablen:

```
$empfaenger = "an@whoever.de";
$betreff = "testmail";
$nachricht = "na, das klappt doch";
mail($empfaenger, $betreff, $nachricht, "from: $email");
```

Die Variable `$email` korrrespondiert (beispielsweise) mit dem Namen eines Formularfeldes, in das eine E-Mail-Adresse eingegeben wurde.

md5

string `md5`(string str)

Die Funktion wendet die `md5`-Verschlüsselung auf str an und gibt den verschlüsselten Text zurück.

Beispiel:

```
echo md5('test');
```

Ausgabe:

```
098f6bcd4621d373cade4e832627b4f6
```

mkdir

int `mkdir`(string Pfadname, int modus)

Mit `mkdir` wird ein Verzeichnis angelegt. Im Parameter Modus wird festgelegt, welche Rechte die verschiedenen Gruppen (Eigentümer, Gruppe, die Welt) erhalten.

Bei *UNIX* kann man über die Rechtevergabe für den Eigentümer/Owner, Gruppe/Group und dem Rest der Welt/Public verschiedene Zugriffsrechte vergeben. Die Vergabe der Rechte bei `mkdir()` erfolgt über eine dreistellige Zahl, der eine Null vorangestellt wird. Die erste Ziffer gibt das Recht des Eigentümers/Owners, die zweite der Gruppe/Group und

die dritte der Welt/Public an. Die Ziffern ermittelt man aus den drei möglichen Rechten Lesen/Schreiben/Ausführen, wobei jedem Recht ein Wert zugeordnet ist, der für jedes gewährte Recht addiert wird.

Lesen/Read = r = 4
Schreiben/Write = w = 2
Ausführen/Execute = x = 1

Rechenbeispiel:	User	Group	World	
drwxr-x-w-	r + w + x	r + – + x	- + w + -	0752
das entspricht	4 + 2 + 1 =7	4 + 0 + 1 =5	0 + 2 + 0 =2	0752

Beispiele:

Gesamt	User/Group/World	Oktal
drwxrwxrwx	rwx/rwx/rwx	0777
drwxr-xr-x	rwx/r-x/r-x	0755
drwx------	rwx/---/---	0700
drwxr-x---	rwx/r-x/---	0750
drwxr-xr--	rwx/r-x/r--	0754

Eingabebeispiel:

```
if(mkdir("bilder",0700)){
echo "Verzeichnis erfolgreich angelegt";
}
```

max

mixed max(mixed argument1, argument2[,…])

Die Funktion ermitteln den höchsten Wert der übergebenen Werte. Zwei Werte sind logischerweise zwingend erforderlich. Sie können die Werte auch in einem Array übergeben.

```
$array=array("3", "5", "7", "12");
echo max($array);
echo max(7,11,9);
```

Ausgabe:

```
12
11
```

siehe auch min()

mysql_close

int `mysql_close`([int Verbindungskennung])

Die Funktion beendet die Verbindung zum Datenbankserver. Auf Webservern wird ansonsten die Verbindung automatisch mit Beendigung des Scripts geschlossen.

mysql_connect

int `mysql_connect`([string hostname[:port][:/path/to/socket][, string Benutzername][, string Kennwort]]])

Die Funktion stellt eine Verbindung zum Datenbankserver her. Alle Parameter sind im Prinzip optional, aber in der Regel werden die entsprechenden Zugangsdaten für die Datenbank (die Sie vom Provider erhalten haben) in die Parameter geschrieben. Bleiben die Parameter leer, gilt: *hostname*, Name des *Benutzers*, dem der Server gehört, leeres *Kennwort*.

mysql_create_db

int `mysql_create_db`(string datenbankname [, int Verbindungskennung])

Die Funktion legt eine Datenbank mit dem Namen datenbankname an. Sie müssen das Recht haben, eine Datenbank anlegen zu dürfen.

mysql_db_query

int `mysql_db_query`(string datenbank, string anfrage [, int verbindungskennung])

Die Funktion sendet eine *SQL-Anweisung* (anfrage) an die Datenbank (datenbank). Das Ergebnis wird bei einer `SELECT`-Anweisung in einer Ergebnisliste zurückgegeben.

siehe auch die Erklärungen der SQL-Befehle mysql_query(), mysql_fetch_array(), mysql_result()

mysql_fetch_array

array `mysql_fetch_array`(int Datensatz[, int Ergebnistyp])

Die Funktion schreibt aus der Ergebnisliste einer Datenbankabfrage den aktuellen Datensatz in ein assoziatives Array. Die Feldnamen der Tabelle werden in dem Array als Schlüssel verwendet.

siehe auch mysql_query(), mysql_db_query(), mysql_result()

mysql_free_result

int `mysql_free_result`(int Ergebnisliste)

Mit dieser Funktion wird der Arbeitsspeicher, der von einer Ergebnisliste einer SQL-Anfrage belegt wird, wieder freigegeben.

mysql_num_rows

int `mysql_num_rows`(int Ergebniskennung)

Die Funktion gibt die Menge der Datensätze in der Ergebnisliste einer Datenbankabfrage zurück.

