



PHP 5.6 -
Dynamische Webseiten erstellen
Grundlagen

Bénédict
BVS
Business-School

Webcode – Übungsdateien

442E-45F5-9C99

PHP 5.6

Stephan Heller

1. Ausgabe, März 2015

ISBN: 978-3-86249-415-6

**Grundlagen zur
Erstellung dynamischer
Webseiten**

GPHP56



HERDT

1 Informationen zu diesem Buch.....	4
1.1 Voraussetzungen und Ziele.....	4
1.2 Aufbau und Konventionen	5
2 Einführung in PHP	6
2.1 PHP-Code in Webseiten.....	6
2.2 Informationen zu PHP	9
2.3 PHP-Version 5.6.x.....	11
3 Grundlegende Sprachelemente	12
3.1 PHP in HTML einbinden.....	12
3.2 Codieren von PHP-Skripten	14
3.3 Daten im Browser ausgeben	16
3.4 Grundlagen zur Fehlersuche in PHP-Skripten	21
3.5 Übung	24
4 Variablen und Operatoren	26
4.1 Variablen.....	26
4.2 Variablen und Operatoren für Zahlen.....	28
4.3 Variablen und Operatoren für Zeichenketten.....	30
4.4 Konstanten	34
4.5 Übung	37
5 Kontrollstrukturen.....	38
5.1 Kontrollstrukturen einsetzen.....	38
5.2 Die einfache if-Anweisung.....	39
5.3 Die if-Anweisung mit else-Zweig.....	42
5.4 Erweiterte if-Anweisung mit elseif.....	44
5.5 Verschachtelte if-Anweisungen	45
5.6 Fallauswahl mit der switch-Anweisung.....	46
5.7 Schleifen.....	49
5.8 Mit der while-Schleife arbeiten.....	49
5.9 Mit der for-Schleife arbeiten.....	51
5.10 Schleifen abbrechen	52
5.11 Übung	54
6 Arrays	56
6.1 Grundlagen zu Arrays	56
6.2 Indizierte eindimensionale Arrays erstellen	57
6.3 Assoziative eindimensionale Arrays erstellen	58
6.4 Arrays mit der Kurzschreibweise erstellen	59
6.5 Mit eindimensionalen Arrays arbeiten.....	60
6.6 Daten aus eindimensionalen Arrays extrahieren	62
6.7 Mehrdimensionale indizierte Arrays erstellen	64
6.8 Mit mehrdimensionalen assoziativen Arrays arbeiten.....	65
6.9 Daten aus mehrdimensionalen Arrays extrahieren	67
6.10 Den passenden Array-Typ verwenden.....	69
6.11 Weitere Informationen zu Arrays in PHP	69
6.12 Übung	70
7 Mit Formularen arbeiten	72
7.1 Interaktion mit PHP.....	72
7.2 Formulare mit PHP auswerten	74
7.3 Übung	85
8 Funktionen	88
8.1 Funktionen erstellen und aufrufen	88
8.2 Mit Funktionen arbeiten.....	90
8.3 Der Gültigkeitsbereich von Variablen	98
8.4 PHP-Dateien einbinden mit include () und require ()	100
8.5 Andere Dateitypen einbinden	103
8.6 Übung	104
9 Mit Daten aus externen Dateien arbeiten	106
9.1 Externe Dateien nutzen	106
9.2 Dateien öffnen, lesen und schließen.....	106
9.3 Weitere Möglichkeiten zum Lesen von Dateien	109
9.4 In Dateien schreiben	111
9.5 Zugriffszähler für eine Webseite	114
9.6 Übung	116
10 Zeichenketten-Funktionen.....	118
10.1 Zeichenketten ausgeben.....	118
10.2 Nach Zeichenketten suchen	122
10.3 Position und Teil einer Zeichenkette ermitteln	125
10.4 Zählen innerhalb von Zeichenketten	126
10.5 Zeichenketten vergleichen.....	128
10.6 Zeichenketten modifizieren.....	128
10.7 Mit Arrays und Zeichenketten arbeiten	132
10.8 Übung	134

11 Datum und Zeit	136
11.1 Datum und Zeit ermitteln.....	136
11.2 Datum und Zeit formatieren	138
11.3 Datumsangabe an Sprache anpassen.....	139
11.4 Länder- und Spracheinstellungen ändern	140
11.5 Zeitfunktionen	142
11.6 Datumsangaben überprüfen	146
11.7 Übung.....	147
12 Sessions.....	148
12.1 Mit Sessions arbeiten	148
12.2 Session starten bzw. fortsetzen.....	149
12.3 Daten einer Session speichern	151
12.4 Daten einer Session abrufen.....	154
12.5 Sessiondaten und Session löschen	155
12.6 Fallbeispiel "Shop"	157
12.7 Übung.....	164
13 Grundlagen Datenbank MySQL.....	166
13.1 MySQL-Datenbanken mit phpMyAdmin verwalten	166
13.2 MySQL-Datenbanken mit phpMyAdmin erstellen.....	168
13.3 Mit einer MySQL-Tabelle arbeiten	172
13.4 SQL-Dumps exportieren und importieren	173
13.5 PHP und MySQL	174
13.6 MySQL-Abfragen	179
13.7 Rückgabe aus MySQL-Abfrage auswerten.....	181
13.8 Formulardaten in einer MySQL-Datenbank speichern	184
13.9 Übung.....	185
A Installation und Konfiguration der Software	188
A.1 Installation und Konfiguration von XAMPP.....	188
A.2 Mit XAMPP arbeiten.....	190
A.3 Installation und Konfiguration von Notepad++	192
A.4 Mit den XAMPP-Konfigurationsdateien arbeiten.....	193
A.5 Zugriffsrechte von MySQL mit phpMyAdmin regeln.....	195
A.6 Globale Zugriffsrechte des MySQL- Administrators <code>root</code> ändern	197
Stichwortverzeichnis.....	200

1 Informationen zu diesem Buch

In diesem Kapitel erfahren Sie

- ✓ welche Kenntnisse für dieses Buch vorausgesetzt werden
- ✓ wie dieses Buch aufgebaut ist
- ✓ welche Konventionen für dieses Buch gelten

1.1 Voraussetzungen und Ziele

Zielgruppe

Das Buch richtet sich an Webentwickler, Studenten und Schüler, die fundiertes Grundwissen über die dynamische Webentwicklung mit PHP erhalten möchten.

Empfohlene Vorkenntnisse

Um effektiv mit PHP arbeiten zu können, sind Kenntnisse in folgenden Bereichen vorteilhaft:

- ✓ Betriebssysteme
- ✓ Arbeiten mit Browsern
- ✓ HTML-Grundlagen (mindestens Zeichen- und Absatzformatierung, Tabellen, Formulare)
- ✓ Grundkenntnisse Datenbanken bzw. MySQL

Lernziele

Das Buch führt Sie schrittweise in PHP ein. Sie lernen, wie PHP grundsätzlich funktioniert, erfahren, wie grundlegende Sprachelemente aussehen und wie Sie PHP in HTML verwenden können. Sie gewinnen Kenntnis darüber, welche Arten von Variablen es gibt, was Arrays sind und in welcher Weise Sie mit Variablen arbeiten können. Sie lernen Kontrollstrukturen und Funktionen kennen, welche die Grundbestandteile eines PHP-Skriptes darstellen.

Diese Grundlagen benutzen Sie dann in unterschiedlichen Anwendungsbeispielen. Beispielsweise werten Sie Daten aus HTML-Formularen aus und verarbeiten diese weiter. Sie lernen, wie Sie Daten (z. B. Datumswerte) verändern können, wie Sie über Sessions PHP-Skripten ein „Gedächtnis“ verleihen und in welcher Art Sie auf Datenbanken zugreifen können.

Hinweise zu Soft- und Hardware

PHP ist eine Open-Source-Software. Sie benötigen zur Erstellung von Webseiten mit PHP-Anweisungen nicht mehr als einen Texteditor. Damit Sie Ihre PHP-Programme direkt an Ihrem Rechner testen können, empfehlen wir Ihnen die folgende Software:

- ✓ die kostenfreie Entwicklungsumgebung XAMPP in der Version 5.6.3 vom 19. November 2014 (<https://www.apachefriends.org/index.html>) unter anderem mit den Bestandteilen:
 - ✓ PHP, Version 5.4.4,
 - ✓ Apache-Webserver, Version 2.4.10,
 - ✓ MySQL, Version 5.6.21;
- ✓ den Freeware-Texteditor Notepad++ in der Version 6.7.3 (<http://notepad-plus-plus.org/download/>);
- ✓ einen Webbrower. In diesem Buch sind alle Screenshots mit dem Firefox-Browser erstellt. Grundsätzlich spielt der Browser für das Arbeiten mit PHP keine Rolle. Sie können ebenfalls Google Chrome, Microsoft Internet Explorer, Apple Safari oder auch Opera verwenden.

Diese Software wurde mit ihren Standardeinstellungen für die Funktionsbeschreibungen in diesem Buch verwendet. Installations- und Konfigurationshinweise zur Software finden Sie im Anhang.

1.2 Aufbau und Konventionen

Typografische Konventionen

Damit Sie bestimmte Elemente auf einen Blick erkennen und zuordnen können, werden diese im Text durch eine besondere Schreibweise hervorgehoben. So werden beispielsweise Bezeichnungen für Programm-elemente wie Register oder Schaltflächen immer *kursiv* geschrieben und wichtige Begriffe **fett** hervorgehoben.

Kursivschrift	kennzeichnet alle von Programmen vorgegebenen Bezeichnungen für Schalt-flächen, Dialogfenster, Symbolleisten, Menüs bzw. Menüpunkte (z. B. <i>Datei - Schließen</i>) sowie alle vom Anwender zugewiesenen Namen wie Dateinamen, Ordnernamen, eigene Symbolleisten, Hyperlinks und Pfadnamen.
Courier New	kennzeichnet Programmtext, Programmnamen, Funktionsnamen, Variablennamen, Datentypen, Operatoren etc.
[]	Bei Darstellungen der Syntax einer Programmiersprache kennzeichnen eckige Klammern optionale Angaben.
/	Bei Darstellungen der Syntax einer Programmiersprache werden alternative Elemente durch einen Schrägstrich voneinander getrennt.

Was bedeuten die Symbole im Buch?



Hilfreiche Zusatzinformation



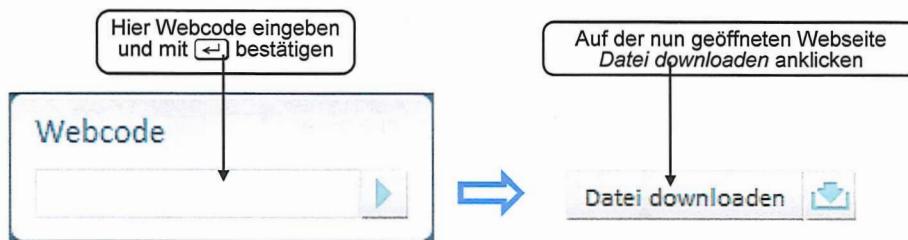
Praxistipp



Warnhinweis

Beispiel-, Übungs- und Ergebnisdateien

Mit dem Kauf dieses Buches haben Sie die Berechtigung erworben, die Beispiel-, Übungs- und Ergebnisdateien sowie die Linkliste, die die im Buch genannten Internetadressen – nach Kapiteln sortiert – enthält, von der Website des HERDT-Verlages unter www.herd़t.com herunterzuladen. Nutzen Sie hierzu den Webcode, der sich rechts oben auf der Titelseite dieses Buches befindet:



Mithilfe der Beispieldateien können Sie die erläuterten Programmfunctionen direkt nachvollziehen. Die Übungen am Ende der einzelnen Kapitel lassen sich ohne weitere Vorbereitungen mit den jeweiligen Übungsdateien durchführen.

Anhand der Ergebnisdateien kontrollieren Sie schnell, ob Sie die entsprechenden Übungen korrekt ausgeführt haben.

Weitere Medien von HERDT nutzen

Hat Ihnen das vorliegende Buch gefallen, besuchen Sie doch einmal unseren Webshop unter www.herd़t.com.

Wir wünschen Ihnen viel Spaß und Erfolg mit diesem Buch.

Ihr Redaktionsteam des HERDT-Verlags

2 Einführung in PHP

In diesem Kapitel erfahren Sie

- ✓ was unter dem Begriff PHP zu verstehen ist
- ✓ wo PHP ausgeführt wird und was an den Browser geliefert wird
- ✓ welche Entwicklungsgeschichte PHP hat
- ✓ was neu ist in PHP 5.6.x

2.1 PHP-Code in Webseiten

Was ist PHP?

PHP (Abkürzung für PHP: Hypertext Preprocessor, früher auch Personal Home Page Tools) ist eine **Open-Source-Skriptsprache**, die speziell für den Einsatz im **Internet** entwickelt wurde.

Die Stärken von PHP liegen in der recht leichten Erlernbarkeit und in der breiten Funktionspalette.

PHP setzt dort an, wo HTML seine Grenzen erreicht: HTML-Seiten sind starr. Mithilfe von PHP können auf einer Webseite

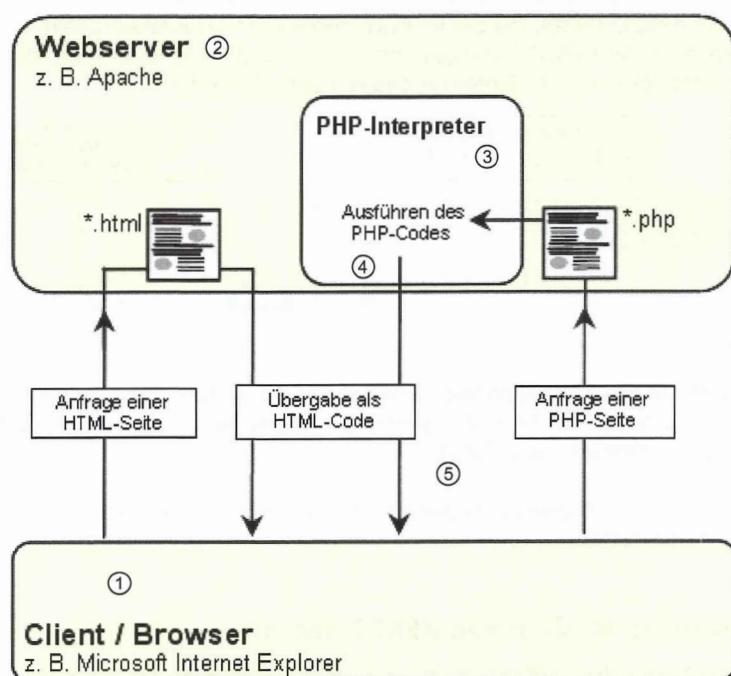
- ✓ Interaktionen eingebaut,
- ✓ Datenbanken gesteuert oder
- ✓ die Seite individuell an das Benutzerverhalten angepasst werden.

Webseiten mit PHP-Code verarbeiten

PHP-Code wird von einem Webserver, also serverseitig verarbeitet. Das Ergebnis wird danach vom Webserver als HTML-Code (Quellcode) an den Browser gesendet. Ob der HTML-Code aus einer statischen HTML-Datei stammt oder das Ergebnis eines ausgeführten PHP-Skripts ist, ist für den Browser nicht ersichtlich. Zur Darstellung erhält der Browser in beiden Fällen HTML-Code.

Öffnet der Betrachter eine PHP-Webseite im Browser ①,

- ✓ werden die Anweisungen von PHP auf dem Server ② interpretiert ③,
- ✓ ausgeführt ④ und
- ✓ das Ergebnis als HTML-Code an den Browser zurückgesendet ⑤.

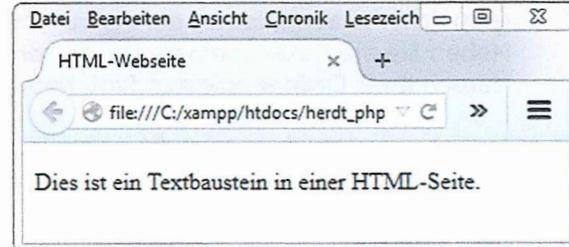


Der Nutzer sieht im Browser nur die Darstellung des zurück gelieferten HTML-Codes (Resultat nach den Vorgängen ③ und ④). Der PHP-Quellcode selbst ist dem Betrachter nicht zugänglich. Programmieralgorithmen, die Verarbeitung von Daten oder beispielsweise Zugriffe auf Datenbanken bleiben dem Nutzer verborgen.

Webseite ohne PHP-Code – Beispiel: *html-inhalt.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML-Webseite</title>
  </head>
  <body>
    <p>Dies ist ein Textbaustein in einer HTML-Seite.</p>
  </body>
</html>
```

HTML-Dateien können ohne weitere Software von jedem beliebigen Browser angezeigt werden. Es reicht ein Doppelklick auf den Dateinamen, um die Datei zu öffnen. In der Adresszeile des Browsers erscheint der Pfad des Dateisystems, in dem die HTML-Datei liegt.



Webseite um PHP-Code erweitern

- **Schritt 1:** Kopieren Sie die Datei *html-inhalt.html* und ändern Sie den Dateinamen in *html-inhalt-und-php.php*.

Achten Sie auf die Angabe der Dateinamenerweiterung *php*. Sobald die Dateinamenerweiterung von *html* in *php* geändert ist, haben Sie eine lauffähige PHP-Datei erzeugt.

PHP-Dateien benötigen im Gegensatz zu HTML-Dateien einen Webserver, der PHP verarbeitet (interpretiert). Webserver werden von Internet-Providern zur Verfügung gestellt. Mit dem XAMPP-Paket (vgl. Abschnitt A.1) können Sie einen lokalen Webserver für die Entwicklung von PHP auf Ihrem eigenen Rechner installieren.



Die Ausgabe der Datei *html-inhalt-und-php.php* ist identisch mit der Datei *html-inhalt.html*. Beide Dateien enthalten den gleichen HTML-Code und sonst keine weiteren Inhalte. HTML-Code wird vom PHP-Interpreter weder interpretiert noch ausgeführt, sondern direkt an den Browser gesendet und dort dargestellt.

Über PHP-Code soll zusätzlich der Text Ich freue mich auf PHP! ausgegeben werden:

- **Schritt 2:** Öffnen Sie die Datei *html-inhalt-und-php.php* im Editor (z. B. Notepad++) und fügen Sie PHP-Code in den HTML-Code ein:

Beispiel: *html-inhalt-und-php.php*

```
①  <!DOCTYPE html>
<html>
  <head>
    <title>HTML-Webseite mit PHP</title>
  </head>
  <body>
    <p>Dies ist ein Textbaustein in einer HTML-Seite.<br>
      ② <?php
        echo "Ich freue mich auf PHP!";
      ?>
    </p>
  </body>
</html>
```

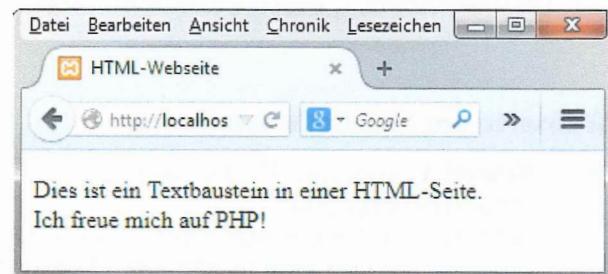
- ① Fügen Sie hinter dem vorhandenen Text einen **Zeilenumbruch** `
` ① ein, um die Ausgabe, die vom HTML- und PHP-Code erzeugt wird, in zwei Zeilen zu bewirken.
- ② Beginnen Sie einen PHP-Block im HTML-Code mit dem öffnenden PHP-Tag `<?php` ②.

- ③ Verwenden Sie zur Ausgabe des Textes den PHP-Befehl echo, gefolgt vom Text in Anführungszeichen. Befehle werden in PHP durch ein **Semikolon** abgeschlossen.
- ④ Beenden Sie den PHP-Block mit einem schließenden PHP-Tag ?> ④.

 PHP-Code kann an jeder **beliebigen** Stelle und **beliebig** oft im HTML eingefügt werden. Wichtig ist, dass jeder PHP-Block von dem öffnenden PHP-Tag <?php und dem schließendem PHP-Tag?> umschlossen ist.

- **Schritt 3:** Starten Sie den XAMPP-Webserver über das Control Panel, falls dieser nicht als Dienst installiert ist (vgl. Installationshinweise im Download). Rufen Sie die PHP-Datei im Browser mit folgender URL auf: [http://localhost/\[Pfad zur Datei\]/html-inhalt-und-php.php](http://localhost/[Pfad%20zur%20Datei]/html-inhalt-und-php.php). Falls Sie die Datei *html-inhalt-und-php.php* direkt im Ordner */htdocs* des Webservers gespeichert haben, lautet der Aufruf im Browser <http://localhost/html-inhalt-und-php.php>. Haben Sie die Datei in einem Unterordner abgelegt, müssen Sie *[Pfad zur Datei]* durch den Namen Ihrer Ordner ersetzen (vgl. Kapitel 3).

Die nebenstehende Abbildung zeigt die Webseite „html-inhalt-und-php.php“ im Browser.



Den vom Server zurückgelieferten HTML-Code sehen Sie in der nachfolgenden Abbildung:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML-Webseite</title>
</head>
<body>
<p>Dies ist ein Textbaustein in einer HTML-Seite.<br>
Ich freue mich auf PHP!</p>
</body>
</html>
```

Die PHP-Anweisungen wurden durch den auszugebenden Inhalt ersetzt. Der Betrachter sieht den ursprünglichen PHP-Code im HTML-Code nicht.

Merkmale von PHP

- ✓ PHP wurde speziell für die Programmierung dynamischer Webseiten entwickelt. Funktionen in PHP sind deshalb auf die Programmierung von Internetanwendungen abgestimmt.
- ✓ PHP zeichnet sich durch einen großen Funktionsumfang aus.
- ✓ PHP ist plattformunabhängig. Webserver mit PHP-Unterstützung stehen sowohl für Linux, Mac OS und Windows zur Verfügung.
- ✓ Die PHP-Anweisungen werden direkt auf dem Server ausgeführt und nicht vom Browser (Client).
- ✓ Der PHP-Quellcode ist für den Betrachter nicht sichtbar, sondern nur die auf dem Webserver generierte HTML-Ergebnisdatei.
- ✓ PHP ist fehlertoleranter als andere Programmiersprachen wie beispielsweise JAVA.
- ✓ PHP ist kostenlos erhältlich.

2.2 Informationen zu PHP

Entwicklung von PHP

1994 begann Rasmus Lerdorf mit der Entwicklung einer Skriptsprache für das Internet. Die Sprache sollte einfach zu erlernen sein. Die erste Version nannte er PHP, abgeleitet von „Personal Home Page Tools“, einer Sammlung von diversen Skripten zum Einbinden in Webseiten. Der Interpreter beherrschte wenige Befehle, sodass der Funktionsumfang klein war.

Ein Jahr später erweiterte Lerdorf den Funktionsumfang, indem er den Interpreter neu programmierte. Hinzu kamen die Funktionen des vorhandenen Formular Interpreters, kurz FI. Dieser Interpreter war in der Lage, Formulardaten ohne CGI (Common Gateway Interface) zu verarbeiten. Die Kombination der Home Page Tools und des Formular Interpreters (PHP2) nannte er PHP/FI. Das Neue an dieser Version war die Anbindung an die MySQL-Datenbank.

Das Privatprojekt Lerdorfs wurde 1997 von einem Team von Programmierern übernommen. Mit dabei Andi Gutmans und Zeev Suraski, die Gründer der Firma Zend Technologies Ltd.. Die Funktionen der PHP/FI-Version wurden in das PHP3-Format portiert und neue Befehle hinzugefügt. Mit der Version PHP3 und ihrer Anbindung an verschiedenste Datenbanken begann die Verbreitung der Skriptsprache zur dynamischen Seitengenerierung. Konkurrenzprodukte sind andere serverseitige Programmiersprachen wie ASP.NET, Java Server Pages (JSP) oder ColdFusion.

Seit PHP4 bildet ein von Zend entwickelter Compiler zur schnelle(re)n Ausführung des Codes, die sogenannte „Zend Engine“, das Rückgrat der PHP-Programmierung. PHP wurde durch Erweiterungen seines Funktionsumfangs, wie z. B. das Sessionmanagement, verbessert.

PHP-Versionen und Verbreitung

PHP5 ist seit Sommer 2004 auf dem Markt. Eine stark verbesserte Zend Engine II ist ebenso wie zahlreiche Verbesserungen, z. B. in der objektorientierten Programmierung und in Fragen rund um die Sicherheit der Programmierung, dafür verantwortlich, dass die Zahl der Webseiten mit PHP-Unterstützung schnell wächst. Mittlerweile laufen über 20 Mio. Websites auf Webservern mit PHP-Unterstützung. PHP wird auf 81% aller Webseiten eingesetzt (Stand: Oktober 2013, Quelle: <http://de.wikipedia.org/wiki/PHP#Verbreitung>) und ist damit die am häufigsten verwendete Programmiersprache zum Erstellen von Webseiten.

PHP wird ständig weiter entwickelt. Die Entwicklung von PHP 5.3.x ist eingestellt, PHP 5.4.x, 5.5.x und 5.6.x werden jedoch weiterentwickelt (Stand: Dezember 2014), um Fehler oder Sicherheitslücken zu beseitigen. Zeitgleich wird an PHP 7 gearbeitet. Aufgrund von Problemen bei der Entwicklung von PHP 6 wurde diese Version eingestellt, eine Version PHP 6 wird es nicht geben. Die PHP-Version 5.6.3 wurde im November 2014 veröffentlicht.

Internet-Provider setzen in der Regel nur stabile Versionen von PHP ein. Der Anteil von Providern, die PHP 5.6 anbieten, liegt unter 10% (Stand: November 2014, Quelle: <http://phpmagazin.de/artikel/php-versionen-verbreitung-177514>). Das bedeutet für die Entwicklung, dass Sie neue PHP-Features nicht nutzen sollten, wenn der eigene Provider eine ältere PHP-Version einsetzt. Die Information, welche PHP-Version der Provider anbietet, auf dem das geplante Webprojekt laufen soll, sollten Sie sich vor Beginn der Entwicklung einholen. Wann PHP 5.6 bei den Providern eingesetzt wird, hängt von verschiedenen Faktoren ab und kann nur schwer prognostiziert werden. Jahreszeiträume von ein bis zwei Jahren zwischen Veröffentlichung und Einsatz bei den Providern sind nicht ungewöhnlich.



Funktionsumfang von PHP

PHP hat heute einen großen Funktionsumfang, mit dem weitestgehend alle gängigen Anforderungen im Webumfeld gelöst werden können. Eine Auswahl oft benötigter Funktionen finden Sie nachfolgend:

- ✓ HTML-Seiten generieren, in denen benutzerbezogene Daten verarbeitet sind.
- ✓ Verarbeiten unterschiedlicher Daten-Typen, wie Integer, Fließkommazahlen, Zeichenketten oder Arrays.
- ✓ Auslesen, Überprüfung und Verarbeiten von Formulardaten.
- ✓ Daten aus Datenbanken auslesen, bearbeiten und wieder in Datenbanken speichern.

- ✓ Zeichenketten auslesen, verändern, abschneiden, umwandeln, in bestimmten Formaten wieder ausgeben.
- ✓ Datums- und Zeitangaben auslesen, erkennen, generieren, verändern.
- ✓ E-Mails erstellen und versenden.
- ✓ Grafiken öffnen oder generieren, verändern, speichern.
- ✓ Dateien (z. B. Text- oder CSV-Dateien) öffnen, auslesen, verändern, speichern.
- ✓ Über Schnittstellen Dienste anderer Webseiten aufrufen, einlesen, auswerten.
- ✓ Cookies speichern und auslesen, Sessions verwalten.

Über die Kernfunktionalitäten ist PHP zusätzlich um verschiedenste Bibliotheken erweiterbar, beispielsweise ImageMagick. Die Bibliothek ImageMagick bietet mehr Funktionalitäten als PHP-eigene Funktionen zur Bildverarbeitung.

Im Rahmen der Grundlagen von PHP 5.6 werden die Basisfunktionalitäten von PHP gezeigt, die ausreichend sind, um eine dynamische Webseite zu programmieren.

Die Kombination von PHP und der Datenbank MySQL ist auf der Mehrzahl aller Webserver zu finden. PHP ist jedoch nicht auf MySQL beschränkt, sondern kann mit unterschiedlichen Datenbanken arbeiten – entweder direkt über eigene PHP-Funktionalitäten oder auch über ODBC-Schnittstellen. So kann PHP beispielsweise auch mit Oracle Datenbanken arbeiten.

PHP unterstützt darüber hinaus die Kommunikation mit anderen Services, z. B. Protokollen wie LDAP, IMAP, SNMP, NNTP, POP3 oder HTTP.

Zudem ist PHP zur Kommandozeilenprogrammierung (z. B. für sogenannte „cronjob“-Skripte, die wiederkehrende Aufgaben automatisieren) oder für die Entwicklung von Desktop-Anwendungen einsetzbar.

Informationen zu PHP im Internet

Folgende ausgewählte Webseiten bieten ausführliche Informationen zu PHP:

✓ http://www.php.net	Die wichtigste Seite für PHP-Programmierer. Hier finden Sie die komplette Dokumentation von PHP, alle Funktionen einschließlich PHP-Code-Beispielen und PHP-Versionshinweise. Auch Probleme mit Funktionen werden dort besprochen.
✓ http://www.php.net/manual/de/	Deutschsprachiges Onlinehandbuch der PHP-Dokumentationsgruppe
✓ http://www.selfphp.de	Befehlsreferenz - Tutorial - Kochbuch - Forum: für PHP-Einsteiger und professionelle Entwickler. Eine aktuelle und vollständige Version steht auch zum Download bereit: http://www.selfphp.de/de/extras/download.php
✓ http://de.selfhtml.org	Das Webprojekt SELFHTML bietet eine gute Dokumentation (Tutorial und Referenz) zu HTML, CSS und JavaScript. Auch wenn die Seite seit 2007 nicht mehr gepflegt wird, sind dort alle wesentlichen Grundlagen der Webentwicklung zu finden.
✓ http://wiki.selfhtml.org	SELFHTML-Wiki ist das Nachfolgeprojekt von SELFHTML und berücksichtigt bereits HTML5-Elemente.



<http://www.php.net> ist die Standardreferenz für PHP-Entwickler. Große Teile der Referenz stehen in Deutsch zur Verfügung. Gerade bei neueren Features lohnt sich aber der Wechsel in die englische Variante, da diese mitunter auf einem aktuelleren Stand ist.

2.3 PHP-Version 5.6.x

Viele der Neuerungen von PHP 5.6.x sind für das Erlernen der Grundlagen von PHP nicht relevant. Relevante Neuerungen für dieses Buch sind der Exponential-Operator (vgl. Kapitel 4) und der Splat-Operator (vgl. Kapitel 8).

Die wichtigsten Änderungen in PHP 5.6.x sind nachfolgend aufgeführt:

Neuerungen

- ✓ Exponential-Operator: ** (doppeltes Sternchen).
- ✓ Der Splat-Operator ... (Auslassungszeichen, drei Punkte).
- ✓ „Variadic Functions“. Über den Splat-Operator werden die übergebenen Parameter innerhalb der Funktion in ein Array übergeben.
- ✓ Per Spalt-Operator im Funktionsaufruf können mehrere Parameter als ein Array übergeben werden und werden mehrere Parameter im Funktionskopf übergeben.
- ✓ „Constant Scalar Expressions“ ermöglicht das Speichern eines Ausdrucks in Konstanten.
- ✓ Der default_charset wird nun als Standard für die Funktionen htmlentities(), html_entity_decode() und htmlspecialchars() verwendet.
- ✓ Neue Funktionen: hash_equals() und __debugInfo().
- ✓ Mehrere neue Funktionen in den Erweiterungen GMP, LDAP, cURL, OCI8, OpenSSL und Zip
- ✓ Neue globale Konstanten in den Erweiterungen LDAP, OpenSSL und PostgreSSL

Änderungen

- ✓ Dateiupload von Dateien größer 2 GB ist nun möglich.
- ✓ Verbesserung der SSL/TLS Unterstützung.
- ✓ Die Funktionen crypt(), substr_compare(), unserialize(), crypt_create_iv(), stream_socket_enable_crypto() wurden verändert.
- ✓ Funktionen des XMLReader wurden erweitert.
- ✓ Veränderte Funktionalitäten in der Erweiterung Open SSL.

Seit PHP 5.6 veraltete Funktionen/Features

- ✓ Funktionen sollten nur noch entsprechend ihrer Deklaration aufgerufen werden. In zukünftigen PHP-Versionen wird ein nicht deklarationsgerechter Aufruf von Funktionen zum Fehler führen.
- ✓ iconv- und mbstring-Einstellungen bezüglich der Zeichencodierung gelten als veraltet.

Änderungen, die nicht abwärtskompatibel sind

- ✓ Die Funktion json_decode() wurde verschärft. true, false und null im JSON müssen nun klein geschrieben werden.
- ✓ GMP-Ressourcen werden als Objekt behandelt (GMP ist eine Bibliothek zur Verarbeitung beliebig großer Integer).
- ✓ Mcrypt-Funktionen benötigen jetzt gültige Schlüssel und Initialisierungsparameter.

Da es sich beim Update auf PHP Version 5.6 um ein größeres Update handelt, ist nicht auszuschließen, dass kleinere Fehler oder Inkompatibilitäten noch nicht entdeckt wurden. Kleinere Updates wird es von daher regelmäßig mit der Versionsnummer 5.6.x geben. Die Quelltexte in diesem Buch sind mit dem XAMPP-Paket erstellt, welches die PHP-Version 5.6.3 unterstützt. Die schwerwiegendsten Fehler sollten bereits beseitigt sein.



3 Grundlegende Sprachelemente

In diesem Kapitel erfahren Sie

- ✓ wie Sie PHP-Code in ein HTML-Dokument einbinden
- ✓ wie Sie eine PHP-Datei aufrufen
- ✓ wie Sie Zeichenketten im Browser ausgeben
- ✓ wie Sie Kommentare im Quelltext einfügen
- ✓ wie Sie mit Fehlern in PHP-Skripten umgehen

Voraussetzungen

- ✓ Windows- und HTML-Kenntnisse

3.1 PHP in HTML einbinden

Dateinamenerweiterung für Dateien mit PHP-Code

PHP kann direkt innerhalb des HTML-Codes eingegeben werden. Damit der Webserver erkennt, dass es sich um eine Datei mit PHP-Anweisung(en) handelt, werden die Dokumente mit der speziellen Dateinamenerweiterung `*.php` versehen. Möglich sind zwar auch `*.php4`, `*.php5` oder `*.phtml` (dies kann u. a. in der `httpd.conf` konfiguriert werden), empfohlen wird jedoch die Dateinamenerweiterung `*.php`. Diese entspricht der Standardkonfiguration.



In einer Datei mit der Dateinamenerweiterung `*.htm` bzw. `*.html` werden PHP-Anweisungen nicht interpretiert.

PHP-Anweisungen einfügen

PHP-Anweisungen können an beliebiger Stelle im HTML-Dokument stehen. Ebenso können Sie beliebig – und beliebig oft – zwischen HTML und PHP wechseln.

Um dem Interpreter zu zeigen, dass innerhalb des HTML-Codes ein PHP-Befehl folgt, den er bearbeiten muss, wird ein spezieller Tag angegeben. Zum Einbinden eines PHP-Befehls (oder mehrerer PHP-Befehle) in eine Seite ist die folgende Variante die Standardschreibweise.

<?php PHP-Anweisung (en) ?>	XML-Schreibweise Ein sogenannter PHP-Block wird durch das Tag <code><?php</code> geöffnet und nach den PHP-Anweisungen mit dem Tag <code>?></code> geschlossen. Dies ist die empfohlene und allgemeingültige Schreibweise.
-----------------------------------	---

Die weiteren Varianten werden nur der Vollständigkeit halber erwähnt. Abhängig von der PHP-Version müssen Sie dafür in der Konfigurationsdatei `php.ini` die Werte zur PHP-Einbindung setzen (vgl. Anhang, Abschnitt *Wichtige Einträge in der php.ini*). In der Regel werden Sie Ihre Webseite bei einem professionellen Internet-Provider hosten. Dort haben Sie **keinen** Zugriff auf die `php.ini`. Die Anpassung auf die Kurzschreibweise auf dem lokalen Entwicklungssystem ist zwar möglich, da Sie jedoch die Kurzschreibweise bei Ihrem Provider nicht entsprechend anpassen können, stellt diese Anpassung eine Fehlerquelle dar. Von der Konfiguration der Kurzschreibweise ist deshalb abzuraten. Auch die ASP- und die Skript-Schreibweise werden in der Praxis kaum verwendet, und die Verwendung ist auch nicht zu empfehlen. Beachten Sie, dass Änderungen an der Konfigurationsdatei erst nach einem Neustart des Webservers wirksam werden.

<? PHP-Anweisung(en) ?>	Kurzschrifweise Falls die Konfigurationsvariable <code>short_open_tag</code> in der Datei <code>php.ini</code> auf den Wert <code>On</code> gesetzt wurde, können Sie <code>php</code> im öffnenden PHP-Tag weglassen.
<% PHP-Anweisung(en) %>	ASP-Schrifweise Wenn Sie für die Konfigurationsvariable <code>asp_tags</code> den Wert <code>On</code> gesetzt haben, können Sie ebenfalls die ASP-Schrifweise verwenden, um PHP-Anweisungen einzubinden.
<script language="php"> PHP-Anweisung(en) </script>	Skript-Schrifweise Alternativ können Sie einen Skript-Tag verwenden. Diese Schrifweise entspricht der JavaScript-Angabe in HTML.

Verwenden Sie nach Möglichkeit die allgemeingültige **XML-Schrifweise** (<?php ... ?>) zum Einfügen von PHP-Anweisungen in Ihren Skripten. Dies ist die gängige Einstellung der Webserver von Internet-Providern. Damit stellen Sie sicher, dass Ihre Seite auf allen Servern läuft.



Informationsdatei zur PHP-Konfiguration anzeigen

Sie können sich mit einem Skript die PHP-Konfiguration des Servers anzeigen lassen, auf dem dieses Skript ausgeführt wird. Zu diesem Zweck stellt PHP die Funktion `phpinfo()` bereit. Damit erhalten Sie detaillierte Informationen zur installierten PHP-Version samt installierten Erweiterungen.

Wenn Sie lokal z. B. mit dem empfohlenen XAMPP-Paket arbeiten, rufen Sie im Webbrowser Ihren lokalen Rechner als (PHP-)Webserver auf. Über die Adresse `http://localhost` – oder alternativ über die IP-Adresse `http://127.0.0.1` – wird automatisch das im Webserver definierte sogenannte *document root*-Verzeichnis geöffnet. Bei einer XAMPP-Standardinstallation ist das unter Windows der Ordner `C:\xampp\htdocs`. Unter Mac OS finden Sie das root-Verzeichnis unter `/Applications/XAMPP/htdocs`.



PHP-Dateien können nicht mit einem Doppelklick auf die Datei geöffnet werden. Sie können die Datei zwar mit der Maus in den Browser ziehen, dann wird diese jedoch lediglich als HTML-Datei dargestellt, PHP-Anweisungen werden dann nicht interpretiert. Damit der PHP-Code ausgewertet wird, muss der Aufruf der PHP-Datei über einen PHP-Interpreter erfolgen, in den Beispielen in diesem Buch immer über die Serveradresse `http://localhost` oder `http://127.0.0.1` und anschließend jeweils Pfad- und Dateiname. Weitere Informationen zu XAMPP finden Sie in der Installationsanleitung mit Hinweisen zur Konfiguration im Download.

Beispiel: `phpinfo.php`

- ▶ Erstellen Sie folgende Datei und speichern Sie sie im Verzeichnis `C:\xampp\htdocs` unter dem Dateinamen `phpinfo.php`.
- ▶ Um das Skript zu testen, rufen Sie im Browser die PHP-Datei mit folgender Adressangabe auf Ihrem lokalen Webserver auf:
`http://localhost/phpinfo.php` oder `http://127.0.0.1/phpinfo.php`

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

Beispieldatei "phpinfo.php"

Installierte PHP-Version	
System	Windows NT DAIK-TOSHIBA-2 6.1 build 7601 (Windows 7 Home Premium Edition Service Pack 1) i586
Build Date	Nov 12 2014 17:12:19
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript/nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x86\instantclient_12_1\ sdk\shared" "--with-oci8-12c=c:\php-sdk\oracle\x86\instantclient_12_1\ sdk\shared" "--enable-object-out-dir=..obj" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\windows
Loaded Configuration File	C:\xampp\php\php.ini
Scanned this dir for additional .ini files	(none)

Ausschnitt aus der Anzeige der PHP-Konfiguration

Am Anfang der Seite wird Ihnen die installierte PHP-Version angezeigt.

3.2 Codieren von PHP-Skripten

PHP-Skripte können aus Hunderten von Codezeilen bestehen. Oft ist es sinnvoll, Codezeilen mehrfach zu verwenden. Achten Sie deshalb darauf, dass Ihr Code

- ✓ leicht zu lesen und zu verstehen ist,
- ✓ gut kommentiert bzw. dokumentiert ist,
- ✓ auch von anderen Programmierern leicht zu verstehen und damit einfach zu pflegen ist,
- ✓ selbsterklärende Variablen und Funktionsnamen berücksichtigt,
- ✓ keine kryptischen Bezeichner enthält.

Nachfolgend finden Sie einige Tipps, wie Sie Ihre PHP-Skripte klar verständlich und übersichtlich gestalten können.

PHP-Anweisungen voneinander trennen

Jede PHP-Anweisung wird durch ein Semikolon beendet, damit der Interpreter erkennt, wo der auszuführende Befehl zu Ende ist. Schreiben Sie zur besseren Übersicht jede PHP-Anweisung in eine eigene Zeile und verwenden Sie Einrückungen, um den Code übersichtlicher zu machen.

Beispiel

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

Skripte gut kommentieren

Um Quellcode zu erläutern, werden Kommentare verwendet:

- ✓ Kommentare erhöhen die Nachvollziehbarkeit der Programmierung.
- ✓ Kommentare enthalten z. B. eine Kurzbeschreibung der Verwendung einer Funktion.
- ✓ Kommentare beschreiben den Zweck von Variablen oder erklären den Algorithmus eines Programms.

Kommentare können auch genutzt werden, um einen Abschnitt des PHP-Codes – für Testzwecke oder um Fehler einzugrenzen – als Kommentar zu kennzeichnen und damit vor dem PHP-Interpreter zu verstecken (dies wird als „auskommentieren“ bezeichnet), sodass dieser Abschnitt vorerst nicht ausgeführt wird.



PHP-Kommentare sind für den Betrachter nicht zu sehen, auch nicht im HTML-Quelltext der Seite im Browser. Die Kommentare werden beim Auswerten des PHP-Codes auf dem Webserver übergangen.