Beispiel (Scriptfragment):

```
$datenb=@msql_connect("localhost" ,"user", "password");
$db_selcet = @mysql_select_db(datenbankname);
$result = mysql_query("select * from tabelle");
$menge = mysql_num_rows($result);
echo "es gibt $menge Datensätze";

while($row = mysql_fetch_row($result)){
echo $row[1] ." ".$row[2];
```

mysql_result

int `mysql_result`(int Ergebniskennung, int Datensatz-Index [, mixed Feld])

Die Funktion gibt anhand der Ergebniskennung und der Angabe des Datensatz-Indexes den Inhalt eines Feldes zurück.

Beispiel (Scriptfragment):

```
$datenb=@msql_connect("localhost" ,"user", "password");
$db_selcet = @mysql_select_db(datenbankname);
$result = mysql_query("select * from tabelle");
for($i=0;$i<msql_num_rows($result);$i++{
$feld = mysql_result($result, $i, "tabelle.ID");

echo "ID: $feld";
$feld2 = mysql_result($result, $i, 1);
echo "Name: $feld2 ";
}
```

nl2br

string `nl2br` (string string)

Die Funktion wandelt aus einem String sämtliche Zeilenumbrüche in die HTML-Entsprechung (
) um, so dass sie im Browser angezeigt werden.

number_format

string `number_format`(float zahl [,int dezimalstellen [,string Dezimaltrennzeichen [,string Tausendertrennzeichen]]])

Die Funktion formatiert eine Zahl. Die Bedeutung der Parameter:

zahl	der zu formatierende Wert
Dezimalstellen	Anzahl der Nachkommastellen, es wird nicht gerundet, sondern abgeschnitten
Dezimaltrennzeichen	Trennzeichen vor den Nachkommastellen
Tausendertrennzeichen	legt das Tausendertrennzeichen fest

opendir

int `opendir`(string Pfad)

Mit `opendir` (vergleichbar mit `fopen`) wird ein Handle auf ein Unterverzeichnis erzeugt. Dieses Handle benötigen Sie, um auf das Verzeichnis, z.B. um den Inhalt aufzulisten, zuzugreifen.

readdir

string `readdir`(int Handle)

Die Funktion liest Step by Step den gesamten Inhalt eines Verzeichnisses aus. Am Ende des Verzeichnisses wird `false` zurückgegeben. Auch die Einträge ».« und »..« werden ausgelesen. Nach dem Aufruf wandert die Funktion automatisch zum nächsten Eintrag, sodass beim zweiten Aufruf der nächste Eintrag des Verzeichnisses zurückgegeben wird usw.

Mit einem Handle `$verzeichnis1` und der Zuweisung des Verzeichniseintrags in eine Variable `$file` könnten die entsprechenden Programmzeilen folgendermaßen aussehen:

```
$verzeichnis1 = opendir ("akuellesVerzeichnis");
while($file = readdir($verzeichnis1))
{echo "file";
}
```

readfile

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

Der Befehl liest eine Datei (filename) aus und gibt sie direkt im Browser aus.

Ein Beispiel, wenn in der TXT-Datei (bsp.txt) folgender Text steht:

Dies ist der Inhalt der Text-Datei
 Wir hoffen, Ihnen hilft das Beispiel

```
readfile(bsp.txt);
```

Ausgabe:

```
Dies ist der Inhalt der Text-Datei
Wir hoffen, Ihnen hilft das Beispiel
```

siehe auch fopen(), fgets(), fclose()

require

`require`(string dateiname)

Die Funktion liest die angegebene Datei ein und wertet sie aus. Die Datei wird nur einmal eingelesen.

rewind

int `rewind`(int file)

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

Beispiel:

```
$file = fopen("buch.txt", "r+");
for ($i=5; $i<10;$i++;
{rewind($file);
echo fgets($file, 100);
}
```

siehe auch fopen(), fgets(), fputs(), close()

round

double `round`(double val[, int kommastellen])

Die Funktion rundet eine Zahl (val) auf oder ab. In dem optionalen Parameter (Kommastellen) kann man die Nachkommastellen festlegen, nach denen gerundet wird.

```
echo round(10.86); // Ausgabe: 11
echo round(10.25); // Ausgabe: 10
echo round(10.86 , 1); // Ausgabe: 10.9
```

siehe auch ceil(), floor()

setcookie

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

Die Funktion sendet einen Cookie – übertragen im `Header` – an den Browser. Sie dürfen sie nur einsetzen, wenn zuvor keinerlei Ausgabe an den Browser geschickt wurde, da in diesem Fall *PHP* die `Header`-Informationen automatisch erstellt und »voreilig« an den Browser sendet. Als Ausgabe werden auch Leerzeichen und Leerstellen angesehen. Alle Parameter sind optional außer der Name des Cookies (name) und der Wert (value).