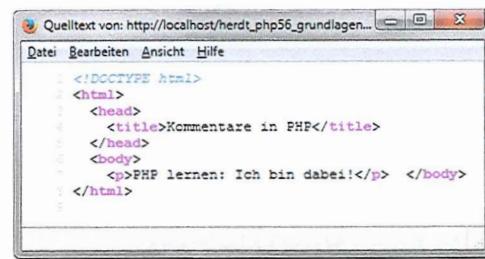
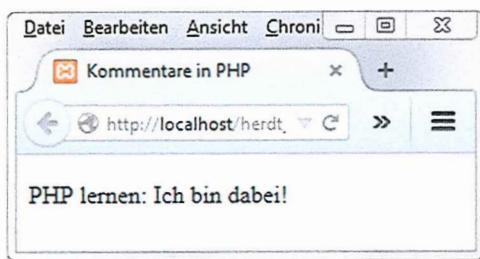
Syntax und Bedeutung der Kommentare

Beispiel: *komentar.php*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Kommentare in PHP</title>
  </head>
  <body>
    <?php
①      echo "<p>PHP lernen: " // einzeiliger Kommentar am Ende eines PHP-Codes
②      // einzeiliger Kommentar (komplette Zeile)
③      # einzeiliger Kommentar per Raute

④      /*
        mehrzeilige Kommentare werden häufig verwendet,
        um den Quellcode ausführlicher zu beschreiben
      */
⑤      echo "Ich bin dabei!</p>";
    ?>
  </body>
</html>
```

- ①+② Einzeilige Kommentare beginnen mit den Zeichen `/*` und benötigen kein abschließendes Zeichen, um dem PHP-Interpreter zu signalisieren, dass der Kommentar beendet ist. Sie werden dazu verwendet, Erläuterungen ① direkt hinter einem Befehl zu notieren oder auch ② eine gesamte Zeile als Kommentar zu kennzeichnen.
- ② Auch das Raute-Zeichen `#` kann für einzeilige Kommentare verwendet werden. Zwischen `/*` und `#` besteht für PHP kein Unterschied.
- ④+⑤ Mit den Zeichen `/*` werden mehrzeilige Kommentare eingeleitet ④ (sogenannte Kommentarblöcke). Sie ermöglichen eine ausführliche Erklärung des Quellcodes. Mit den Zeichen `*/` werden diese Kommentare beendet ⑤.



Kommentare im PHP-Code sind bei der Ausgabe im Browser herausgefiltert, auch im HTML-Quelltext.

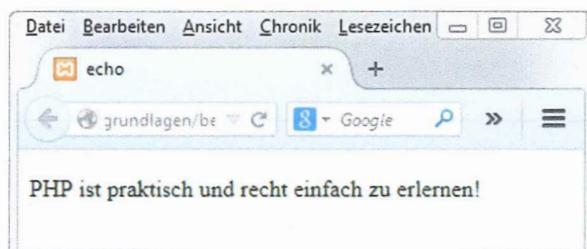
3.3 Daten im Browser ausgeben

Mit dem Befehl echo Daten im Browser anzeigen

Da PHP eine eingebettete Skriptsprache ist, die vom Server ausgeführt wird, bleibt PHP-Quellcode im Gegensatz zu HTML, JavaScript und CSS für den Betrachter unsichtbar. PHP kann jedoch Ausgaben erzeugen. Zeichen und mehrere Zeichen hintereinander (sogenannte Zeichenketten) werden über den Befehl echo in die HTML-Ergebnisdatei geschrieben und damit auf dem Bildschirm ausgegeben.

Syntax der echo-Anweisung

```
echo "<p>PHP ist praktisch und recht einfach zu erlernen!</p>";
```



Anzeige der Beispieldatei "echo.php"

Umgang mit HTML-Syntax bei der Ausgabe

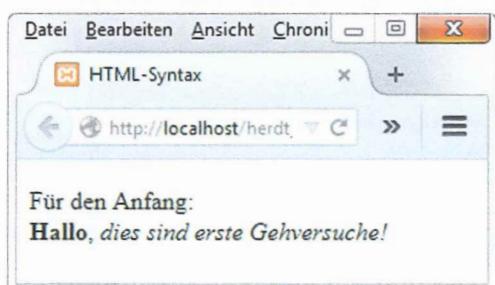
Die echo-Anweisung kann auch HTML-Tags enthalten.

- Schreiben Sie beliebige HTML-Tags einfach in die Anführungszeichen des Befehls echo.

Zeichenketten in Anführungszeichen – und dazu zählen auch HTML-Tags – werden an den Browser übergeben und vom Browser als normaler HTML-Code dargestellt.

Beispiel: html1.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML-Syntax</title>
  </head>
  <body>
    <?php
      echo "<p>Für den Anfang: <br>";
      echo "<strong>Hallo</strong>, <em>dies sind erste Gehversuche!</em></p>";
    ?>
  </body>
</html>
```



Ausgabe der Beispieldatei "html1.php"

Beispiel: *html2.php*

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML-Syntax</title>
  </head>
  <body>
    <?php
      echo '<p>Für den Anfang: <br>';
      echo '<strong>Hallo</strong>, <em>dies sind erste Gehversuche!</em></p>';
    ?>
  </body>
</html>
```

Der Quelltext *html2.php* ist fast der gleiche wie im Beispiel *html1.php*. Er unterscheidet sich dadurch, dass hier die Zeichenketten mit einfachen Anführungszeichen (Hochkommata) umschlossen sind. Beide Schreibweisen sind gültig, beide PHP-Skripte erzeugen die gleiche Ausgabe.

Der Unterschied von Zeichenketten in einfachen und doppelten Anführungszeichen ist relevant für den PHP-Interpreter und wie er mit diesen umgeht. Zeichenketten in doppelten Anführungszeichen werden geparsst (ausgewertet), sprich: der PHP-Interpreter „schaudt“ in die Zeichenkette rein, findet er PHP-Variablen oder Steuerungszeichen, werden diese interpretiert. Zeichenketten in einfachen Anführungszeichen werden hingegen nicht ausgewertet. Variablen und Steuerungszeichen werden nicht interpretiert und direkt ausgegeben. Da das Parsen von Zeichenketten in einfachen Anführungszeichen wegfällt, wird diese Variante von PHP schneller verarbeitet. Welche Variante Sie einsetzen, hängt vom Einzelfall ab.



HTML-Code selber gilt als normaler Text und wird so ausgegeben, wie er in der Zeichenkette steht.

Fehler durch Anführungszeichen

Sind innerhalb einer Zeichenkette in doppelten Anführungszeichen weitere doppelte Anführungszeichen enthalten, beendet das erste doppelte Anführungszeichen innerhalb der Zeichenkette die Zeichenkette. Danach erwartet PHP weiteren PHP-Code, findet jedoch den Rest der Zeichenkette, was zu einem Fehler führt. Das gleiche gilt entsprechend für den Einsatz von einfachen Anführungszeichen.

Einen häufigen Fehler sehen Sie im folgenden – FALSCHEN – Beispiel:

Dieses Anführungszeichen innerhalb der Zeichenkette beendet die Zeichenkette.

```
echo "Der "Eiffelturm" ist das Wahrzeichen von Paris."; //FALSCH!
```

Das Beispiel funktioniert nicht wie erwartet. Die Zeichenkette wird durch das Anführungszeichen vor dem Wort *Eiffelturm* beendet. Danach folgt bis zum Zeilenende normaler Text, den PHP nicht auswerten kann. Sie erhalten eine Fehlermeldung, die Sie auf einen Syntaxfehler aufmerksam macht.

Anführungszeichen ausgeben

Um dem Problem der Anführungszeichen zu begegnen, sieht PHP sogenannte **Escape-Sequenzen** vor. Das wichtigste Escape-Zeichen ist der Backslash \, dieser **entwertet** Zeichen, die für die PHP-Syntax relevant sind:

Beispiel: *sonderzeichen1.php*

```
echo "<p>Der \"Eiffelturm\" ist das Wahrzeichen von Paris.</p>";
echo "<p>Der 'Eiffelturm' ist das Wahrzeichen von Paris.</p>";
echo '<p>Der \'Eiffelturm\' ist das Wahrzeichen von Paris.</p>';
echo '<p>Der "Eiffelturm" ist das Wahrzeichen von Paris.</p>';
```

Escape-Sequenzen sind Zeichenkombinationen für Sonderzeichen, z. B. Anführungszeichen, Steuerzeichen oder Zeilenumbruch. Im obigen Beispiel verliert das Anführungszeichen durch den vorangestellten Backslash `\` seine Funktion als Begrenzer der Zeichenkette. Es wird nur das Anführungszeichen selbst übergeben und im Ergebnis dargestellt. Die nebenstehende Abbildung zeigt die Ausgabe der Datei `sonderzeichen1.php`.

```
Der 'Eiffelturm' ist das Wahrzeichen von Paris.
```

Beispiele für Escape-Sequenzen sind:

Escape-Sequenz	Ergebnis in der Zeichenkette	Hinweis
<code>\"</code>	"	Nur relevant, wenn die Zeichenkette von doppelten Anführungszeichen umschlossen ist
<code>\'</code>	'	Nur relevant, wenn die Zeichenkette von einfachen Anführungszeichen umschlossen ist
<code>\\"</code>	\	Nur relevant, wenn die Zeichenkette von doppelten Anführungszeichen umschlossen ist

Escape-Sequenz	Ergebnis in der Zeichenkette
<code>\n</code>	Zeilenumbruch, der nur im HTML-Quellcode der Ergebnisseite zu sehen ist. Dies dient der Übersichtlichkeit des Quellcodes, z. B. für eine Fehlersuche, da standardmäßig die gesamte Ausgabe eines PHP-Blocks nur eine Zeile im HTML-Code der Ergebnisseite einnimmt. Im Browser wird das <code>\n</code> als ein Leerzeichen angezeigt. Mehrere Leerzeichen und Zeilenumbrüche durch Steuerungszeichen im HTML-Code werden in der Browserdarstellung zu einem Leerzeichen zusammengefasst.



Sogenannte Steuerungszeichen wie `\n` für einen Zeilenumbruch im HTML-Quellcode werden nur innerhalb von doppelten Anführungszeichen vom PHP-Interpreter erkannt und ausgewertet. Steuerungszeichen in einfachen Anführungszeichen werden vom Browser als solche angezeigt und verlieren ihre beabsichtigte Funktion. Die nebenstehende Abbildung (Ausgabe der Beispieldatei `sonderzeichen2.php`) zeigt, dass Steuerungszeichen in einfachen Anführungszeichen im Browser ausgegeben werden.

```
Der 'Eiffelturm' ist das Wahrzeichen von Paris.
```

\n

Beispiel: sonderzeichen3.php

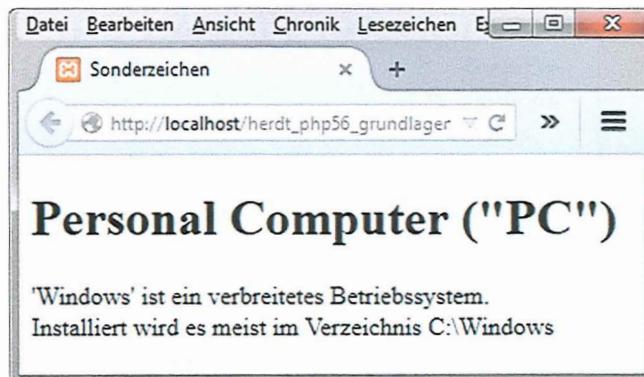
```

<!DOCTYPE html>
<html>
  <head>
    <title>Sonderzeichen</title>
  </head>
  <body>
    <?php
①      echo "\n<h1>Personal Computer ("PC")</h1>\n";
②      echo "<p>'Windows' ist ein verbreitetes Betriebssystem.<br>\n";
③      echo "Installiert wird es meist im Verzeichnis C:\\Windows</p>\n";
    ?>
  </body>
</html>

```

- ① Wenn Sie den Ausgabebefehl echo mit doppelten Anführungszeichen verwenden, setzen Sie innerhalb der Zeichenkette einen Backslash \ vor die Anführungszeichen, damit das Zeichen für PHP **entwertet** und wie beabsichtigt angezeigt wird.
- ② Alternativ können Sie auch einfache Anführungszeichen nutzen, um Text hervorzuheben. Wird die Zeichenkette durch doppelte Anführungszeichen begrenzt, beendet ein einfaches Anführungszeichen die Zeichenkette nicht. Die gilt entsprechend umgekehrt, wenn die Zeichenkette mit einfachen Anführungszeichen umschlossen ist und Sie doppelte Anführungszeichen in der Zeichenkette verwenden.
- ③ Wenn Sie einen Backslash ausgeben möchten, setzen Sie einen zweiten Backslash davor.

Die nebenstehende Abbildung zeigt, wie übersichtlich der Quelltext der Datei ist. Dies kann Ihnen – vor allem bei längeren Skripten – bei der Fehlersuche helfen. Ohne Einsatz der im Beispiel verwendeten Escape-Sequenz \n würden Sie alle Ausgaben in einer Zeile finden.



Ausgabe der Beispieldatei "sonderzeichen3.php"

```

<!DOCTYPE html>
<html>
  <head>
    <title>Sonderzeichen</title>
  </head>
  <body>
    <h1>Personal Computer ("PC")</h1>
    <p>'Windows' ist ein verbreitetes Betriebssystem.<br>
    Installiert wird es meist im Verzeichnis C:\\Windows</p>
  </body>
</html>

```

Quelltext der Beispieldatei "sonderzeichen3.php"

Deutsche Umlaute und ß – UTF-8-Zeichensatz

Seit PHP 5.4 ist in der *php.ini* der Standardzeichensatz UTF-8 konfiguriert. Dieser Zeichensatz umfasst alle deutschen Umlaute. Das bedeutet, Sie können ohne weiteres ä, ö, ü und ß einsetzen.

In älteren PHP-Versionen wird der Standardzeichensatz ISO 8859-1 verwendet. Je nach Angabe im charset-Metatag des HTML-Dokuments kann es zu Problemen bei der Darstellung von Umlauten kommen. Um eine fehlerhafte Darstellung zu vermeiden, können Sie statt der Umlaute und des ß-Zeichens die entsprechenden HTML-Entities (ä, &uuml usw.) verwenden.





Achten Sie darauf, dass die PHP-Dateien selbst mit dem UTF-8-Zeichensatz angelegt sind. Diese Einstellung kann im Editor Notepad++ über die obere Menüleiste vorgenommen werden.

Ist die PHP-Datei selbst mit einem anderen Zeichensatz erstellt, kommt es bei Umlauten oft zu Darstellungsproblemen. Die folgenden Quelltextbeispiele zeigen einerseits die Problematik, aber auch Lösungen dafür. Generell gilt: Solange Sie die PHP-Dateien mit dem UTF-8-Zeichensatz angelegt haben und die Einstellungen von PHP in der Standardeinstellung belassen, ist beim Einsatz von Umlauten und dem ß keine Sonderbehandlung notwendig.

Kodierung Sprachen Einstellungen Makro
ANSI
UTF-8 ohne BOM
UTF-8
UCS-2 Big Endian
UCS-2 Little Endian
Zeichensatz

Richtige Einstellung des UTF-8-Zeichensatzes der PHP-Dateien

Beispiel: zeichensatz1.php (falsche Kodierung)

```
<!DOCTYPE html>
<html>
    <head>
        <title>falscher Zeichensatz 1</title>
    </head>
    <body>
        <?php
            ① echo "<p>Viel Spaß im Frühling!<br>\n";
            ② echo htmlentities("Viel Spaß im Frühling!", ENT_QUOTES, "ISO-8859-1");
            echo "\n</p>";
        ?>
    </body>
</html>
```

- ① Die Datei `zeichensatz1.php` ist mit einem falschen Zeichensatz angelegt. Hier wird eine Zeile mit Sonderzeichen ausgegeben. Im folgenden Screenshot ist die Auswirkung in der ersten Zeile zu sehen, die Ausgabe ist fehlerhaft.
- ② Über die PHP-Funktion `htmlentities()` werden Umlaute und das ß in sogenannte HTML-Entities umgewandelt. Das sind Zeichen, die jeder Browser kennt und entsprechend richtig - und das unabhängig von einem Zeichensatz - anzeigt.



Mit PHP 5.6 wurde der Standardwert für den 3. Parameter der Funktion `htmlentities`, welcher für die Zeichenkodierung verantwortlich ist, an den `default_charset` angepasst (vgl. Installationshinweise im Download). Je nach Zeichensatz der PHP-Datei wandelt `htmlentities` Sonderzeichen gar nicht oder falsch um. php.net empfiehlt, den Zeichenkodierungsparameter explizit anzugeben. PHP unterstützt unterschiedliche Zeichensätze. Für unbekannte Zeichensätze wird ISO-8859-1 verwendet.



Fehlerhafte Anzeige von Sonderzeichen.



Rechts im Quelltext sind in der Zeile 8 die Sonderzeichen als HTML-Entities zu sehen.

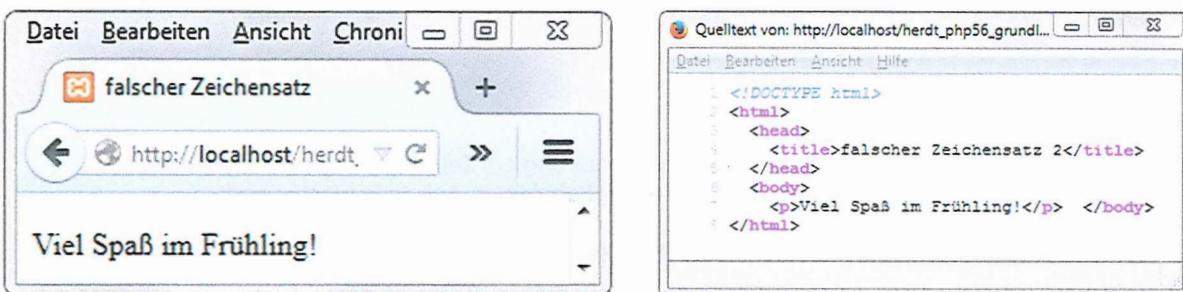
Beispiel: zeichensatz2.php

```

① <!DOCTYPE html>
<html>
<head>
<title>falscher Zeichensatz 2</title>
</head>
<body>
<?php
echo utf8_encode("<p>Viel Spaß im Frühling!</p>") ;
?>
</body>
</html>
```

Auch die PHP-Datei *zeichensatz2.php* ist mit dem falschen Zeichensatz angelegt.

- ① Über die Funktion `utf8_encode()` werden Nicht-UTF-8-Zeichen in UTF-8-Zeichen umgewandelt. Damit wird die korrekte Ausgabe des Satzes mit Umlauten und ß sichergestellt. In der untenstehenden Abbildung links ist die fehlerfreie Darstellung zu sehen. In der Abbildung rechts können Sie sehen, dass die Umlaute und das ß auch im Quelltext richtig angezeigt werden.



Die Zeichensatzproblematik tritt häufig dann auf, wenn Daten aus anderen Quellen (z. B. aus Datenbanken) stammen, auf deren Zeichensatz Sie keinen Einfluss haben. In dem Fall haben Sie mit diesen beiden Beispielen Lösungen für das Problem an der Hand. Röhrt das Problem aus einem falschen Zeichensatz einer Datei, ist die Umwandlung der Datei in eine UTF-8-Datei zu empfehlen. Dies erspart Ihnen umständliches Umwandeln von Zeichenketten.



Alle Beispieldateien, die im Buch verwendet werden und als Download bereitstehen (mit Ausnahme der beiden zuletzt vorgestellten) sind mit dem Zeichensatz UTF-8 erstellt. Das bedeutet, Sonderzeichen können als ä, ö, ü und ß im Quellcode verwendet werden, eine besondere Behandlung ist nicht notwendig.

3.4 Grundlagen zur Fehlersuche in PHP-Skripten

Fehlerarten in PHP

PHP kennt drei Kategorien von Fehlern:

Bezeichnung	Relevanz	Erläuterung
Fehler	Schwerwiegend – sofort beheben. PHP-Skripte mit Fehlern brechen das Skript an der Stelle ab oder werden erst gar nicht ausgeführt.	Am häufigsten treten schwerwiegende Fehler (<i>fatal error</i>) und Fehler bei der Analyse des PHP-Codes (<i>parse error</i>) auf. Eine Fehlermeldung wird ausgegeben. <ul style="list-style-type: none"> ✓ Bei schwerwiegenden Fehlern wird das Skript bis an diese Fehlerstelle ausgeführt. ✓ Bei einem <i>parse error</i> entdeckt der PHP-Interpreter den Fehler sofort und führt das Skript nicht aus.

Bezeichnung	Relevanz	Erläuterung
Warnung	Wichtig – zu beachten, möglichst sofort beheben. PHP-Skripte mit Warnungen werden trotzdem bis zu Ende ausgeführt.	Warnungen (<i>warning</i>) deuten meist auf schwerere Fehler hin, die sofort behoben werden sollten. Es kann passieren, dass Ihr Skript nicht mehr wie gewünscht funktioniert. Eine Fehlermeldung wird ausgegeben, das Skript wird jedoch weiter ausgeführt.
Benachrichtigung	Gut zu wissen – Behebung empfohlen. Benachrichtigungen haben keinen Einfluss auf die Ausführung von PHP-Skripten.	Benachrichtigungen (<i>notice</i>) stören selten den Ablauf des Skripts. Im Sinne einer „sauberen“ Programmierung sollten Sie allerdings die Ursache der <i>notice</i> beheben.



Bei einer PHP 5.6.x-Standardinstallation werden im Browser Fehler, Warnungen und Benachrichtigungen angezeigt. Fehler lassen sich meist schnell finden, da neben der eigentlichen Meldung sowohl der Dateiname einschließlich Dateipfad sowie die Zeilennummer ausgegeben werden, in der der Fehler aufgetreten ist. Treten Fehler auf, werden im Browser die englischen Bezeichnungen der Fehlerkategorien angezeigt:

✓ *error*

✓ *warning*

✓ *notice*

Wie können Fehler entstehen?

Fehler und Fehlerbehebung sind bei der PHP-Programmierung ein Teil des alltäglichen Entwicklungsprozesses. Häufig handelt es sich um Tipp- oder Flüchtigkeitsfehler. Beispiele hierfür sind:

- ✓ fehlendes Semikolon zum Abschluss eines Befehls;
- ✓ fehlendes \$-Zeichen bei der Verwendung von Variablen, z. B. `stadt` anstelle von `$stadt`, oder ein falsches Zeichen, z. B. `§stadt` anstelle von `$stadt`;
- ✓ sonstige Syntaxfehler, z. B. Schreibfehler wie `eco` statt `echo`;
- ✓ fehlendes Komma oder fehlende Klammer, siehe folgendes Beispiel:

Beispiel: *fehlertest.php*

	<!DOCTYPE html> <html> <head> <title>PHP-Fehlerausgabe erzwingen</title> </head> <body> <h1>Viele Fehler in einem Skript</h1> <?php ① error_reporting(E_ALL); // PHP-Standardeinstellung ② echo "<p>Der Wert der Variablen i ist: " . \$i . "</p>"; // Variable nicht definiert: notice ③ echo 4 / 0; ④ prin_r(\$i); // Befehl falsch geschrieben: fatal error ?> </body> </html>
--	--

- ① Über den Schalter `E_ALL` für die Funktion `error_reporting()` weisen Sie PHP an, Fehler aller Kategorien `error`, `warning` und auch `notices` anzuzeigen (`E_ALL` ist seit PHP 5.4 die Standardeinstellung für `error_reporting()`). Sprich: auch ohne die Zeile ① in Ihrem PHP-Code werden `error`, `warning` und `notices` angezeigt. Bei älteren PHP-Versionen müssen Sie `error_reporting(E_ALL)` explizit setzen, wenn Sie die drei Fehlerkategorien angezeigt haben möchten). Statt `error_reporting(E_ALL)` können Sie auch die Anweisung `error_reporting(-1)` notieren, die Auswirkung ist die gleiche.
- ② Durch die Verwendung der – vorher nicht definierten – Variablen `$i` erzeugen Sie eine Fehlermeldung der Kategorie `notice`. Zur Behebung eines solchen Fehlers genügt es, in den Zeilen vorher, der Variablen einen Wert zuzuweisen, z. B. `$i = 0`. Grundlagen zu Variablen in PHP finden Sie im folgenden Kapitel.
- ③ Die nicht erlaubte Division durch 0 erzeugt eine Fehlermeldung der Kategorie `warning`.
- ④ Eine falsche Schreibweise von `prin_r()` (`print_r` wäre richtig) erzeugt einen schwerwiegenden Fehler der Kategorie `fatal error`. Ein `fatal error` führt immer dazu, dass der PHP-Interpreter das Skript nicht weiter ausführt, er bricht die Verarbeitung des Skriptes an dieser Stelle sofort ab.

Viele Fehler in einem Skript

Notice: Undefined variable: i in C:\xampp\htdocs\herdt_php56_grundlagen\beispiele\kap03\fehlertest.php on line 10
Der Wert der Variablen i ist:
Warning: Division by zero in C:\xampp\htdocs\herdt_php56_grundlagen\beispiele\kap03\fehlertest.php on line 11
Fatal error: Call to undefined function prin_r() in C:\xampp\htdocs\herdt_php56_grundlagen\beispiele\kap03\fehlertest.php on line 12

Ausgabe Beispieldatei "fehlertest.php"

Hinweise zur Fehlersuche

Im Beispiel meldet PHP die jeweilige Zeile, in der ein Fehler aufgetreten ist. Zum Auffinden des Fehlers schauen Sie sich die Zeile an, die Ihnen in der Fehlermeldung mitgeteilt wird. Es kann vorkommen, dass die Fehlermeldung auf eine bestimmte Zeile hinweist, der Fehler jedoch in den Zeilen davor verursacht wurde, beispielsweise aus einer zuvor falsch erstellten Wertzuweisung. Finden Sie in der angegebenen Zeile keinen Fehler, suchen Sie in den vorhergehenden Zeilen (notfalls bis zum Dateianfang). In den meisten Fällen finden Sie jedoch den Fehler in der angegebenen Zeile.



Der Schalter `E_STRICT` ist Teil von `E_ALL`, damit werden standardmäßig auch `notice`-Meldungen ausgegeben. Über `error_reporting(E_ERROR | E_WARNING | E_PARSE)` können Sie die Fehlermeldungen auf die wichtigsten Meldungen einschränken. Alternativ können Sie über `error_reporting(E_ALL ^ E_NOTICE)` alle Meldungen um die `notice`-Meldungen reduzieren. Das `^`-Zeichen vor `E_NOTICE` bedeutet "NICHT".



Für Webseiten, die online sind, sollten Sie die Anweisung `error_reporting(0);` verwenden. Dieser PHP-Befehl mit dem Parameter `0` deaktiviert alle Fehlermeldungen. Fehlermeldungen irritieren einerseits die Webseitenbesucher, andererseits liefern sie potentiell wertvolle Informationen über Ihre Programmierung, was ein Sicherheitsrisiko darstellen kann. Auf dem Entwicklungssystem sollten hingegen möglichst alle Meldungen aktiviert sein, damit diese erkannt und behoben werden können.

3.5 Übung

Konventionen und Kommentare

Übungsdatei: --

Ergebnisdatei: **kapAVAntworten (Theorie).doc**

- ① Geben Sie an, auf welche Weise ein PHP-Block eingebunden werden kann und warum die XML-Schreibweise empfohlen wird.
- ② Nennen Sie die Gründe für die Angabe von Kommentaren und stellen Sie die Varianten gegenüber.

Grundlegende Sprachelemente

Übungsdatei: --

Ergebnisdatei: **ausgabe.php**

- ③ Schreiben Sie ein PHP-Skript, das eine Webseite erstellt. Die Webseite soll einige Informationen über Ihren Lieblingssportler bzw. -künstler enthalten. Verwenden Sie eine HTML-Überschrift wie z. B. *Gitarrenlegende Eric Clapton*.
- ④ Wechseln Sie dann in einen PHP-Block: Mithilfe des Befehls echo soll die Seite in mehreren Zeilen Informationen über für Sie interessante Eckdaten der ausgewählten Person enthalten, z. B. beste CDs, Geburtstag etc.
Bauen Sie in die Zeichenkette Sonderzeichen ein, wie z. B. Anführungszeichen. Verwenden Sie das p-Tag, welches HTML für Textblöcke (Paragraph) vorsieht. Verwenden Sie Steuerungszeichen, um den HTML-Code übersichtlicher darzustellen.
- ⑤ Kommentieren Sie Ihr PHP-Skript ausführlich.
- ⑥ Formatieren Sie einen Satz innerhalb der echo-Befehle in einer anderen Farbe und in Fettdruck.
- ⑦ Speichern Sie die Datei unter dem Namen *ausgabe.php* und rufen Sie die Seite in Ihrem Browser auf.
- ⑧ Bauen Sie im Kapitel genannte mögliche Fehler ein und prüfen Sie die auftretenden Fehlermeldungen im Browser.



Beispiel-Ergebnisdatei "ausgabe.php"

4 Variablen und Operatoren

In diesem Kapitel erfahren Sie

- ✓ was Variablen sind
- ✓ welche Datentypen in PHP genutzt werden
- ✓ wie Sie mit Operatoren arbeiten
- ✓ wie Sie Variablen ausgeben
- ✓ welche Besonderheiten bei der Verwendung von Variablen in PHP zu berücksichtigen sind
- ✓ wie Sie Konstanten definieren und mit ihnen arbeiten

Voraussetzungen

- ✓ Einbinden von PHP

4.1 Variablen

Mit Variablen arbeiten

Variablen dienen dazu, Informationen zu speichern, die für die weitere Ausführung des Programms notwendig sind. In PHP ergibt sich der Datentyp einer Variablen automatisch durch den Datentyp des Wertes, welcher der Variablen zugewiesen wird. Anders als in einigen anderen Programmiersprachen müssen Sie den Datentyp **nicht** definieren. Der Datentyp einer Variablen kann sogar im laufenden Skript wechseln, ohne dass es zu einem Fehler kommt.

Datentypen in PHP

PHP kennt folgende Datentypen:

Datentyp	Bezeichnung	Beispiel
Wahrheitswert	<i>bool(ean)</i>	true (wahr) oder false (falsch)
Ganzzahl	<i>integer</i>	42 oder -23
Fließkommazahl	<i>float (auch double)</i>	1.95883 oder -207.14
Zeichenkette	<i>string</i>	"HERDT-Verlag Bodenheim" oder 'Andreas'
ohne Wert	<i>null</i>	ohne Wert - einziger möglicher Wert: NULL
Array (ein- oder mehrdimensionale Arrays)	<i>array</i>	("Frankfurt", "Berlin", "Zürich") oder ("England" => "London", Frankreich" => "Paris")
Ressource	<i>resource</i>	Verweis auf eine Ressource, wie z. B. ein Bild
Objekt	<i>object</i>	Wird in der objektorientierten Programmierung einer Klasseninstanz zugeordnet

Namensgebung bei Variablen

Der Name einer Variablen muss in PHP immer mit dem Dollarzeichen **\$** beginnen. Daran erkennt PHP, dass es sich um eine Variable handelt. Für die Benennung von Variablen gelten folgende Regeln:

Eine Variable

- ✓ darf nur aus Buchstaben, Ziffern und – als einzigm Sonderzeichen – dem Unterstrich `_` bestehen.
- ✓ muss mit einem Buchstaben oder dem Unterstrich `_` beginnen (z. B. `$miete` oder `$_miete`). Danach kann eine beliebige Anzahl Buchstaben, Ziffern oder Unterstriche folgen.
- ✓ darf keine Leerzeichen enthalten.
- ✓ kann Groß- und Kleinbuchstaben enthalten, wobei zwischen Groß- und Kleinschreibung unterschieden wird, z. B. `$PrimZahl` ist nicht gleich `$primzahl`.
- ✓ sollte nach Möglichkeit keine Umlaute oder `ß` enthalten.
- ✓ Besteht ein Variablenname aus mehreren Begriffen, können die einzelnen Begriffe durch Unterstriche voneinander getrennt werden, z. B. `$post_versand_gebuehr`. Gebräuchlich ist auch die sogenannte CamelCase-Schreibweise, bei der jeder Begriff mit einem Großbuchstaben beginnt, z. B. `$PostVersandGebuehr`. Die Schreibweise sollte für alle Variablen einheitlich gewählt werden.
- ✓ Durch das Voranstellen eines einzelnen Buchstabens kann auf den Datentyp hingewiesen werden, wofür die Variable verwendet wird, z. B. `$iNummer` (`i` für `integer`), `$fPreis` (`f` für `float`) oder `$sOrt` (`s` für `string`).
- ✓ sollte nicht identisch sein mit einem sogenannten **reservierten Wort**.



Reservierte Wörter (PHP-Keywords)

In PHP sind bestimmte Zeichenketten bzw. Wörter als Schlüsselwörter definiert. Diese können zwar als Variablenname verwendet werden, jedoch sollten Sie dies vermeiden, da Keywords als Variablenbezeichnung PHP-Code unnötig verkomplizieren. Als Konstantennamen (vgl. Abschnitt 4.4), Funktionsnamen (vgl. Kapitel 8) oder Klassennamen sind reservierte Wörter verboten.

Liste reservierter Schlüsselwörter				
<code>__halt_compiler()</code>	<code>abstract</code>	<code>and</code>	<code>array()</code>	<code>as</code>
<code>break</code>	<code>callable</code>	<code>case</code>	<code>catch</code>	<code>class</code>
<code>clone</code>	<code>const</code>	<code>continue</code>	<code>declare</code>	<code>default</code>
<code>die()</code>	<code>do</code>	<code>echo</code>	<code>else</code>	<code>elseif</code>
<code>empty()</code>	<code>enddeclare</code>	<code>endfor</code>	<code>endforeach</code>	<code>endif</code>
<code>endswitch</code>	<code>endwhile</code>	<code>eval()</code>	<code>exit()</code>	<code>extends</code>
<code>final</code>	<code>finally</code>	<code>for</code>	<code>foreach</code>	<code>function</code>
<code>global</code>	<code>goto</code>	<code>if</code>	<code>implements</code>	<code>include</code>
<code>include_once</code>	<code>instanceof</code>	<code>insteadof</code>	<code>interface</code>	<code>isset()</code>
<code>list()</code>	<code>namespace</code>	<code>new</code>	<code>or</code>	<code>print</code>
<code>private</code>	<code>protected</code>	<code>public</code>	<code>require</code>	<code>require_once</code>
<code>return</code>	<code>static</code>	<code>switch</code>	<code>throw</code>	<code>trait</code>
<code>try</code>	<code>unset()</code>	<code>use</code>	<code>var</code>	<code>while</code>
<code>xor</code>	<code>yield</code>			

Wertzuweisung

Variablen werden in PHP durch die Angabe des Zuweisungsoperators `=` mit Werten gefüllt. PHP erfordert keine Datentyp-Definitionen. Bei der Wertzuweisung untersucht PHP den Wert und weist der Variablen automatisch einen passenden Datentyp zu. Weisen Sie innerhalb eines PHP-Skripts einer bereits definierten Variablen einen Wert eines anderen Datentyps zu, passt PHP den Datentyp automatisch an.

4.2 Variablen und Operatoren für Zahlen

Mit numerischen Datentypen arbeiten

Die numerischen Datentypen werden in Ganzzahl- und Fließkommazahl-Datentypen unterteilt. Sie werden für Berechnungen, Aufzählungen und Nummerierungen eingesetzt. Ganzzahlen, also Zahlen ohne Nachkommastellen, werden als *integer* bezeichnet. Zahlen mit Nachkommastellen (Fließkommazahl, auch Gleitkommazahl) werden als *float* (auch *double*) bezeichnet.

 In allen Beispielen ab diesem Kapitel werden die HTML-Tags `<!DOCTYPE html>`, `<html>`, `<head>`, `<title>` und `<body>` nicht mit abgedruckt. Sie sind für das Verständnis der PHP-Skripte nicht relevant. Die Beispieldateien zum Buch enthalten selbstverständlich diese Tags.

Beispiel: *preis.php*

Im folgenden Beispiel wird der Preis für den Einkauf beim Obstbauern berechnet:

```
<?php
① $preis_apfel = 2.59;
② $menge = 4;
③ $gesamtpreis = $preis_apfel * $menge;
④ echo $gesamtpreis;
?>
```

Beispieldatei "preis.php"

- ① Es wird die Variable `$preis_apfel` angelegt und der Zahlenwert `2.59` zugewiesen. `$preis_apfel` wird damit zu einer Variablen mit dem Datentyp *float* (Fließkommazahl). PHP verwendet den Punkt als Dezimaltrennzeichen (englische Notation), die deutsche Notation von Fließkommazahlen mit einem Komma ist in PHP nicht zulässig und führt zum Fehler.
- ② Die Variable `$menge` wird mit dem Zahlenwert `4` gefüllt. Die Variable `$menge` ist damit eine Variable mit dem Datentyp *integer* (Ganzzahl).
- ③ Die Variable `$gesamtpreis` wird eingeführt. Ihr wird das Ergebnis aus der Multiplikation von `$preis_apfel` und `$menge` zugewiesen. Die Variable `$gesamtpreis` wird aufgrund des Rechenergebnisses ebenfalls zu einer Variablen vom Datentyp *float* (Fließkommazahl).
- ④ Über den Befehl `echo` wird der Wert der Variablen `$gesamtpreis` ausgegeben.



Zur besseren Lesbarkeit empfiehlt sich die (optionale) Verwendung von Leerzeichen z. B. zwischen Operatoren und Variablen.

Arithmetische Operatoren

Mit arithmetischen Operatoren können mathematische Berechnungen durchgeführt werden. Sie erwarten entweder Ganzzahl- bzw. Fließkommazahl-Variablen oder feste Werte als Parameter und liefern ein numerisches Ergebnis zurück. Sie können folgende Operatoren verwenden:

Operator	Name	Bedeutung	Beispiel	Wert nach der Operation
+	Addition	<code>\$a + \$b</code> ergibt die Summe von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a + \$b;</code>	<code>\$c = 12</code>
-	Subtraktion	<code>\$a - \$b</code> ergibt die Differenz von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a - \$b;</code>	<code>\$c = 8</code>

Operator	Name	Bedeutung	Beispiel	Wert nach der Operation
*	Multiplikation	$\$a * \b ist das Produkt aus $\$a$ und $\$b$.	$\$a = 10;$ $\$b = 2;$ $\$c = \$a * \$b;$	$\$c = 20$
/	Division	$\$a / \b ist der Quotient von $\$a$ und $\$b$.	$\$a = 10;$ $\$b = 2;$ $\$c = \$a / \$b;$	$\$c = 5$
**	Potenz	$\$a ** \b ist die Potenz aus $\$a$ hoch $\$b$.	$\$a = 10;$ $\$b = 2;$ $\$c = \$a ** \$b;$	$\$c = 100$
%	Modulo	$\$a \% \b ist der Rest der ganz-zahligen Division von $\$a$ und $\$b$.	$\$a = 10;$ $\$b = 3;$ $\$c = \$a \% \$b;$	$\$c = 1$
++	Präinkrement	$++\$a$ erhöht die Variable $\$a$ um 1 vor der weiteren Verwendung.	$\$a = 10;$ $\$b = 2;$ $\$c = ++\$a + \$b;$	$\$a = 11$ $\$c = 13$
--	Prädekrement	$--\$a$ verringert die Variable $\$a$ um 1 vor der weiteren Verwendung.	$\$a = 10;$ $\$b = 2;$ $\$c = --\$a + \$b;$	$\$a = 9$ $\$c = 11$
++	Postinkrement	$\$a++$ erhöht die Variable $\$a$ um 1 nach der Verwendung.	$\$a = 10;$ $\$b = 2;$ $\$c = \$a++ + \$b;$	$\$a = 11$ $\$c = 12$
--	Postdecrement	$\$a--$ verringert die Variable $\$a$ um 1 nach der Verwendung.	$\$a = 10;$ $\$b = 2;$ $\$c = \$a-- + \$b;$	$\$a = 9$ $\$c = 12$
+=	Zuweisungsoperator	$\$a += \b weist der Variablen $\$a$ den Wert $\$a + \b zu (Kurzschreibweise für $\$a = \$a + \$b$).	$\$a = 10;$ $\$a += 5;$	$\$a = 15$
-=;	Zuweisungsoperator	$\$a -= \b ist die Kurzschreibweise für $\$a = \$a - \$b$.	$\$a = 10;$ $\$a -= 5;$	$\$a = 5$
*=	Zuweisungsoperator	$\$a *= \b ist die Kurzschreibweise für $\$a = \$a * \$b$.	$\$a = 10;$ $\$a *= 5;$	$\$a = 50$
/=	Zuweisungsoperator	$\$a /= \b ist die Kurzschreibweise für $\$a = \$a / \$b$.	$\$a = 10;$ $\$a /= 5;$	$\$a = 2$

Werden mehrere Berechnungen in einer Zuweisung durchgeführt (z. B. $\$a = \$b - \$c * \d), werden die üblichen mathematischen Rechenregeln angewandt:

- ✓ Bearbeitung der Berechnung von links nach rechts
- ✓ Punkt- vor Strichrechnung
- ✓ Geklammerte Ausdrücke werden zuerst ausgewertet.

Der Potenz-Operator ****** ist neu in PHP 5.6. $\$a ** \b ist die Potenz aus $\$a$ hoch $\$b$. $\$a$ vor dem ****** ist dabei die Basis (Grundzahl), $\$b$ hinter dem ****** ist der Exponent (Hochzahl).

Beispiel: berechnung.php

```

<?php
① $preis_apfel = 2.59;
$menge_jonagold = 4;
$menge_idared = 10;
$menge_elstar = 15;
② $gesamtmenge = $menge_jonagold + $menge_idared + $menge_elstar;
③ $gesamtpreis = $gesamtmenge * $preis_apfel;
④ echo $gesamtpreis;
?>

```

- ① Den Variablen \$preis_apfel, \$menge_jonagold, \$menge_idared und \$menge_elstar werden Werte zugewiesen. Verwenden Sie möglichst aussagekräftige Variablenbezeichnungen. Dies erhöht die Nachvollziehbarkeit Ihrer Skripte.
- ② Die Addition der Variablen \$menge_jonagold, \$menge_idared und \$menge_elstar ergibt 29. Dieser Wert wird der neuen Variablen \$gesamtmenge zugewiesen und somit zwischengespeichert.
- ③ Die Variable \$gesamtmenge wird mit dem Preis \$preis_apfel multipliziert (ergibt 75.11) und der Variablen \$gesamtpreis zugewiesen. Mehrere Berechnungen, wie hier die Addition ② und die Multiplikation ③, können Sie in mehrere Schritte aufteilen. Das macht den PHP-Code verständlicher.
- ④ Abschließend wird der Wert der Variablen \$gesamtpreis per echo im Browser ausgegeben.

4.3 Variablen und Operatoren für Zeichenketten

Mit Zeichen-Datentyp (*string*) arbeiten

Der Zeichen-Datentyp kann beliebige Zeichen des erweiterten ASCII-Zeichensatzes enthalten. Der in PHP verwendete Zeichen-Datentyp ist die Zeichenkette, auch *string* genannt. Zeichenketten werden bei der Wertzuweisung in Anführungszeichen bzw. Hochkommata eingeschlossen.

Zeichenkettenoperator

Sie können mehrere Zeichenketten oder Zahlen (Ganz- oder Fließkommazahlen) und Zeichenketten über den Zeichenkettenoperator miteinander verknüpfen. Der Fachbegriff für die Zeichenkettenverknüpfung ist **Konkatenation**. Das Ergebnis einer Konkatenation ist immer ein Wert vom Datentyp *string*.

Operator	Bedeutung	Beispiel
.	Verketten von Zeichenketten	\$a = "Göttingen ist "; \$b = "eine schöne"; \$c = \$a . \$b . " Stadt."; echo \$c;
.=	Anhängen einer Zeichenkette an eine bereits vorhandene Variable	\$a = "Göttingen ist"; \$a .= " eine schöne"; \$a .= " Stadt."; echo \$a;

In beiden Beispielen erhalten Sie als Ausgabe *Göttingen ist eine schöne Stadt*. Sie sehen auch, dass Sie sich innerhalb der Zeichenketten selbst um die Leerzeichen am Übergang der einzelnen Zeichenketten kümmern müssen.

 Gemäß mathematischen Rechenregeln hat die Konkatenation die gleiche Priorität wie Addition und Subtraktion. Hier werden die Operationen von links nach rechts durchgeführt. Die Konkatenation hat aber eine geringere Priorität als Multiplikation und Division, hier wird die Konkatenation erst danach durchgeführt, unabhängig davon, in welcher Reihenfolge die Operationen stehen.

```
<?php
$b = 3;
$c = 5;
① $a = "Zahl " . $b * $c;
② $a = "Zahl " . $b + $c;
③ $a = $b + $c . " Zahl";
④ $a = "Zahl " . ($b + $c);
?>
```

- ① \$a erhält den Wert Zahl 15 - zuerst wird die Multiplikation durchgeführt, danach die Konkatenation (Verknüpfung) der Zeichenketten.
- ② Hier werden die Operationen von links nach rechts durchgeführt. Im ersten Schritt wird über eine Konkatenation der *string*-Wert Zahl mit dem *integer*-Wert 3 zu der Zeichenkette Zahl 3 verknüpft. Zu dieser Zeichenkette soll der Wert der Variablen \$c (= 5) addiert werden. Damit diese Addition durchgeführt werden kann, wandelt PHP den Datentyp von Zahl 3 (*string*) durch die automatische Typkonvertierung in den Datentyp *integer* um. Aus Zahl 3 (*string*) wird 0 (*integer*). Somit werden 0 und 5 addiert. Der Wert von \$a ist 5.
- ③ Wird die Addition vor der Zeichenkettenverknüpfung durchgeführt, wird das Ergebnis der Addition mit der Zeichenkette verbunden (konkateniert). \$a hat anschließend den Wert 8 Zahl.
- ④ Soll eine Addition oder Subtraktion vor einer Zeichenkettenverknüpfung durchgeführt werden, müssen entsprechend Klammern gesetzt werden. Hier wird \$a der Wert Zahl 8 zugewiesen.

Ausgabe von Variablen

Bei der Ausgabe einer Variablen kann deren Wert oder deren Bezeichnung ausgegeben werden. Ob der Wert oder der Variablennamen angezeigt wird, hängt von der Einbettung der Variable im PHP-Code ab.

Beispiel: var_ausgabe.php

Hier werden die verschiedenen Ausgabemöglichkeiten aufgezeigt.

```
<?php
$stadt = "Göttingen";
$uni = "Georg-August-Universität";
$jahr = 1736;
$heute = 2012;
// Wert der Variablen werden angezeigt
② echo "<p>$jahr wurde die $uni in $stadt gegründet.</p>";
// Name der Variablen werden angezeigt
③ echo '<p>$jahr wurde die $uni in $stadt gegründet.</p>';
// Ausgabe von Berechnungen mit Variablen sowie Zeichenketten
④ echo "<p>Die Gründung der $uni in $stadt erfolgte vor " . ($heute - $jahr) .
    " Jahren.</p>";
?>
```

- ① Den Variablen \$stadt, \$uni, \$jahr und \$heute werden Zeichenketten- bzw. Ganzahlenwerte zugewiesen.
- ② Zeichenketten in **doppelten Anführungszeichen** werden von PHP geparsst (ausgewertet). Die Variablen werden erkannt, ausgelesen und angezeigt.
- ③ Zeichenketten in **Hochkommata (einfache Anführungszeichen)** werden von PHP **nicht** geparsst. Hier werden die Werte der Variablen nicht ausgewertet, sondern die Variablennamen ausgegeben.
- ④ Wenn Sie eine Ausgabe mit dem Ergebnis einer Berechnung verknüpfen möchten, müssen Sie die Berechnung außerhalb der Zeichenketten vornehmen und diese in Klammern setzen. Innerhalb der Anführungszeichen werden Operatoren nur als einfache Zeichen einer Zeichenkette verstanden.



Ausgabe der Beispieldatei "var_ausgabe.php"



Das \$-Zeichen, das als Erkennungszeichen der Variablen dient, wird von PHP innerhalb von Zeichenketten in doppelten Anführungszeichen erkannt und interpretiert. Soll das \$-Zeichen als Teil der Zeichenkette angezeigt werden, muss dem \$-Zeichen ein Backslash \ vorangestellt werden. Dadurch wird das \$-Zeichen vor PHP **versteckt**. PHP interpretiert dann das \$-Zeichen nicht, es wird deshalb als \$ im Browser angezeigt. Das Voranstellen eines Backslashes \ wird als **escapen** bezeichnet.

Beispiel: zeichenkette.php

Nachfolgend werden Variablen auf verschiedene Arten mit Werten gefüllt und über den Befehl echo im Browser ausgegeben.