Parameter	Bedeutung
expire	erwartet die Zeit, wann das Cookie abläuft, als UNIX-Zeitstempel. Verwenden Sie die Funktion time(), um den Zeitstempel zu ermitteln und addieren Sie die Haltbarkeitsdauer in Sekunden hinzu.
path	eine Pfadangabe als String, für die das Cookie gültig ist
domain	Domäne an die das Cookie zurückgesendet werden soll
secure	Mit 1 wird das Cookie nur zurückgesendet, wenn eine sichere Verbindung besteht.

Beachten Sie, dass die unterschiedlichen Browsertypen teilweise unterschiedlich mit diesen Angaben umgehen, und es zu Fehlern kommen kann.

siehe auch header()

str_replace

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

Ein Text wird durchsucht (inhalt) und alle Vorkommen eines Strings (zeichen) durch einen anderen (ersatzzeichenkette) ersetzt. Groß- und Kleinschreibung wird beachtet.

Beispiel:

```
$alttext = "viele Sterne";
$alttext = str_replace("viele Sterne", "***", $alttext);
echo "hier gibt es $alttext";
```

Nun wird im Browser ausgegeben:

```
hier gibt es viele ***
```

siehe auch ereg_replace()

strlen

int `strlen` (string zeichenkette)

Die Funktion gibt die Länge einer Zeichenkette zurück.

```
$text="Ein kurzer Text";
echo strlen($text);
```

Ausgabe:

```
15
```

time

int `time`()

Die Funktion gibt die aktuelle Zeit zurück, die über den sogenannten UNIX-timestamp ermittelt wird. Der zählt die Sekunden ab dem 1.1.1970, 00:00 Uhr.

Deswegen gibt `echo time();` einen Wert zurück, der in der Form kaum brauchbar ist, beispielsweise 6001010913832. Um die Zeit zu formatieren, wird die Funktion `date()` eingesetzt.

siehe auch date(), mktime()

trim

string `trim`(string text1)

Die Funktion entfernt überflüssige Zeichen am Anfang und Ende einer Zeichenkette (text1). Die Funktion wird vor allem eingesetzt, um Leerstellen zu entfernen, man kann sie aber auch dazu verwenden, die folgenden nicht ausgewerteten Zeichen zu entfernen:

\nneue Zeile
\rZeilenumbruch (Carriage Return)
\thorizontaler Tabulator
HTML zeigt überflüssige Leerstellen übrigens nicht an.

siehe auch ltrim(), rtrim()

uniqid

int `uniqid`(string prefix [,boolean Icg])

Die Funktion erzeugt eine eindeutige ID mit dem angegebenen Präfix. Die ID wird generiert auf Basis der aktuellen Zeit in Mikrosekunden. Das Präfix kann 114 Zeichen lang sein und wird der ID vorangestellt.

Wenn in den optionalen Parameter (lcg) der Wert `true` geschrieben wird, fügt `uniqid` einen zusätzlichen Wert hinzu, mit der Folge, dass die ID noch einmaliger wird.

Beispiel:

```
$testzahl = uniqid("buch", true);
```

Die Ausgabe, eine lange Zahl bestehend aus Buchstaben und Zahlen, beginnt mit »buch«-.

Anmerkung: Der Einsatz der Funktion `uniqid` mit Icg ist vor allem sinnvoll bei Webseiten mit sehr hohen Zugriffszahlen, also da, wo tatsächlich im Mikrosekundentakt auf die Seiten zugegriffen wird, wie etwa bei bekannten Suchmaschinen. Für den »Hausgebrauch« spielt das Argument Icg keine große Rolle!

unlink

int `unlink`(string dateiname)

Mit dem Befehl wird eine Datei (dateiname) vom Server gelöscht.

unset

int `unset`(mixed variable)

Die Funktion löscht eine Variable oder ein Array. Der Scriptteil könnte so aussehen:

```
$var="test";
if($var){echo $var;}else{echo "Die Variable wurde gelöscht";}
unset($var);
if($var){echo $var;}else{echo "Die Variable wurde gelöscht";}
```

Ausgabe:

```
test
Die Variable wurde gelöscht
```

urldecode

string `urldecode`(string zeichenkette)

Der Befehl dekodiert einen String (zeichenkette), der vorher mit `urlencode()` kodiert wurde.

siehe auch urlencode()

urlencode

string `urlencode`(string name)

Mit dieser Funktion kann ein String (name) für die Übertragung via URL kodiert werden. Alle Sonderzeichen (außer Bindestrich, Unterstrich, Punkt) werden durch das Prozentzeichen gefolgt von zwei Hexzeichen kodiert. Sie können die Informationen wieder mit `urldecode()` dekodieren.

siehe auch urldecode()

Stichwortverzeichnis