```
<?php
/*
Peter hat zu Hause noch amerikanische Dollar (USD) gefunden.
Welchen Wert (in Euro) hat sein Fund?
*/
① $dollar = 1240.45;
$kurs = 1.25; // Umrechnungskurs Dollar-Euro
$euro = $dollar / $kurs;
② $bezeichnung_dollar = "US Dollar (USD)";
$bezeichnung_euro = "EURO";

③ $ausgabe = "<p>Peter sagt: 'Meine " . $dollar . " " . $bezeichnung_dollar;
④ $ausgabe .= " sind " . $euro . " " . $bezeichnung_euro . " wert.'</p>";
⑤ echo $ausgabe;
⑥ echo "<p>Peter sagt: 'Meine $dollar $bezeichnung_dollar sind $euro
      $bezeichnung_euro wert.'</p>";
⑦ echo '<p>Peter sagt: \'Meine $dollar $bezeichnung_dollar sind $euro
      $bezeichnung_euro wert.\'</p>';
?>
```

- ① Den Variablen \$dollar und \$kurs werden Werte zugewiesen und der Wert der Variablen \$euro wird berechnet.
- ② Den Variablen \$bezeichnung_dollar und \$bezeichnung_euro werden Zeichenketten zugewiesen. Verwenden Sie sprechende Variablenbezeichnungen. Umso besser der Name der Variablen deren Aufgabe im PHP-Code beschreibt, umso einfacher ist ein PHP-Skript (für Sie und andere Entwickler) zu „lesen“. Siehe Zeilen ⑥ und ⑦.
- ③ Mithilfe des Operators . wird der Variablen \$ausgabe eine Zeichenkette zugewiesen. Hierfür werden einzelne Zeichenketten, Variablen und HTML-Code miteinander verbunden. Bei der Verkettung der Zeichenketten müssen Sie selbst für die Leerzeichen zwischen den Wörtern sorgen.

- ④ Mithilfe des Operators `.=` wird die Zeichenkette \$ausgabe verlängert. Diese Vorgehensweise empfiehlt sich, um den Code übersichtlicher und gut lesbar zu gestalten.
- ⑤ Es erfolgt die Ausgabe der Variablen \$ausgabe mithilfe des Befehls echo.
- ⑥ Sie können die gleiche Ausgabe auch direkt über eine Zeichenkette erreichen. Die Variablenwerte der angegebenen Variablen werden ausgegeben, wenn Sie die Zeichenkette durch doppelte Anführungszeichen begrenzen.
- ⑦ Steht hingegen eine Variable innerhalb einer durch Hochkommata begrenzten Zeichenkette, wird der Name der Variablen ausgegeben und nicht ihr Wert. Beachten Sie auch die Notwendigkeit, den Hochkommata innerhalb der Zeichenkette ein `,` voranzustellen. Fehlen diese Zeichen, erhalten Sie eine Fehlermeldung, da das zweite bzw. nachfolgende Hochkomma die Zeichenkette beenden würde.

Ausgabe der Beispieldatei "zeichenkette.php"

Variablen in PHP: Alles ganz einfach – oder?

Variablen in PHP zu verwenden ist, gemessen an anderen Programmiersprachen wie z. B. JAVA, vergleichsweise einfach. Eine Variablendeclaration zur Festlegung des Datentyps ist nicht notwendig. Es reicht, einer Variablen einen Wert zuzuweisen, PHP weist automatisch anhand des Wertes den richtigen Datentyp zu. Damit ist PHP wesentlich fehlertoleranter als andere Programmiersprachen.

Dieser einfache Umgang mit Variablen bringt jedoch einige Besonderheiten mit sich:

- ✓ Variablen können in Zeichenketten verwendet werden, z. B. zur Ausgabe durch den echo-Befehl. Ist die Zeichenkette durch doppelte Anführungszeichen begrenzt, werden die Werte der Variablen ausgelesen und zurückgeliefert. In anderen Programmiersprachen müssen häufig Variablen und Zeichenketten getrennt voneinander notiert und gegebenenfalls miteinander verkettet werden.
- ✓ Eine Variable in PHP kann während des Programmablaufs den Datentyp ändern. Dies kann entweder von Ihnen explizit so programmiert sein oder der Datentyp wird automatisch durch eine Berechnung geändert. Dieses Verhalten ist ausdrücklich erwünscht. Eine Fehlermeldung – wie sie viele Programmierer anderer Programmiersprachen erwarten – wird daher nicht ausgegeben.
- ✓ PHP erlaubt es, Rechenoperationen mit Zeichenketten durchzuführen. Rechenoperationen von Zahlenwerten mit Zeichenketten führen in PHP **nicht** zu einer Fehlermeldung. Dabei ist zu beachten:
 - ✓ Führende Leerzeichen in der Zeichenkette werden ignoriert.
 - ✓ Beginnt eine Zeichenkette mit einer Zahl - auch nach führenden Leerzeichen - wird die Zahl extrahiert und für Berechnungen verwendet. Beispielsweise ergibt "10 graue Mäuse" den Zahlwert 10, "30.7 ABC 99.3" ergibt 30.7, "4AD" ergibt 4.
 - ✓ Beginnt eine Zeichenkette – auch nach führenden Leerzeichen – nicht mit einer Zahl, ergibt sich automatisch ein Zahlwert von 0, der für Rechenoperationen verwendet wird.

Beispiel: var_verhalten.php

Im nachfolgenden Beispiel sehen Sie die beschriebenen Besonderheiten bei der Verwendung von Variablen in PHP:

```

<body>
<?php
① $test = "10";           // String
echo $test . " (" . gettype($test) . ")<br>";

② $test = $test * 2;      // Integer (20)
echo $test . " (" . gettype($test) . ")<br>";

③ $test = $test + 1.75;   // Fließkommazahl (21.75)
echo $test . " (" . gettype($test) . ")<br>";

④ $test = 5 + "10 Tassen Tee"; // Integer (15)
echo $test . " (" . gettype($test) . ")<br>";

⑤ $test = $test + "Kaffeetassen: 530"; // Integer (bleibt 15)
echo $test . " (" . gettype($test) . ")<br>";
?>
</body>
```

- ① Der Variablen \$test wird der Wert 10 zugewiesen, allerdings in Anführungszeichen. Damit handelt es sich um eine Zeichenkette, die Variable hat also den Datentyp *string*. Im folgenden PHP-Code wird diese Variable über verschiedene Operationen verändert. Über die Funktion `gettype()` wird der Datentyp ausgegeben, um zu prüfen, welchen Datentyp die Variable nach der Veränderung hat.
- ② Die Zeichenkettenvariable \$test wird mit der Zahl 2 multipliziert (Zahlwert $10 * 2 = 20$). Durch die Rechenoperation ermittelt PHP den Zahlwert aus einer Variablen des Datentyps *string*. In diesem Beispiel wird die Variable \$test in den Dateityp *integer* mit dem Wert 10 umgewandelt, der dann für die weitere Berechnung verwendet wird.
- ③ Durch Addition einer Fließkommazahl ändert die Variable nochmals ihren Datentyp und wird zu einer Variablen des Datentyps *float*.
- ④ Hier wird zu der Zahl 5 die Zeichenkette 10 Tassen Tee addiert. PHP wandelt die Zeichenkette 10 Tassen Tee automatisch in 10 um, um einen verwendbaren Wert für die Rechenoperation zu haben. Die Summe aus Zahl und Zeichenkette ergibt 15.
- ⑤ Die Variable \$test hat aus der vorherigen Rechenoperation den Wert 15. Durch die beabsichtigte Addition wandelt auch hier PHP die Zeichenkette Kaffeetassen: 530 in eine Zahl um. Da die Zeichenkette mit einem Buchstaben beginnt, ermittelt PHP den Wert 0. Das Ergebnis der Addition ist 15 ($15 + 0 = 15$).



Sie können jederzeit abfragen, welchen Datentyp eine bestimmte Variable aufweist und entsprechend darauf reagieren. Zur Abfrage des Datentyps können Sie die Funktion `gettype(<Variablename>)` verwenden, z. B.:

```
$variable = 39;
echo gettype($variable); // gibt in diesem Fall "integer" aus
```

4.4 Konstanten

Variablen können – wie der Name schon sagt – variable Werte haben. Über Operatoren können Variablen im laufenden PHP-Skript verändert werden und neue Werte annehmen. Es gibt allerdings auch Werte, deren Wertänderung durch das Skript nicht gewollt ist oder deren Änderung keinen Sinn macht, wie z. B. die feststehende Länge einer Marathonstrecke oder der Kreiskonstanten Pi. Für diesen Zweck sieht PHP Konstanten vor.

Konstanten sind Variablen ähnlich. Der Unterschied besteht darin, dass Konstanten einmalig bei ihrer Definition ein Wert zugewiesen wird, der danach nicht mehr verändert werden kann.

Folgende Merkmale unterscheiden Konstanten von Variablen:

- ✓ Konstanten haben kein vorangestelltes \$-Zeichen im Bezeichner.
- ✓ Konstanten werden über die Funktion `define()` definiert, nicht durch eine einfache Zuweisung wie bei Variablen.
- ✓ Alternativ können Sie Konstanten über das Schlüsselwort `const` definieren. Dabei wird der Wert mit dem `=` Operator wie bei normalen Variablen zugewiesen.
- ✓ Konstanten können skalare Datenwerte, Arrays und seit der Version PHP 5.6 auch skalare Ausdrücke enthalten. Skalarwerte sind Ganzzahlen, Fließkommazahlen, Zeichenketten oder boolesche Werte. Seit PHP 5.5 können auch Arrays als Konstante gespeichert werden, allerdings nur über das Schlüsselwort `const`. Der Versuch, ein Array als Konstante über `define()` zu initialisieren, führt zu einer Warnung (`warning`), eine Konstante wird in dem Fall nicht definiert. Objekte können nicht als Konstante gespeichert werden.
- ✓ Neu in PHP 5.6 sind **konstante skalare Ausdrücke (Constant Scalar Expressions)**. Es können feststehende Ausdrücke (Rechenoperationen, die immer gleich bleiben) als Konstante definiert werden.
- ✓ Ist eine Konstante einmal definiert, kann sie zur Laufzeit des PHP-Skripts weder gelöscht noch neu definiert werden.

Konstanten definieren

```
define ("NAME", Wert);
```

Zur Definition von Konstanten verwenden Sie die Funktion `define()`. Sie müssen zwei Argumente angeben: die Bezeichnung der Konstanten und den Wert, den Sie der Konstanten zuweisen wollen. Der Name der Konstanten muss dabei in **einfachen** oder **doppelten Anführungszeichen** stehen.

```
const NAME = Wert;
```

Alternativ können Sie Konstanten über das Schlüsselwort `const` definieren (seit PHP 5.3). Im Gegensatz zu `define()` muss beim `const` der Name der Konstanten ohne Anführungszeichen angegeben werden. Die Zuweisung der Werte geschieht über den Operator `=`.

Werte für Konstanten

Konstanten können skalare Werte zugewiesen werden (z. B. `100`), aber auch das Ergebnis einer Rechenoperation (z. B. `3 * 100`) oder Zeichenketten, die per Konkatenation zusammengefügt werden (z. B. `"Ich stehe" . " im Wald."`). Aber auch bereits definierte Konstanten können zur Definition einer neuen Konstante verwendet werden.

Seit PHP 5.6 können Sie bei `const` auch **konstante skalare Ausdrücke** verwenden. In älteren PHP-Versionen führte die Definition einer Konstanten per `const`, der das Ergebnis einer Rechenoperation oder eine Konkatenation zugewiesen werden sollte, zu einem Fehler (**parse error**). Jetzt können Sie über `const` auch Rechenausdrücke bei der Konstantendefinition verwenden, in denen Sie neben anderen skalaren Werten auch andere Konstanten einsetzen können.

Schreiben Sie die Konstanten komplett in Großbuchstaben, dies entspricht gängigen Konventionen. Unabhängig davon, ob Sie für normale Variablen die Schreibweise mit Unterstrichen oder den CamelCase verwenden: Konstanten in Großbuchstaben sind im PHP-Code deutlich zu erkennen, Konstanten und Variablen sind so einfacher zu unterscheiden.



Beispiel: *konstante.php*

Im nachfolgenden Beispiel wird die Definition und Verwendung von Konstanten gezeigt.

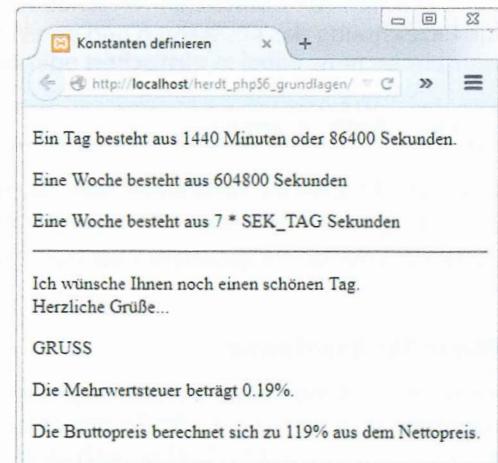
```
<?php
① define('SEK_TAG', 86400); // Anzahl der Sekunden pro Tag
② define("MIN_TAG", 24 * 60); // Anzahl der Minuten pro Tag
③ define("GRUSS", "<hr><p>Ich wünsche Ihnen noch einen schönen Tag.<br>Herzliche
    Grüße...</p>");

④ const NETTO = 100;
const MWST = 0.19;
const BRUTTO = NETTO + NETTO * MWST;

⑤ echo "<p>Ein Tag besteht aus " . MIN_TAG . " Minuten oder " . SEK_TAG . "
    Sekunden.</p>";
⑥ echo "<p>Eine Woche besteht aus " . (7 * SEK_TAG) . " Sekunden</p>";
⑦ echo "<p>Eine Woche besteht aus 7 * SEK_TAG Sekunden</p>";
⑧ echo GRUSS;
echo "GRUSS";
⑨ echo "<p>Die Mehrwertsteuer beträgt " . MWST . "%.</p>";
echo "<p>Die Bruttopreis berechnet sich zu " . BRUTTO . "% aus dem
    Nettopreis.</p>";
?>
```

Beispieldatei "konstante.php"

- ① Einer Konstanten namens `SEK_TAG` wird der Zahlenwert 86400 zugewiesen.
- ② Der Konstanten `MIN_TAG` wird ebenfalls ein Zahlenwert zugewiesen, in diesem Fall das Ergebnis einer Rechenoperation.
- ③ Einer Konstanten namens `GRUSS` wird eine Zeichenkette zugewiesen. Hierbei kann es sich durchaus um komplette HTML-Bausteine handeln.
- ④ Über `const` werden die Konstanten `NETTO`, `MWST` und `BRUTTO` definiert. Für die `BRUTTO`-Konstante wird ein *konstant skalarer Ausdruck* verwendet.
- ⑤ In dieser Zeile werden mehrere Zeichenketten und die definierten Konstanten `MIN_TAG` und `SEK_TAG` ausgegeben. Die Konstanten werden mit den Zeichenketten konkateniert.
- ⑥ Hier wird eine Rechenoperation mit der Konstanten `SEK_TAG` durchgeführt und das Ergebnis ausgegeben. Zur Abtrennung von der Zeichenkette setzen Sie die Berechnung in Klammern. Somit wird das Ergebnis zuerst berechnet und dann als Teil der Zeichenkette ausgegeben.
- ⑦ Ähnlich wie bei ⑥, nur dass hier die Konstante innerhalb der Zeichenkette (ohne Konkatenation) angegeben wurde. Im Resultat erhalten Sie nur die Ausgabe des Konstantennamens. Obwohl PHP Zeichenketten in doppelten Anführungszeichen parst, werden Konstanten in Zeichenketten nicht ausgewertet. PHP kann Zeichen und Konstanten nicht unterscheiden, zur Ausgabe einer Konstanten ist deswegen immer eine Konkatenation notwendig. Das gleiche gilt für Rechenoperationen, die in Zeichenketten notiert sind. Auch hier kann PHP nicht erkennen, ob es sich um das Zeichen `*` handelt oder den Operator `*`.
- ⑧ Es wird eine Konstante verwendet, die aus einem HTML-Block besteht. Es erfolgt die Ausgabe der definierten Zeichenkette. In der Folgezeile steht die Konstante wieder in einer Zeichenkette. Es erfolgt lediglich die Ausgabe des Konstantennamens, also die Zeichenfolge "GRUSS".
- ⑨ In dieser Zeile werden die Konstanten ausgegeben, die per `const` definiert wurden. Dort können Sie auch erkennen, dass die Konstante `BRUTTO` den berechneten Wert angenommen hat.



Ausgabe der Beispieldatei "konstante.php"

4.5 Übung

Werte von Variablen erkennen

Übungsdatei: --

Ergebnisdatei: antworten-4.php

- ① Welche Ausgabe erhalten Sie bei den nachfolgenden Codezeilen?

Die Variablen haben folgende Werte: \$a = 7, \$b = "30 Euro", \$c = "!".

- a) echo \$a . \$b . \$c;
- b) echo "Text";
- c) echo "Text" . \$a;
- d) echo "Text" \$a . \$b;
- e) echo \$a + \$b + \$c;
- f) echo \$a * \$b / \$c;
- g) echo ('\'Text\'' . \$a . " Text " . \$b);

Mit Variablen, Operatoren und Konstanten arbeiten

Übungsdatei: --

Ergebnisdatei: buero.php

- ② Erstellen Sie mit folgenden Angaben ein PHP-Skript, das Sie unter dem Namen *buero.php* speichern.

Variable	Bezeichnung	Variable	Preis (netto)
\$bezeichnung_tisch	Schreibtisch	\$preis_tisch	1999.00 €
\$bezeichnung_stuhl	Bürostuhl	\$preis_stuhl	589.00 €
\$bezeichnung_lampe	Lampe	\$preis_lampe	29.00 €
\$bezeichnung_pctisch	Computertisch	\$preis_pctisch	999.00 €

Berechnen Sie den Gesamtpreis (\$netto_gesamt) der eingekauften Artikel.

- ③ Berechnen Sie für den gerade berechneten Gesamtpreis den Bruttopreis (\$brutto_gesamt) mithilfe einer Konstanten namens MWST. Der Mehrwertsteuersatz, der zur Berechnung verwendet wird, beträgt 19 %. Die verwendete Zeichenkette für die Währung Euro stellen Sie bitte ebenfalls über eine Konstante (EURO) bereit.
- ④ Berechnen Sie zusätzlich die Bruttopreise aller Artikel.
- ⑤ Lassen Sie alle errechneten Werte in verständlicher Form mit Beschriftungen anzeigen.

```

Übung (Kapitel 4)
http://localhost/herdt_php! Google

Übung (Kapitel 4): Mit Variablen,
Operatoren und Konstanten arbeiten

Netto-Gesamtpreis der eingekauften Artikel: 3616 Euro.
Brutto-Gesamtpreis der eingekauften Artikel: 4303.04 Euro.

Brutto-Preis Schreibtisch: 2378.81 Euro.
Brutto-Preis Bürostuhl: 700.91 Euro.
Brutto-Preis Schreibtischlampe: 34.51 Euro.
Brutto-Preis Computertisch: 1188.81 Euro.

```

Lösungsvorschlag "buero.php"

5 Kontrollstrukturen

In diesem Kapitel erfahren Sie

- ✓ wie Sie bedingte Anweisungen programmieren
- ✓ welche Möglichkeiten der Fallauswahl es gibt
- ✓ wie Sie Schleifen programmieren
- ✓ wie Sie die Ausführung von Schleifen beeinflussen können

Voraussetzungen

- ✓ Grundlagen PHP

5.1 Kontrollstrukturen einsetzen

Zeilenweise Ausführung

Die Anweisungen in einem PHP-Skript werden in der Reihenfolge ausgeführt, in der sie angegeben sind, falls keine Anweisung einen Befehl enthält, der diese Reihenfolge ändert. Ein PHP-Skript wird in diesem Fall sequenziell, also Zeile für Zeile, abgearbeitet.

Warum sind Kontrollstrukturen notwendig?

Oft ist es erforderlich, dass eine oder mehrere Anweisungen ...

- ✓ nur unter bestimmten Bedingungen durchgeführt werden sollen, z. B. ein Rabatt für einen Einkauf wird nur gewährt, falls eine Mindestbestellmenge vorliegt;
- ✓ wiederholt durchgeführt werden sollen, z. B. beim Lesen aller oder gefilterter Einträge aus einer Liste.

Um von der sequenziellen Abarbeitung der Befehle abzuweichen, setzen Sie Kontrollstrukturen ein. Sie können gezielt den Ablauf des Skripts durch Bedingungen oder Schleifen steuern.

Anweisungen über Bedingungen auswählen

Eine Bedingung ist eine Möglichkeit, den Ablauf eines Skripts durch Entscheidungen zu beeinflussen. In einer Bedingung werden Ausdrücke verglichen. Das Ergebnis dieses Vergleichs kann dabei entweder mit "Ja" (TRUE) oder mit "Nein" (FALSE) beantwortet werden. Hierbei können Sie mit folgenden Operatoren arbeiten:

Vergleichsoperatoren

Diese Operatoren, auch relationale Operatoren genannt, vergleichen Ausdrücke miteinander und liefern ein logisches Ergebnis, entweder TRUE (wahr) oder FALSE (falsch).

Operator	Name	Bedeutung	Beispiel	Ergebnis
<code>==</code>	Gleich	<code>\$a == \$b</code> ergibt TRUE, wenn \$a und \$b gleich sind.	<code>\$a = 4; \$b = 4.0; \$a == \$b;</code>	TRUE
<code>!= oder <></code>	Ungleich	<code>\$a != \$b</code> ergibt TRUE, wenn \$a und \$b ungleich sind.	<code>\$a = 4; \$b = 4; \$a != \$b;</code>	FALSE

Operator	Name	Bedeutung	Beispiel	Ergebnis
==	Identisch	\$a == \$b ergibt TRUE, wenn \$a und \$b gleich und vom selben Datentyp sind.	\$a = 4; \$b = 4.0; \$a == \$b;	FALSE
!=	Nicht identisch	\$a != \$b ergibt TRUE, wenn \$a und \$b ungleich oder nicht vom selben Datentyp sind.	\$a = 4; \$b = 4.0; \$a != \$b;	TRUE
<	Kleiner	\$a < \$b ergibt TRUE, wenn \$a kleiner \$b ist.	\$a = 3; \$b = 4; \$a < \$b;	TRUE
>	Größer	\$a > \$b ergibt TRUE, wenn \$a größer \$b ist.	\$a = 3; \$b = 4; \$a > \$b;	FALSE
<=	Kleiner gleich	\$a <= \$b ergibt TRUE, wenn \$a kleiner oder gleich \$b ist.	\$a = 4; \$b = 4; \$a <= \$b;	TRUE
>=	Größer gleich	\$a >= \$b ergibt TRUE, wenn \$a größer oder gleich \$b ist.	\$a = 5; \$b = 4; \$a >= \$b;	TRUE

Im Unterschied zum **Gleichheitsoperator** (zwei Gleichheitszeichen) wird beim **Identisch-Operator** (drei Gleichheitszeichen) zusätzlich der Datentyp verglichen. Wird beispielsweise die Zahl 1 mit der Zeichenkette "1" per verglichen, ist die Überprüfung wahr. Werden die beiden Werte hingegen per verglichen, ist die Abfrage falsch, da die Datentypen nicht übereinstimmen.



Vor allem bei 0 und 1, sowohl als Zahl als auch als Zeichenkette, und bei true und false liefern Gleichheitsoperator und Identisch-Operator unterschiedliche Ergebnisse, wie die folgende Tabelle zeigt:

Vergleich	Ergebnis
1 == "1"	TRUE
TRUE == 1	TRUE
FALSE == 0	TRUE
1 === "1"	FALSE
TRUE === 1	FALSE
FALSE === 0	FALSE
"1" === "1"	TRUE
1 === 1	TRUE
TRUE === TRUE	TRUE
FALSE === FALSE	TRUE

5.2 Die einfache if-Anweisung

In PHP erreichen Sie auf verschiedene Arten eine Verzweigung des Programmablaufs. Die einfachste und häufig eingesetzte Variante ist die Bedingungsprüfung mit der **if**-Anweisung. **Wenn** die Bedingung erfüllt ist, **dann** wird der in Klammern angegebene Anweisungsblock ausgeführt. Ist die Bedingung nicht erfüllt, wird der Anweisungsblock übersprungen.

Syntax und Bedeutung der if-Anweisung

- ✓ Die if-Anweisung beginnt mit dem reservierten Wort `if`.
- ✓ Die Bedingung ① steht in runden Klammern.
- ✓ Der Anweisungsblock ② steht in geschweiften Klammern.
Die geschweiften Klammern sind optional. Falls diese fehlen und die Bedingung `TRUE` ist, wird die Anweisung nur bis zum nächsten Semikolon ausgeführt.
- ✓ Als Bedingung ist ein logischer Ausdruck anzugeben, der einen der beiden Zustände `TRUE` (wahr) oder `FALSE` (falsch) zurückliefert.
- ✓ Liefert die Bedingung `TRUE` zurück, werden die Anweisungen ② ausgeführt. Ist die Bedingung `FALSE`, werden die Anweisungen ② ignoriert.
- ✓ Nach dem Ende der if-Anweisung werden alle weiteren Anweisungen unabhängig von der Bedingung abgearbeitet.

```
if(Bedingung) {
    Anweisungsblock; ②
}
```

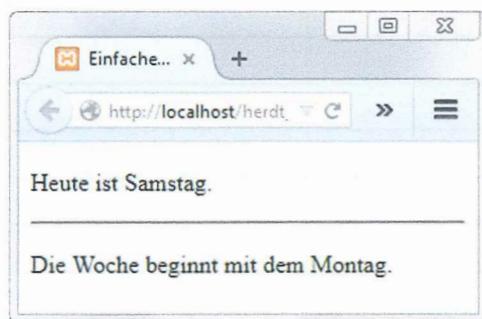


Die auszuführenden Anweisungen ② werden auch Anweisungsblock genannt, auch wenn sie nur aus einer Anweisung bestehen.

Beispiel: if-1.php

```
<?php
① $wochentag = "Samstag";
② if ($wochentag == "Samstag") {
③     echo "<p>Heute ist Samstag.</p>";
④ }
⑤ echo "<hr><p>Die Woche beginnt mit dem Montag.</p>";
?>
```

- ① Der Variablen `$wochentag` wird der Wert `Samstag` zugewiesen.
- ② Es folgt die Prüfung der Bedingung, ob die Variable `$wochentag` den Wert `Samstag` aufweist. Die Prüfung ergibt `TRUE`, sodass der folgende Anweisungsblock ausgeführt wird.
- ③ Der Anweisungsblock in geschweiften Klammern besteht im Beispielskript nur aus einer Meldung, die auf dem Bildschirm ausgegeben wird.
- ④ Die letzte Ausgabe wird unabhängig von dem Ergebnis der Bedingungsprüfung auf dem Bildschirm angezeigt. Der an die Bedingung geknüpfte Anweisungsblock wurde bereits in der Zeile darüber durch das Zeichen `}` beendet.



Anzeige der Beispieldatei "if-1.php"

Häufiger Fehler bei Bedingungsprüfungen



Ein häufiger Fehler ist, dass statt eines Vergleichsoperators mit zwei Gleichheitszeichen `= =` nur ein einzelnes Gleichheitszeichen `=` geschrieben wird (oft als Flüchtigkeitsfehler). Ein einzelnes Gleichheitszeichen `=` ist jedoch eine Zuweisung. Beispielsweise wird über `$wochentag = "Samstag"` der Variablen `$wochentag` der Wert `"Samstag"` zugewiesen. Da diese Zuweisung erfolgreich ist, erkennt PHP immer ein `TRUE` in der Bedingung.

Ein eher selten eingesetzter, aber nützlicher Trick ist es, Variable und Prüfwert zu vertauschen: "Samstag" == \$wochentag. Für den Vergleich selbst spielt die Reihenfolge der einzelnen Argumente keine Rolle. Wird versehentlich ein Gleichheitszeichen vergessen, kommt es sofort zu einem Fehler (*parse error*). 

Beispiel: if-2.php

Sie können auch mehrere if-Anweisungen nacheinander verwenden. So können Sie flexibel mehrere Werte oder Variablen prüfen und darauf individuell reagieren. Mehr als drei if-Anweisungen in Folge sind unübersichtlich, in diesem Fall sei auf die übersichtlichere switch-Anweisung später in diesem Kapitel verwiesen.

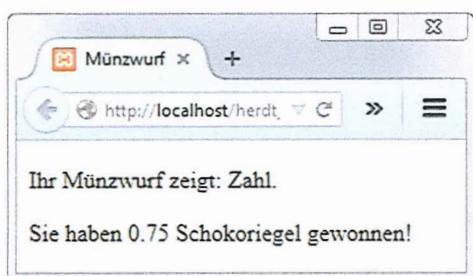
```

① <?php
②     $muenzwurf = "Zahl"; // alternativ: "Kopf" oder "Rand"
③     if ($muenzwurf == "Kopf") {
        echo "<p>Ihr Münzwurf zeigt: Kopf.</p>";
        $gewinn = 0.5;
    }
④     if ($muenzwurf == "Zahl") {
        echo "<p>Ihr Münzwurf zeigt: Zahl.</p>";
        $gewinn = 0.75;
    }
⑤     if ($muenzwurf == "Rand") {
        echo "<p>Ihr Münzwurf zeigt: Rand.</p>";
        $gewinn = 2.5;
    }
⑥     if ($gewinn > 2) {
        echo "<p><strong>Super! Das ist selten...</strong></p>";
    }
⑦     echo "<p>Sie haben $gewinn Schokoriegel gewonnen!</p>";
?

```

Beispieldatei "if-2.php"

- ① Der Variablen \$muenzwurf wird der Wert Zahl zugewiesen. Experimentieren Sie mit dem Skript: Tragen Sie als Wert zum Testen auch einmal Kopf bzw. Rand ein.
- ② Es wird geprüft, ob der Variablenwert gleich Kopf ist. Trifft die Bedingung zu, wird eine Ausgabe sowie eine Variablenzuweisung (\$gewinn) vorgenommen.
- ③ Nach einer weiteren Prüfung, ob der Variablenwert gleich Zahl ist, erfolgt – falls die Überprüfung erfolgreich ist – eine Ausgabe sowie eine weitere Variablenzuweisung (\$gewinn).
- ④ Es folgt eine dritte Prüfung, ob der Variablenwert gleich Rand ist. Bei erfolgreicher Prüfung wird der angegebene echo-Befehl ausgeführt. Es erfolgt zusätzlich eine weitere Variablenzuweisung (\$gewinn).
- ⑤ Abschließend wird geprüft, ob die Variable \$gewinn größer als 2 ist. In dem Fall erfolgt eine weitere Ausgabe.
- ⑥ Die letzte Ausgabe wird in jedem Fall ausgeführt. Sie gehört nicht mehr zu den vorherigen if-Anweisungen.



Ausgabe der Beispieldatei "if-2.php"

if-Anweisung ohne geschweifte Klammern

Beispiel: if-1a.php

Die if-Anweisung (wie auch andere Kontrollstrukturen) benötigt nicht zwingend geschweifte Klammern { }, die den Anweisungsblock umschließen. Werden diese weggelassen, erkennt PHP die Anweisung bis zum nächsten Semikolon als Anweisungsblock.



Diese Schreibweise ist hier nur der Vollständigkeit halber erwähnt. Kontrollstrukturen ohne geschweifte Klammern { } sind schwieriger zu lesen, gerade wenn keine Einrückungen eingehalten werden. Um Fehler zu vermeiden, sollten stets geschweifte Klammern { } verwendet werden.

```
<?php
$wochentag = "Samstag";
if ($wochentag == "Samstag")
    echo "<p>Heute ist Samstag.</p>";
② echo "<hr><p>Die Woche beginnt mit dem Montag.</p>";
?>
```

Im Gegensatz zum Beispiel if-1.php fehlen hier die geschweiften Klammern. Die Ausgabe ist jedoch identisch.

- ① Der Anweisungsblock der if-Anweisung hat hier keine geschweiften Klammern { }. Die vollständige Anweisung wird durch das Semikolon am Ende der Zeile ① beendet.
- ② Diese Zeile befindet sich nicht mehr in der if-Anweisung.

5.3 Die if-Anweisung mit else-Zweig

Soll nicht nur ein Anweisungsblock durchgeführt werden, falls die Bedingung erfüllt ist, sondern ein anderer, sofern die Bedingung nicht erfüllt ist, verwenden Sie die if-else-Anweisung.

Syntax und Bedeutung der if-else-Anweisung

- ✓ Die Alternative leiten Sie mit dem Schlüsselwort else ein.
- ✓ Danach folgen die alternativen Anweisungen.
- ✓ Falls die Bedingung erfüllt ist, wird Anweisungsblock 1 ausgeführt, sonst Anweisungsblock 2 (wenn-dann-sonst).
- ✓ Auch hier sind die geschweiften Klammern { } optional.

```
if(Bedingung) {
    Anweisungsblock 1;
} else {
    Anweisungsblock 2;
}
```

Beispiel: ifelse.php

Wenn die Variable \$menge größer als 5 ist, soll eine Meldung ausgegeben werden. Ist \$menge kleiner als 5, soll eine alternative Meldung am Bildschirm angezeigt werden.

```
<?php
① $menge = 4;
echo "<p>Sie haben $menge Kilo bestellt.</p>";
② if ($menge > 5) {
    echo "<p>Glückwunsch, der Versand ist ab 5 Kilo kostenfrei.</p>";
③ } else {
    echo "<p>Bei kleinen Mengen kostet der Versand leider drei Euro.</p>";
}
?>
```

- ① Der Variablen \$menge wird der Wert 4 zugewiesen.
- ② Es erfolgt die Bedingungsprüfung, ob \$menge größer als 5 ist. Falls die Prüfung TRUE ergibt, wird der folgende Anweisungsblock ausgeführt.
- ③ Trifft die Bedingung nicht zu, wird der mit else eingeleitete alternative Anweisungsblock ausgeführt.

Kurzschreibweise: Ternärer Operator

PHP kennt eine Kurzschreibweise für die if-else-Anweisung. Abfragen lassen sich damit kürzer gestalten. Die Nachvollziehbarkeit und Verständlichkeit der Programmierung kann allerdings durch Einsatz des ternären Operators leiden. Ternäre Operatoren sind schwieriger zu lesen als if-else-Anweisungen.

wenn Bedingung ? dann TRUE : sonst FALSE

Beispiel: *ternaeroperator.php*

```
<?php
    $schalter = 1; // mögliche Werte: 1 / 0
    echo "<p>Das Licht ist " . ($schalter == 1 ? "AN" : "AUS") . " !</p>";
?>
```

Runde Klammern um den ternären Operator führen dazu, dass dieser gesondert vor der Verknüpfung mit der Zeichenkette ausgeführt wird.

Verknüpfung von Bedingungen

In vielen Fällen ist eine Verknüpfung von zwei oder mehreren Bedingungen in einer if-Anweisung notwendig. Bei der Verknüpfung von Bedingungen können Sie mit folgenden logischen Operatoren arbeiten.

Operator	Name	Bedeutung	Beispiel	Ergebnis
AND &&	UND	\$a and \$b ergibt TRUE, wenn sowohl \$a als auch \$b ungleich 0 sind, ansonsten wird FALSE zurückgegeben. Sobald eine der Bedingungsprüfungen FALSE ergibt, werden die weiteren Bedingungen nicht mehr geprüft.	\$a = 1; \$b = 0; \$a AND \$b;	FALSE
OR 	ODER	\$a or \$b ergibt TRUE, wenn mindestens eine der beiden Variablen ungleich 0 ist. Sobald eine der Bedingungsprüfungen TRUE ergibt, werden die weiteren Bedingungen nicht mehr geprüft.	\$a = 1; \$b = 0; \$a OR \$b;	TRUE
XOR	ENTWEDER ODER	\$a xor \$b ergibt TRUE, wenn entweder \$a oder \$b ungleich 0 sind, aber nicht, wenn beide ungleich 0 sind.	\$a = 1; \$b = 0; \$a XOR \$b;	TRUE
!	NICHT	!\$a ergibt die Umkehrung des Wahrheitswertes.	\$a = 0; !\$a;	TRUE

Beispiel: *bedingungen_verknuepfen.php*

Durch Verknüpfung von Bedingungen können Sie z. B. prüfen, ob eine Altersangabe zwischen zwei Werten liegt und entsprechend darauf reagieren.

```
<?php
$alter = 26;
① if ($alter >= 20 && $alter < 30) {
    echo "<p>Wert der Variablen $alter: Zwischen 20 und 29.</p>";
② } else {
    echo "<p>Wert der Variablen $alter: NICHT zwischen 20 und 29.</p>";
}
?>
```

- ① Der Wert der Variablen `$alter` wird überprüft. Damit die Bedingung erfüllt ist und der Anweisungsblock im `if`-Zweig ausgeführt wird, müssen **beide** Bedingungen korrekt sein: `$alter >= 20` und `$alter < 30`.
- ② Wenn allerdings nur eine oder gar keine Bedingung zutrifft, wird der Anweisungsblock des `else`-Zweiges ausgeführt.

5.4 Erweiterte if-Anweisung mit elseif

Zusätzlich zu verschachtelten `if`-Anweisungen können Sie mit `elseif` weitere Bedingungen prüfen. Da Sie `elseif` beliebig oft einsetzen können, ist es möglich, eine beliebige Anzahl von zusätzlichen Bedingungen zu prüfen. Falls keine der vorherigen Bedingungen zutrifft, weder in der `if`-Anweisung noch in weiteren `elseif`-Anweisungen, wird der `else`-Zweig aufgerufen.

```
if (Bedingung 1) {
    Anweisungsblock 1;
} elseif (Bedingung 2) {
    Anweisungsblock 2;
} elseif (Bedingung n) {
    Anweisungsblock n;
} else {
    Anweisungsblock else;
}
```

Allgemeines Beispiel für eine `if`-Anweisung mit `elseif`

- ✓ Für die einzelnen `if`-Anweisungen und ebenso für die einzelnen `elseif`-Anweisungen gelten die gleichen Regeln wie bei einer einfachen `if`-Anweisung.
- ✓ Sie können in einer `if`-Anweisung beliebig viele `elseif`-Anweisungen verwenden.
- ✓ Der `else`-Zweig wird – wie bei der einfachen `if`-Anweisung – nur dann ausgeführt, wenn keine der vorherigen Bedingungen zutrifft.

Beispiel: *if-elseif.php*

Im folgenden Beispiel wird ein Wochentag definiert. Durch eine `if`-Anweisung wird nach folgenden Regeln geprüft, ob es sich um einen Tag handelt, der dem Wochenende zuzurechnen ist oder nicht.

- ✓ *Samstag* oder *Sonntag* führen zur Ausgabe, dass Wochenende ist.
- ✓ *Freitag* führt zur Ausgabe, dass bald Wochenende ist.
- ✓ Alle anderen Werte erzwingen eine Ausgabe, dass kein Wochenende ist.

```

① <?php
    $wochentag = "Freitag"; // testen mit "Mittwoch", "Freitag" und "Samstag"
    echo "<p>Verwendeter Wochentag: $wochentag.</p>";
    if ($wochentag == "Samstag" OR $wochentag == "Sonntag") {
        echo "<p>Es ist Wochenende.</p>";
    } elseif ($wochentag == "Freitag") {
        echo "<p>Bald ist Wochenende.</p>";
    } else {
        echo "<p>Es dauert noch bis zum Wochenende.</p>";
    }
?>

```

- ① Der Variablen \$wochentag wird der Wert Freitag zugewiesen.
- ② Wenn \$wochentag den Wert Samstag oder Sonntag aufweist, erfolgt eine Ausgabe, dass Wochenende ist. Danach ist die Bearbeitung der if-elseif-else-Anweisung beendet.
- ③ Mit der elseif-Anweisung wird ein weiterer möglicher Wert der Variablen \$wochentag geprüft. Ist der Wert der Variablen Freitag, erfolgt die Ausgabe, dass bald Wochenende ist. Auch in diesem Fall ist die Bearbeitung der if-elseif-else-Anweisung beendet.
- ④ Der else-Zweig wird ausgeführt, wenn keine der vorhergehenden Bedingungen zutrifft. In diesem Fall wird ausgegeben, dass es noch dauert, bis Wochenende ist.



Anzeige der Beispieldatei "if-elseif.php" mit \$wochentag = Freitag

5.5 Verschachtelte if-Anweisungen

Oft ist eine Auswahl von einer Bedingung abhängig, die wiederum von einer vorhergehenden Bedingung abhängig ist. Sie haben die Möglichkeit, Bedingungen und deren Anweisungsblöcke ineinander zu verschachteln. Ist die erste Bedingung wahr, wird kontrolliert, ob auch die nächste Bedingung zutrifft. Trifft eine Bedingung nicht zu, wird der alternative Zweig ausgeführt bzw. die verschachtelte Auswahl übersprungen.

```

if (Bedingung 1) {
    Anweisungsblock 1;
} else {
    if (Bedingung 2) {
        Anweisungsblock 2;
    } else {
        if (Bedingung 3) {
            Anweisungsblock 3;
        } else {
            Anweisungsblock 4;
        }
    }
}

```

Allgemeines Beispiel für eine verschachtelte if-Anweisung

- ✓ Für die einzelnen if-Anweisungen gelten die gleichen Regeln wie bei einer einfachen if-Anweisung.
- ✓ Bei einer verschachtelten if-Anweisung, beginnend mit einer öffnenden geschweiften Klammer [{ }, muss diese auch wieder durch eine schließende geschweifte Klammer [}] abgeschlossen werden. Die Verschachtelung erkennen Sie in der Abbildung an den eingerückten Stufen.
- ✓ Der Anweisungsblock 1 wird abgearbeitet, wenn die Bedingung 1 zutrifft. Wenn sie nicht zutrifft, erfolgt die Prüfung der Bedingung 2. Trifft diese zu, erfolgt die Abarbeitung des Anweisungsblocks 2 usw. Trifft keine der Bedingungen zu, wird Anweisungsblock 4 abgearbeitet.



Kennzeichnen Sie eine verschachtelte `if`-Anweisung immer durch eine Einrückung, damit die Struktur auch optisch erkennbar ist. Vermeiden Sie wenn möglich verschachtelte Strukturen, die mehrere Stufen umfassen. Ihr Skript wird sonst zu unübersichtlich und ist schlecht lesbar.

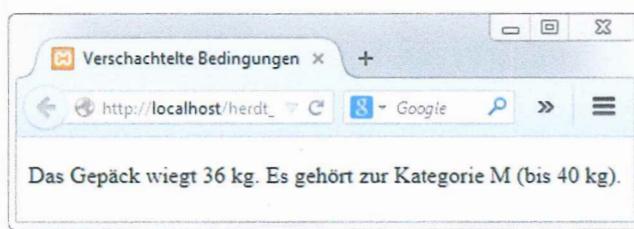
Beispiel: verschachtelung.php

Im folgenden Beispiel wird ein Gepäckstück anhand seines Gewichts in eine Gepäckkategorie eingeordnet. Die Regeln lauten:

- ✓ Wegt das Gepäckstück bis einschließlich 20 kg, gehört es zur Kategorie S;
- ✓ wiegt das Gepäckstück mehr (bis einschließlich 40 kg), gehört es zu Kategorie M;
- ✓ wiegt es mehr als 40 und maximal 60 kg, ist es in Kategorie L einzurichten;
- ✓ ansonsten (über 60 kg) gehört es zur Kategorie XL.

```
<?php
① $gewicht = 36; // Beispiel-Gewicht eines Gepäckstücks in kg
    echo "<p>Das Gepäck wiegt $gewicht kg. Es gehört zur Kategorie ";
② if ($gewicht <= 20) {
③     echo "S (bis 20 kg) ";
④ } else {
⑤     if ($gewicht <= 40) {
⑥         echo "M (bis 40 kg)";
⑦     } else {
⑧         if ($gewicht <= 60) {
⑨             echo "L (bis 60 kg)";
⑩         } else {
⑪             echo "XL (über 60 kg)";
⑫         }
⑬     }
⑭ }
⑮ echo ".</p>";
?>
```

- ① Der Variablen `$gewicht` wird der Wert 36 zugewiesen.
- ② Wenn `$gewicht` kleiner oder gleich 20 ist, wird die nachfolgende `echo`-Anweisung ③ im `if`-Zweig ausgeführt. Danach ist die Bearbeitung der `if`-Anweisung beendet.
- ④ Ist `$gewicht` größer als 20, wird im `else`-Zweig nach einer zutreffenden Bedingung gesucht. Da der `else`-Zweig aus weiteren Prüfungen besteht, finden Sie weitere Bedingungsprüfungen ⑤ - ⑧.



Anzeige der Beispieldatei "verschachtelung.php" bei `$gewicht = 36`

5.6 Fallauswahl mit der `switch`-Anweisung

Wenn Sie eine Variable mit verschiedenen Werten vergleichen und in Abhängigkeit vom Auswertungsergebnis unterschiedlich reagieren möchten, kann die Programmierung mit `if`-Anweisungen sehr aufwendig und durch die Klammerung syntaktisch kompliziert sein. Als Alternative zur verschachtelten Auswahl mit `if` können Sie in diesem Fall die `switch`-Anweisung einsetzen.

Bei der switch-Anweisung bestimmt der Wert einer Variablen, welcher Anweisungsblock ausgeführt wird. Diese Vorgehensweise wird Fallauswahl oder Selektion genannt.

Syntax und Bedeutung der switch-Anweisung

```
switch ($variable) {
    case Wert 1:
        Anweisungsblock 1;
    break;
    case Wert 2:
        Anweisungsblock 2;
    break;
    default:
        Anweisungsblock 3;
}
```

- ✓ Die Fallauswahl wird mit dem reservierten Wort `switch` eingeleitet.
- ✓ Danach folgt die Variable, deren Wert in den folgenden Bedingungsfällen überprüft wird. Anschließend leitet das Zeichen `{` die Fallauswahl ein.
- ✓ Jede Auswahl wird durch das Schlüsselwort `case` (deutsch: *Fall*) eingeleitet, dem die Angabe eines Wertes oder einer Bedingung folgt. Zur Trennung vom folgenden Anweisungsblock wird am Ende der `case`-Angabe ein `:` gesetzt.
- ✓ Stimmt der Wert der Variablen mit einem der aufgeführten Auswahlwerte überein bzw. ergibt die Bedingungsprüfung `TRUE`, dann wird der Anweisungsblock danach bis zur Anweisung `break` ausgeführt. Die restlichen Auswahlblöcke werden nicht ausgeführt.
- ✓ Die `case`-Zweige der `switch`-Anweisung werden der Reihe nach geprüft. Ergibt die Prüfung eine Übereinstimmung, arbeitet PHP den direkt folgenden Anweisungsblock dieses `case`-Zweigs bis zur Anweisung `break` ab. Ist kein `break` am Ende eines `case`-Zweiges angegeben, so werden auch alle nachfolgenden `case`-Blöcke ohne weitere Prüfung ausgeführt, bis eine `break`-Anweisung erfolgt.
- ✓ Stimmt der Wert der Variablen mit keinem der angegebenen Werte überein, wird der Anweisungsblock nach der optionalen `default`-Anweisung durchgeführt. Wird der `default`-Auswahlblock weggelassen, führt das Programm in diesem Fall keine Anweisungen innerhalb der `switch`-Anweisung aus, sondern erst die nächste Anweisung nach dem Ende der Fallauswahl.
- ✓ Die Fallauswahl wird nach dem Zeichen `}` beendet.



Beispiel: switch-case-1.php

Nachfolgend sehen Sie ein Beispiel, in dem der Wert der Variablen geprüft wird:

```
<?php
①    $gewicht = 36; // Beispiel-Gewicht eines Gepäckstücks in kg
    echo "<p>Das Gepäck wiegt $gewicht kg. Es gehört zur Kategorie ";
②    switch ($gewicht) {
③        case ($gewicht <= 20):
            echo "S (bis 20 kg)";
            break;
④        case ($gewicht <= 40):
            echo "M (bis 40 kg)";
            break;
⑤        default:
            echo " XL (über 60 kg)";
    }
    echo "</p>";
?>
```

- ① Der Variablen \$gewicht wird der Wert 36 zugewiesen. Zum eigenen Testen können Sie an dieser Stelle die Wertzuweisung ändern.
- ② In der switch-Anweisung wird angegeben, dass die Variable \$gewicht geprüft werden soll.
- ③ - ⑤ Nacheinander werden die case-Zweige abgearbeitet und es wird nach der passenden Bedingung gesucht. Die Bedingung, die den Wert TRUE liefert, ist bei ④ zu finden. Der passende Anweisungsblock wird bis zum Befehl break ausgeführt.
- ⑥ Falls keiner der case-Zweige zutrifft, z. B. wenn Sie den Wert der Variablen \$gewicht ändern, wird der Anweisungsblock im default-Zweig ausgeführt.

Bei einer switch-Anweisung können Sie in PHP alle denkbaren Fälle prüfen. Dies ist möglich, da in den case-Zweigen der zu testende Wert auf **Gleichheit** geprüft wird:

- ✓ numerische Werte, z. B. case 5 : bei der Prüfung, ob eine Variable den Wert 5 hat
- ✓ Zeichenketten, z. B. case "Spanischer Rotwein" : bei der Prüfung, ob der Wert einer Variablen identisch mit der Zeichenkette "Spanischer Rotwein" ist
- ✓ einfache und verknüpfte Bedingungen, z. B. case (\$a > 10 && \$a <= 20) : bei der Prüfung, ob die Variable \$a einen Wert zwischen 10 und 20 aufweist

Beispiel: *switch-case-2.php*

Darüber hinaus gibt es die Möglichkeit der sogenannten **erweiterten Notation**, die es Ihnen erlaubt, mehrere Fälle zusammenzufassen:

```
<?php
①   $note = 3;
    switch ($note) {
②     case 1: case 2: case 3: case 4:
        echo "<p>Test bestanden.</p>";
        break;
③     case 5:
        case 6:
        echo "<p>Test leider nicht bestanden.</p>";
        break;
④     case "nicht bewertet":
        echo "<p>Der Test wurde abgebrochen und daher nicht bewertet.</p>";
        break;
⑤     default:
        echo "<p>Ich kann keine ganze Note zwischen 1 und 6 erkennen.</p>";
    }
?>
```

- ① Der Variablen \$note wird der Wert 3 zugewiesen. Zum eigenen Testen können Sie an dieser Stelle die Wertzuweisung ändern.
- ② - ③ Im ersten case-Zweig der switch-Anweisung werden mehrere Fälle zusammengefasst. Notieren Sie die einzelnen zu prüfenden Fälle einfach hinter ② oder untereinander ③, jeweils komplett mit dem folgenden [:]. Trifft einer der angegebenen Fälle zu, wird der folgende Anweisungsblock ausgeführt.
- ④ An dieser Stelle erfolgt die Prüfung, ob die Variable \$note mit der angegebenen Zeichenkette "nicht bewertet" übereinstimmt.
- ⑤ Die switch-Anweisung bietet in diesem Beispiel einen optionalen default-Zweig, dessen dazugehöriger Anweisungsblock nur dann ausgeführt wird, wenn die Variable \$note einen anderen Inhalt hat als 1, 2, 3, 4, 5, 6 oder nicht bewertet.

5.7 Schleifen

Schleifen verwenden

Um einen bestimmten Teil des Programms mehrfach auszuführen bzw. zu wiederholen, benötigen Sie Schleifen. Durch die Verwendung von Schleifen sparen Sie Programmcode und können variabel festlegen, wie oft oder bis zum Eintreffen welcher Bedingung die Schleife durchlaufen werden soll.

Folgende Schleifen gibt es in PHP:

- ✓ while- bzw. do-while-Schleife:
Wenn Ihnen als Programmierer **nicht** bekannt ist, wie oft eine Anweisung wiederholt werden soll, bzw. Sie eine Schleife so lange ausführen möchten, bis eine bestimmte Bedingung eingetreten ist, verwenden Sie die while- bzw. do-while-Schleife.
- ✓ for-Schleife:
Wenn Sie die genaue Anzahl kennen oder diese vorher über PHP ermitteln können, wie oft eine Anweisung wiederholt werden soll, verwenden Sie die for-Schleife.

5.8 Mit der while-Schleife arbeiten

In einer while-Schleife prüft PHP zu Beginn die vorgegebene Bedingung und führt den angegebenen Anweisungsblock so lange aus, bis die Bedingung nicht mehr erfüllt ist. Diese Prüfung wird automatisch **vor** jedem Durchlauf wiederholt. Ergibt der Test der Bedingung den Wert **TRUE**, wird die Schleife durchlaufen. Liefert die Bedingung den Wert **FALSE** zurück, werden die Anweisungen der Schleife übersprungen und die Abarbeitung des Programms wird nach der Schleife fortgesetzt.

Eine Prüfung **vor** der Ausführung des Anweisungsblocks wird als **kopfgesteuert** bezeichnet. Ergibt bereits die erste Bedingungsprüfung den Wert **FALSE**, wird der Anweisungsblock gar nicht ausgeführt, d. h., eine while-Schleife kann null Mal bis unendlich oft (sogenannte Endlosschleife) ausgeführt werden.

Syntax und Bedeutung der while-Schleife

- ✓ Das Schlüsselwort **while** leitet die while-Schleife ein. In runden Klammern wird die Bedingung angegeben, die vor der Abarbeitung überprüft wird.
- ✓ Ist der Rückgabewert der Bedingung **TRUE**, wird der Anweisungsblock im Schleifenkörper ausgeführt.
- ✓ Nach der letzten Anweisung des Anweisungsblocks bzw. mit Erreichen der schließenden geschweiften Klammer springt PHP zum Anfang der Schleife zurück und überprüft erneut die Bedingung. Ist das Ergebnis der Bedingungsprüfung **TRUE**, wird der Anweisungsblock erneut ausgeführt.
- ✓ Diese Schritte werden so lange wiederholt, bis die Auswertung der Bedingungsprüfung am Beginn der Schleife den Wert **FALSE** liefert. Die Schleife wird dann verlassen und der PHP-Code nach der Schleife weiter abgearbeitet.

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

Beispiel: **while.php**

```
<?php
① $zahl = 1;
② while ($zahl <= 100) {
    echo "$zahl<br>";
    $zahl = $zahl + $zahl;
⑤ }
?>
```

- ① Der Startwert der Variablen `$zahl` wird auf den Wert 1 festgelegt.
- ② Die Bedingung legt fest, dass die while-Schleife so lange wiederholt werden soll, wie der Wert der Variablen `$zahl` kleiner oder gleich 100 ist. Bedenken Sie, dass die Schleife nicht ein einziges Mal ausgeführt wird, wenn der Wert der Variablen bereits beim ersten Schleifendurchlauf größer als 100 sein sollte.
- ③ In jedem Schleifendurchlauf erfolgt die Ausgabe der Variablen `$zahl`.
- ④ Der Variablen `$zahl` wird an dieser Stelle ein neuer Wert zugewiesen, im Beispiel wird sie verdoppelt.
- ⑤ Die while-Schleife wird durch die schließende Klammer `}` beendet. Nach **jedem** Schleifendurchlauf wird die Bedingung erneut geprüft ②. Der Anweisungsblock in der Schleife wird so lange ausgeführt, bis die Bedingungsprüfung `FALSE` ergibt. Damit ist dann die Schleife beendet und die Abarbeitung des Skripts wird nach der Schleife fortgesetzt.

```
1
2
4
8
16
32
64
```

Anzeige der Beispieldatei "while.php"

Häufige Fehler beim Einsatz von Schleifen

Die in diesem Abschnitt beschriebenen Fehler ergeben keine Fehlermeldungen, sondern einzig ein unerwartetes Verhalten des Programms. Es handelt sich um logische Fehler:

- ✓ Es kann passieren, dass der Variablen, die geprüft werden soll, ein Wert zugewiesen wurde, bei dem die Bedingungsprüfung am Anfang der Schleife gleich beim ersten Schleifendurchlauf den Wert `FALSE` ergibt. Das heißt, die Schleife wird nicht ein einziges Mal ausgeführt, sondern übersprungen. Achten Sie darauf, dass die Bedingungsprüfung im ersten Schleifendurchlauf den Wert `TRUE` ergibt.

Endlosschleifen vermeiden



Schleifen bergen grundsätzlich die Gefahr, dass sie durch eine falsche Logik in der Programmierung in eine Endlosschleife laufen. Dies wird Ihnen nicht als PHP-Fehler gemeldet. Der Aufruf der Datei ist möglich, das Skript bricht ab, wenn die vom Server erlaubte Laufzeit erreicht ist (diese wird in der `php.ini` über den Parameter `max_execution_time` bestimmt und ist standardmäßig auf 30 Sekunden eingestellt). Ist die Skript-Laufzeit erhöht worden, werden große Datenmengen verarbeitet oder viele Datenbankaufrufe in der Schleife abgesendet, kann dies zum Absturz Ihres Browsers oder sogar des ganzen Webservers führen.

Um Endlosschleifen zu vermeiden, müssen Sie darauf achten ...

- ✓ dass der Wert der zu testenden Variablen innerhalb der Schleife verändert wird und
- ✓ dass Sie die Variable so verändern, dass die Schleife nicht endlos ausgeführt wird.

Beispielsweise macht es bei einem Variablenwert 1 und einer Bedingung, dass die Schleife so lange durchlaufen wird, bis die Variable > 5 ist, wenig Sinn, bei jedem Schleifendurchlauf den Wert der Variablen herabzusetzen. In diesem Fall würde die Schleife nicht enden, da die Bedingungsprüfung immer das Ergebnis `TRUE` liefert.

Mit der do-while-Schleife arbeiten

Im Unterschied zur while-Schleife, die die Bedingung für die Wiederholung am Anfang hat, wird in der do-while-Schleife die Bedingung erst **nach** der letzten Anweisung in der Schleife überprüft. Man spricht deshalb von einer **fußgesteuerten** Schleife. Daraus folgt auch, dass der Anweisungsblock immer mindestens einmal ausgeführt wird. Auch hier gilt wie bei der while-Schleife: Liefert die Bedingung einen Wert `TRUE` zurück, wird die Schleife erneut durchlaufen, ansonsten wird sie verlassen.

Syntax und Bedeutung der do-while-Schleife

- ✓ Das Schlüsselwort `do` leitet die Schleife ein.
- ✓ Die Anweisungen im Schleifenkörper werden auf jeden Fall mindestens einmal ausgeführt.
- ✓ Am Ende der Schleife wird die Bedingung der Schleife über das Schlüsselwort `while` ausgewertet. Die Bedingung wird in runden Klammern angegeben.
- ✓ Trifft die Bedingung zu, wird die Schleife erneut ausgeführt.

```
do {
    Anweisungsblock;
}
while (Bedingung);
```

5.9 Mit der for-Schleife arbeiten

Im Unterschied zur `while`-Schleife, die so oft durchlaufen wird, bis der Test der Bedingung `FALSE` ergibt, wird in der `for`-Schleife genau angegeben, wie oft die Schleife durchlaufen werden soll. Bei der `for`-Schleife kann berechnet werden:

- ✓ die Anzahl der Wiederholungen oder
- ✓ Beginn und Ende der Wiederholung.

Übergeben werden der Start- und Endwert sowie die Bedingung, wie der Endwert erreicht werden soll.

Syntax und Bedeutung der for-Schleife

- ✓ Die `for`-Schleife beginnt mit dem reservierten Schlüsselwort `for`.
- ✓ Mit der Anweisung `Initialisierung` wird der Anfangswert für die Schleife festgelegt.

```
for ( Initialisierung; Bedingung; Reinitialisierung) {
    Anweisungsblock;
}
```

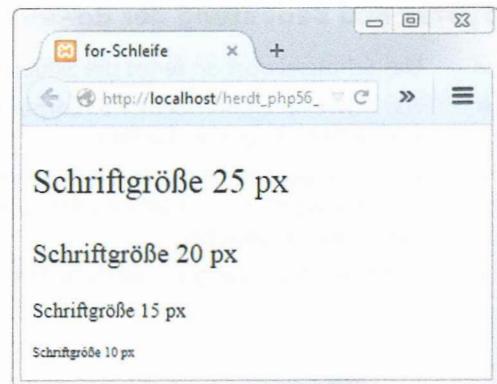
- ✓ Die Anweisung `Bedingung` legt fest, unter welchen Konditionen die Schleife durchlaufen wird. Diese Bedingung, die **vor** jedem Durchlauf abgefragt wird, liefert einen Wert `TRUE` oder `FALSE` zurück. Ist der Wert `TRUE`, wird die Schleife durchlaufen. Andernfalls wird das Programm mit den Anweisungen nach der `for`-Schleife fortgesetzt. Die Bedingung überprüft den Initialwert noch vor dem ersten Durchlauf. Ist das Ergebnis dieser ersten Überprüfung bereits mit den Initialwerten `FALSE`, wird die Schleife erst gar nicht durchlaufen.
- ✓ Die Anweisung `Reinitialisierung` legt fest, wie die Variable verändert werden soll, die in der Bedingung ausgewertet wird. Diese Veränderung geschieht am Ende der Schleife, also nicht am Anfang, d. h., erst beim zweiten Durchlauf der Schleife ist der Wert, der in der `Initialisierung` gesetzt wurde, verändert.
- ✓ Die drei Anweisungen werden jeweils durch ein Semikolon voneinander getrennt.

Beispiel: `for.php`

Die Schrift eines angezeigten Beispieltextes wird mithilfe einer Schleife schrittweise verkleinert.

①	<?php
②	for (\$groesse = 25; \$groesse >= 10; \$groesse -= 5) {
③	echo "<p style='font-size:" . \$groesse . "px'>Schriftgröße \$groesse px</p>";
	}
	?>

- ① Der Startwert der `for`-Schleife wird über die Zuweisung des Wertes 25 an die Variable `$groesse` festgelegt. Die Bedingung besagt, dass die Schleife durchlaufen werden soll, solange der Wert der Variablen `$groesse` größer oder gleich 10 ist. Nach jedem Durchlauf wird der Wert um 5 verringert.
- ② Die Variable `$groesse` steuert über ihren Wert die Schriftgröße der auszugebenden Zeile und wird darüber hinaus ausgegeben.
- ③ Das Ende der `for`-Schleife ist erreicht. Der Interpreter weist der Variablen `$groesse` einen um 5 kleineren Wert zu und springt wieder zurück zur Zeile ①. Dort wird erneut geprüft, ob die Schleife nochmals durchlaufen werden muss.



Anzeige der Beispieldatei "for.php"

Beispiele für Operatoren in `for`-Schleifen

Bei der Arbeit mit einer `for`-Schleife können Sie z. B. mit folgenden Operationen arbeiten:

Bedingung in der <code>for</code> -Schleife	Werte von <code>\$i</code>
<code>for (\$i = 1; \$i <= 5; \$i++)</code>	1, 2, 3, 4, 5
<code>for (\$i = 1; \$i < 5; \$i++)</code>	1, 2, 3, 4
<code>for (\$i = 15; \$i >= 10; \$i--)</code>	15, 14, 13, 12, 11, 10
<code>for (\$i = 15; \$i > 10; \$i--)</code>	15, 14, 13, 12, 11
<code>for (\$i = 0; \$i < 100; \$i = \$i + 10)</code>	0, 10, 20, 30, 40, 50, 60, 70, 80, 90
<code>for (\$i = 1; \$i <= 10; \$i = \$i + 1.2)</code>	1, 2.2, 3.4, 4.6, 5.8, 7, 8.2, 9.4

5.10 Schleifen abbrechen

Schleifenabbruch mit `break`

Nicht immer ist es notwendig, eine `for`-, `while`- oder `do-while`-Schleife so lange zu durchlaufen, bis die definierte Bedingung ein `FALSE` ergibt und damit die Schleife beendet wird. Auch innerhalb des Schleifendurchlaufs können andere Kriterien geprüft werden, die das weitere Durchlaufen der Schleife überflüssig machen.

Beispiel: Sie haben eine Array-Variablen mit 300 Einträgen (Arrays vgl. Kapitel 6) und suchen darin einen bestimmten Wert. Dafür definieren Sie eine `for`-Schleife mit 300 Durchläufen. In jedem Durchlauf wird ein einzelner Eintrag der Array-Variablen untersucht. Sie finden den Wert bereits beim dritten Durchlauf, trotzdem wird die Untersuchung 300-Mal durchgeführt, bis die Bedingung der `for`-Schleife diese beendet. Hier ist es sinnvoll, aus der Schleife **auszusteigen**, um damit Rechenzeit des Skripts zu sparen. Dies ist mit der Anweisung `break` möglich.

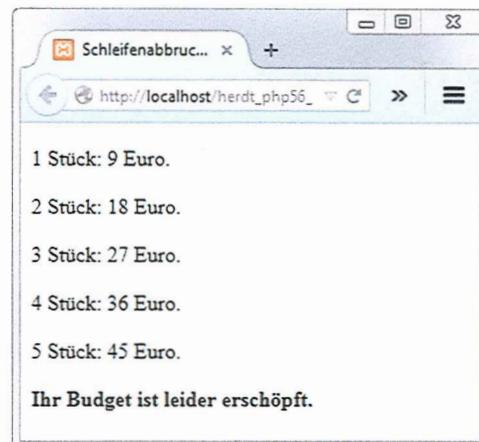
Beispiel: break.php

```

<?php
    $budget = 50;
    $einzelpreis = 9;
    $menge = 1;
    while ($menge <= 15) { // Idealfall: 15 Stück kaufen
        $gesamtpreis = $einzelpreis * $menge;
        if ($gesamtpreis > $budget) { // Budget erschöpft? Wenn ja: Abbruch
            echo "<p><strong>Ihr Budget ist leider erschöpft.</strong></p>";
            break;
        }
        echo "<p>$menge Stück: $gesamtpreis Euro.</p>";
        $menge++;
    }
?>

```

- ① Der Startwert der while-Schleife wird über die Zuweisung des Wertes 1 an die Variable \$menge festgelegt.
- ② Die Bedingung besagt, dass die Schleife durchlaufen werden soll, solange der Wert der Variablen \$menge kleiner oder gleich 15 ist.
- ③ Die Schleife soll allerdings nur ausgeführt werden, bis der Gesamtpreis das zur Verfügung stehende Budget übersteigt. Wenn das passiert, wird die Schleife durch den Befehl break ④ im Anschluss an eine Meldung abgebrochen.
- ⑤ Die Variable \$menge wird bei jedem Schleifendurchlauf um eins erhöht (\$menge++).



Anzeige der Beispieldatei "break.php"

Vorzeitiger Sprung zum nächsten Schleifendurchlauf mit continue

In Schleifen wird der PHP-Code vom Schleifenkopf bis zum Schleifenende vollständig durchlaufen. Es ist jedoch denkbar, dass bei einer bestimmten Bedingung das Ausführen des Codes bis zum Schleifenende überflüssig ist, allerdings sollen weitere Schleifendurchläufe ausgeführt werden.

Für diesen Fall steht das Schlüsselwort `continue` zur Verfügung. Damit beenden Sie einen einzelnen Schleifendurchlauf, die Schleife wird mit dem nächsten Schleifendurchlauf fortgesetzt. Bei `for`-Schleifen wird hier auch die Anweisung der Reinitialisierung ausgeführt, auch wenn das Ende der Schleife noch nicht erreicht wurde.

Beispiel: continue.php

```

<?php
    $zaehler = 5;
    for ($nenner = -2; $nenner <= 2; $nenner++) {
        if ($nenner == 0) {
            echo "<p>Division durch 0 ist verboten.</p>";
            continue;
        }
        $ergebnis = $zaehler / $nenner;
        echo "<p>$zaehler / $nenner = " . $ergebnis . "</p>";
    }
?>

```

- ① In einer `for`-Schleife sollen die Ergebnisse einer Division von `$zaehler` und `$nenner` berechnet und ausgegeben werden. `$zaehler` besitzt mit 5 einen festen Wert. Der Wert der Variablen `$nenner` wird durch die `for`-Schleife verändert (von -2 bis 2).
- ② Um keinen schwerwiegenden Fehler (Division durch 0) auszulösen, soll die Division nicht durchgeführt werden, wenn die Variable `$nenner` den Wert 0 aufweist. In dem Fall wird der **aktuelle Schleifendurchlauf** nach einer `echo`-Anweisung per `continue` ③ beendet. Beachten Sie, dass die Reinitialisierung `$nenner++` auch ausgeführt wird, wenn das Schleifenende noch nicht erreicht ist. Die Variable `$nenner` wird um 1 hochgezählt und die Schleife mit diesem Wert fortgesetzt.
- ④ Sofern der aktuelle Schleifendurchlauf nicht abgebrochen wurde, erfolgt an dieser Stelle die Berechnung und anschließend die Ausgabe des Ergebnisses.

Anzeige der Beispieldatei "continue.php"

5.11 Übung

Kontrollstrukturen in PHP: switch

Übungsdatei: --

Ergebnisdatei: **bewertung_switch.php**

- ① Erstellen Sie ein Programm (`bewertung_switch.php`), das die Bewertung eines Tests in Textform ausgibt. Realisieren Sie diese Aufgabe mithilfe einer `switch`-Auswahlschleife. Legen Sie die erreichte Punktzahl innerhalb des Skripts fest. Die Bedeutung der Punkte lautet:

10 Punkte: Sehr gut
9 Punkte: Gut
8 Punkte: Befriedigend

7 Punkte: Ausreichend
weniger als 7: Leider zu wenige Punkte erreicht

Kontrollstrukturen in PHP: for

Übungsdatei: --

Ergebnisdatei: **bewertung_switch-2.php**

- ② Erweitern Sie das Programm `bewertung_switch.php`, indem Sie eine Schleife programmieren, die automatisch die Bewertung des Tests mithilfe einer `for`-Schleife für erreichte Punktzahlen von 10 bis 0 Punkte ausgibt. Speichern Sie die Datei unter dem Namen `bewertung_switch-2.php`.

```

Übung (Kapitel 5) x +
http://localhost/herdt_php56_grun C Google > = 3

bewertung_switch-2.php

10 Punkte ergeben folgende Bewertung: Sehr gut
9 Punkte ergeben folgende Bewertung: Gut
8 Punkte ergeben folgende Bewertung: Befriedigend
7 Punkte ergeben folgende Bewertung: Ausreichend
6 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
5 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
4 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
3 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
2 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
1 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
0 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.

```

Anzeige der Ergebnisdatei "bewertung_switch-2.php"

Kontrollstrukturen in PHP: while und do-while

Übungsdatei: --

Ergebnisdatei: while_do-while.php

- ③ Entwerfen Sie ein Programm (*while_do-while.php*), das untereinander die Zahlen 1 - 5 ausgibt. Weisen Sie einer Variablen \$zahl den Startwert 1 zu und verwenden Sie eine while-Schleife. Realisieren Sie die Aufgabe anschließend zusätzlich in derselben Datei mit einer do-while-Schleife.
Wenn Sie die Ausgaben von 1 - 5 realisiert haben, weisen Sie der Variablen \$zahl den Startwert 20 zu. Vergleichen Sie das Verhalten der while-Schleife mit dem der do-while-Schleife.

```

Übung (Kapitel 5) x +
http://localhost/herdt C Google > = 3

while- und do-while-Schleife

while-Schleife (Startwert: 1)

1
2
3
4
5

do-while-Schleife (Startwert: 1)

1
2
3
4
5

```

```

Übung (Kapitel 5) x +
http://localhost/herdt C Google > = 3

while- und do-while-Schleife

while-Schleife (Startwert: 20)

do-while-Schleife (Startwert: 20)

20

```

Anzeige der Ergebnisdatei "while_do-while.php" mit Startwert 1 (links) und Startwert 20 (rechts) für die Variable \$zahl

6 Arrays

In diesem Kapitel erfahren Sie

- ✓ welche Arten von Arrays es gibt
- ✓ wie Sie numerische und assoziative Arrays anlegen
- ✓ wie Sie Arrays über die Kurzschreibweise anlegen
- ✓ wie Sie mit ein- und mehrdimensionalen Arrays arbeiten
- ✓ wie Sie einzelne Elemente und komplette Arrays auslesen
- ✓ wo PHP selbstständig Arrays einsetzt

Voraussetzungen

- ✓ Grundlagen PHP
- ✓ Kontrollstrukturen

6.1 Grundlagen zu Arrays

Was sind Arrays?

Bislang wurden in den Beispielen Variablen verwendet, die jeweils nur einen Wert speichern. In diesem Kapitel lernen Sie Arrays kennen. Arrays sind spezielle Variablen, die nicht nur einen, sondern beliebig viele Werte (auch unterschiedlichen Typs) speichern können. Im Sprachgebrauch von Entwicklern ist die englische Bezeichnung „Arrays“ üblich. Mitunter werden Arrays auch Feld-Variablen oder kurz **Felder** genannt.

Arrays sind hilfreich für die reihenweise Verarbeitung von Daten. Stellen Sie sich vor, eine Berechnung liefert Ihnen 20 Ergebnisse. Diese möchten Sie für die spätere Weiterverarbeitung zwischenspeichern. Mit den bislang bekannten Datentypen *boolean*, *integer*, *float* oder *string* können Sie pro Variable nur einen Wert speichern. Um 20 Ergebnisse zu speichern, bräuchten Sie nun 20 einzelne Variablen (`$ergebnis1`, `$ergebnis2`, `$ergebnis3`, `$ergebnis4` ...). Dies ist umständlich und arbeitsaufwändig. In Situationen, in denen Sie die Anzahl der einzelnen Ergebnisse nicht kennen, wäre das Zwischenspeichern mit Einzelvariablen ebenfalls sehr umständlich zu lösen. Arrays bieten auch hier die Lösung: alle Ergebnisse können in **einer** Array-Variablen gespeichert werden.

Nicht nur für das Speichern von zusammengehörigen Werten, sondern auch für die weitere Verarbeitung bieten sich Arrays an. Arrays können einfach mit Schleifen durchlaufen werden. Das ermöglicht eine schnelle Verarbeitung vieler Werte, ohne dass Sie für jeden einzelnen Wert eine einzelne Variable definieren müssen.

Bildlich können Sie sich Arrays wie eine Kommode mit vielen Schubladen vorstellen. Die Kommode wäre das Array, die einzelnen Schubladen die einzelnen Array-Einträge – wobei in jeder Schublade auch wieder eine kleine Kommode mit weiteren Schubladen sein kann (mehrdimensionales Array, siehe nächste Seite).

Der Zugriff auf die einzelnen Ergebnisse erfolgt jeweils über ihre Position innerhalb des Arrays.

1.2
\$ergebnis1

2.4
\$ergebnis2

3.6
\$ergebnis3

...

Ergebnisse einzeln in einfachen Variablen speichern

1.2	2.4	3.6
\$ergebnis		

Alle Ergebnisse in einem Array speichern

Arrays in PHP

Jedes Element eines Arrays besteht aus einem Index bzw. Schlüssel und einem Datenwert. Dabei gibt es zwei Sorten von Arrays, die sich im Aufbau unterscheiden, aber auch dadurch, wie Sie auf die einzelnen Einträge in einem Array zugreifen können:

- ✓ Im **numerisch indizierten** Array werden die einzelnen Werte (value) innerhalb des Arrays über einen Index, eine laufende Nummer, angesprochen.
- ✓ Im **assoziativen** Array werden die einzelnen Werte (value) innerhalb des Arrays über einen eindeutigen Schlüssel, key genannt, angesprochen.

Aufbau von Arrays

Arrays können ein- oder mehrdimensional sein:

- ✓ In einem **eindimensionalen** Array legen Sie eine Liste von Daten ab. Sie können beispielsweise eine Liste von Städten darstellen.
- ✓ Wenn Sie zusammengehörige verschachtelte Daten wie z. B. eine Liste von Ländern und dazugehörigen Werten wie Hauptstädten, Landessprache etc. abbilden möchten, können Sie dies mit einem **mehrdimensionalen** Array tun. Dabei ist es unerheblich, ob es sich um zwei oder mehrere Dimensionen handelt. Mehrdimensionale Arrays sind Array-Variablen, in denen einzelne Einträge selbst Array-Variablen sind.

Eigenschaften von Arrays

- ✓ In PHP können die Werte von Arrays von beliebigen Datentypen sein.
- ✓ Falls Sie keinen Index vergeben, beginnen bei **numerisch indizierten** Arrays die Array-Indizes standardmäßig mit 0. Das erste Element hat dementsprechend den Index 0, das zweite Element hat den Index 1, das dritte Element hat den Index 2 und so weiter. Der Index eines Arrays mit n Elementen reicht somit von 0 bis n - 1. Der Index eines Arrays mit vier Elementen reicht somit von 0 - 3.
- ✓ Arrays müssen in PHP nicht explizit definiert werden. Sie erzeugen ein Array, in dem Sie einer Variablen per `=` und dem Schlüsselwort `array()` die Array-Werte zuweisen – alternativ über die Kurzschreibweise `=` und `[]` (doppelte eckige Klammern). PHP weist dann automatisch den Datentyp `array` zu.
- ✓ Die Anzahl der Array-Einträge muss vorher nicht angegeben werden. Die Größe eines Array hängt von der Anzahl der Einträge ab, mit der Sie das Array erzeugen. Außerdem können Sie nach der Erstellung dem Array zusätzliche Elemente hinzufügen und bestehende Elemente löschen.

6.2 Indizierte eindimensionale Arrays erstellen

Ein indiziertes Array anlegen

PHP bietet Ihnen eine einfache und schnelle Möglichkeit, Arrays mit verschiedenen Werten zu füllen. Dies wird über das `array`-Schlüsselwort realisiert.

Syntax und Bedeutung der `array()`-Funktion

```
$feld = array(Wert1, Wert2, Wert3, ...);
```

- ✓ Die Array-Funktion wird mit `array()` aufgerufen.
- ✓ Der Variablen \$feld werden die einzelnen Werte übergeben, indem sie durch Kommata voneinander getrennt angegeben werden.
- ✓ `integer`, `double` und `boolean`-Werte werden ohne, `string` (Zeichenketten) mit Anführungszeichen angegeben.
- ✓ Die Indizierung der einzelnen Elemente erfolgt automatisch in der Reihenfolge der Angabe.

Beispiel

```
$staedte = array("Frankfurt", "Berlin", "Bern");
```

Es befinden sich drei Werte in dem Array \$staedte: Frankfurt, Berlin und Bern. Die Reihenfolge, in der die Werte zugewiesen worden sind, bestimmt die automatische Indizierung der Werte. Im Array ist dann jeder Wert mit einem Index fest verknüpft, also Index 0 mit Frankfurt usw.

Index	0	1	2
Wert	Frankfurt	Berlin	Bern

Auf indizierte Arrays zugreifen

Um auf ein bestimmtes Element in einem Array zuzugreifen, wird der Index des einzelnen Wertes verwendet. Bei numerisch indizierten Arrays ist dies der automatisch erzeugte fortlaufende Index. Der Index wird dabei in eckigen Klammern direkt hinter dem Variablenamen angegeben:

```
$staedte[Index];
```

Beispiel

```
$auswahl = $staedte[2];
```

Der Variablen \$auswahl wird der Wert des Arrays \$staedte mit dem Indexwert 2, die Zeichenkette Bern, zugewiesen.

6.3 Assoziative eindimensionale Arrays erstellen

Ein assoziatives Array anlegen

Eine andere Art von Arrays sind **assoziativen Arrays**. Bei diesen werden für den Zugriff keine fortlaufenden Indizes benutzt, sondern Schlüssel, über deren Wert auf die einzelnen Werte zugegriffen werden kann. Der Vorteil assoziativer Arrays besteht darin, dass Sie zwei Informationen (ein Informationspaar) ablegen können. Die Schlüssel sind frei wählbar und tragen häufig eine Bedeutung – im Gegensatz zu einem numerischen und meist nicht bedeutungstragenden Index.

Syntax der array () -Anweisung bei assoziativen Arrays

Auch die assoziativen Arrays lassen sich über die array () -Anweisung füllen:

```
$feld = array(Schlüssel1 => Wert1, Schlüssel2 => Wert2, ...);
```

Beispiel:

```
$hauptstaedte = array("Schweiz" => "Bern", "Frankreich" => "Paris");
```

- ✓ Die array () -Anweisung beginnt mit dem reservierten Wort array.
- ✓ Der Variablen werden die einzelnen Wertpaare, bestehend aus Schlüssel (key) und Wert (value), übergeben, indem sie durch Kommata voneinander getrennt angegeben werden.
- ✓ Die Zuweisung von Schlüssel und Wert erfolgt über die Zeichenfolge =>. Dabei steht der Schlüssel links, der Wert rechts vom =>. Diese Zuweisung wird auch als Indizierung bezeichnet.
- ✓ Mögliche Datentypen für den Schlüssel sind *integer* und *string*. Ein *double* als Schlüssel wird zu einem *integer* umgewandelt, *null* zu einer leeren Zeichenkette. Arrays und Objekte sind als Schlüssel nicht erlaubt.
- ✓ Zeichenketten (*string*) als Schlüssel werden in Anführungszeichen angegeben. Sonderzeichen und Leerzeichen im Schlüssel funktionieren zwar prinzipiell. Beim Einsatz von Sonderzeichen bei älteren PHP-Versionen oder abweichender Zeichensatzkonfiguration in der *php.ini* sind Probleme jedoch nicht auszuschließen. Leer- und Sonderzeichen sollten Sie von daher als Array-Schlüssel vermeiden.

- Der Schlüssel eines Arrays ist Case-sensitiv (Groß- und Kleinschreibung wird berücksichtigt). Das heißt, "Schweiz" ist ein anderer Schlüssel als "schweiz". Beide Schlüssel können gleichzeitig in einem Array vorkommen.

Auf assoziative Arrays zugreifen

Beim Zugriff auf einen Wert des Arrays wird der Schlüssel direkt angegeben. Zum Ansprechen des Arrays kann auch eine Variable eingesetzt werden, die den Wert des Schlüssels gespeichert hat:

	Ergebnis
<code>\$auswahl = \$hauptstaedte ["Schweiz"] ;</code>	Die Variable \$auswahl hat den Wert "Bern".

oder:

	Ergebnis
<code>\$k = "Schweiz"; \$auswahl = \$hauptstaedte[\$k] ;</code>	Die Variable \$auswahl hat den Wert "Bern".

Beispiel: array_assoz.php

Hier wird die Zuweisung der Städtenamen mit der `array()`-Anweisung realisiert.

```

① <?php
    $hauptstaedte = array("Schweiz" => "Bern",
                          "Frankreich" => "Paris",
                          "Deutschland" => "Berlin",
                          "Spanien" => "Madrid");
② $k = "Spanien";
③ echo "<p>Die Hauptstadt von $k ist " . $hauptstaedte["Spanien"] . ".</p>";
④ echo "<p>Die Hauptstadt von $k ist " . $hauptstaedte[$k] . ".</p>";
?>

```

- ① Das assoziative Array wird über die `array()`-Funktion und die Zuweisung der Schlüssel sowie Array-inhalte angelegt.
Die Einrückungen und Zeilenumbrüche dienen der Übersichtlichkeit des PHP-Skripts, sie sind jedoch für PHP nicht notwendig. Das ganze Array könnte auch in einer Zeile definiert sein.
- ② Der Variablen `$k` wird die Zeichenkette `Spanien` zugewiesen.
- ③ Das Array wird mit dem Schlüssel `Spanien` angesprochen und liefert den dazugehörigen Wert `Madrid`. Die Bezeichnung des Schlüssels wird als Zeichenkette und in Anführungszeichen in den eckigen Klammern angegeben.
- ④ Alternativ können Sie mit der Variablen `$k`, welche den Wert `Spanien` hat, das entsprechende Element aus dem Array auslesen und anzeigen. In diesem Fall verwenden Sie keine Anführungszeichen.

Werden Arrays mit eigenen Schlüsseln aufgebaut, sind in dem Array keine numerisch indizierten Indizes vorhanden. `$hauptstaedte[3]` würde hier ein leeres Ergebnis liefern, da der Schlüssel 3 nicht definiert ist.



6.4 Arrays mit der Kurzschreibweise erstellen

Array-Kurzschreibweise

Alternativ zu der Syntax mit dem Schlüsselwort `array()` können Sie die Array-Kurzschreibweise einsetzen, um Arrays zu erzeugen. Über die Kurzschreibweise können Sie sowohl indizierte als auch assoziative Arrays erstellen. Die Kurzschreibweise ist schwerer zu lesen als die Array-Deklarationen mit `array()`, bietet jedoch die Möglichkeit, PHP-Code kompakter zu schreiben.

Syntax der Kurzschreibweise

```
$feld = [Wert1, Wert2, Wert3, ...];
```

- ✓ Das Schlüsselwort `array` entfällt in der Kurzschreibweise.
- ✓ Statt der runden Klammern `()` werden die Werte in eckigen Klammern `[]` angegeben.
- ✓ Ansonsten gelten die gleichen Regeln wie bei indizierten und assoziativen Arrays.

Beispiel

```
$staedte = ["Frankfurt", "Berlin", "Bern"];
```

Beispiel: `array_kurzschrifweise.php`

Entsprechend dem Beispiel `array_assoz.php` wird in diesem Beispiel genau die gleiche Array-Variable definiert:

①	<pre><?php \$staedte = ["Schweiz" => "Bern", "Frankreich" => "Paris", "Deutschland" => "Berlin", "Spanien" => "Madrid"]; </pre>
②	<pre>echo "<p>Die Hauptstadt von Spanien ist " . \$staedte["Spanien"] . ".</p>";</pre>
?>	

- ① Das assoziative Array wird über die eckigen Klammern `[]` und die Zuweisung der Schlüssel sowie der Array-Inhalte angelegt. Auch hier dienen Zeilenumbrüche und Einrückungen lediglich der Übersicht des PHP-Codes.
- ② Hier wird das Element durch die direkte Angabe des Schlüssels angesprochen. Die Bezeichnung des Schlüssels wird als Zeichenkette in den eckigen Klammern und in Anführungszeichen hinter dem Array-Variablenname angegeben.



Bevor Sie eine Webseite für den Live-Betrieb programmieren, stellen Sie sicher, dass Ihr Internet-Provider die passende PHP-Version installiert hat (das finden Sie über die Funktion `php_info()` heraus). Die Array-Kurzschreibweise wurde mit der PHP-Version 5.4 eingeführt. Nicht alle Internet-Provider stellen sofort auf die aktuelle PHP-Version um – im November 2014 unterstützten ca. 15 % der Provider nur PHP 5.3. Falls dies bei Ihrem Provider der Fall sein sollte, sollten Sie auf die Kurzschreibweise verzichten. Das erspart Ihnen mögliche Fehler und viele Umbauarbeiten, wenn Ihr PHP-Skript zwar bei Ihnen auf dem Rechner funktioniert, auf dem Server des Internet-Providers aber nicht.

6.5 Mit eindimensionalen Arrays arbeiten

Arrays ändern

Um einen Wert innerhalb einer Array-Variablen zu ändern, geben Sie bei der Wertzuweisung bei der Array-Variablen den entsprechenden Index bzw. Schlüssel des zu ändernden Wertes an und weisen Sie den neuen Wert zu.

```
Indiziert: $staedte[1] = "Paris";
Assoziativ: $staedte["Frankreich"] = "Paris";
```

Sofern bereits ein Element mit dem angegebenen Index bzw. Schlüssel vorhanden ist, wird der alte Wert durch den neuen ersetzt. Gibt es mit dem Schlüssel noch keinen Eintrag in dem Array, fügen Sie mit diesen Anweisungen neue Einträge im Array hinzu.

Wenn Sie den Eintrag Paris löschen wollen, erreichen Sie das mithilfe von `unset()`. In die runden Klammern geben Sie die genaue Bezeichnung des Index bzw. Schlüssels der Array-Variablen ein:

Indiziert: <code>unset(\$staedte[1]);</code>
Assoziativ: <code>unset(\$staedte["Frankreich"]);</code>

Es wäre auch denkbar, die Schreibweise `$staedte[1] = false` oder `$staedte[1] = ""` einzusetzen. Allerdings nehmen Sie mit dieser Syntax lediglich eine neue Wertzuweisung vor, der Wert von `$staedte[1]` wäre dann `false` bzw. leer, der Schlüssel 1 bleibt jedoch im Array erhalten. `unset()` hingegen löscht nicht nur den Wert, sondern die vollständige Schlüssel-Wert-Kombination.



Arrays erweitern

Arrays können einfach erweitert werden, indem Sie einen neuen Wert zuweisen – entweder mit oder ohne Schlüssel. In beiden Fällen wird ein neues Array-Element an das Ende des Arrays angefügt.

Syntax und Bedeutung

Indiziert: <code>\$feld[] = Wert;</code>
Assoziativ: <code>\$feld["key"] = Wert;</code>

- ✓ Um einem Array einen weiteren Wert (oder mehrere Werte) hinzuzufügen, brauchen Sie bei der Wertzuweisung in einem indizierten Array keinen Index anzugeben. Das Array wird am Ende um den Wert (bzw. die Werte) ergänzt und der Index automatisch erhöht.
- ✓ In einem assoziativen Array müssen Sie einen Schlüssel angeben, ansonsten verwendet PHP bei diesem Element des Arrays automatisch einen numerischen Indexwert.
- ✓ Sollte die Array-Variable noch nicht bestehen, erstellt PHP sie automatisch, sobald Sie der Variablen einen Wert zuweisen. Sollte der Schlüssel bereits vergeben sein, fügen Sie kein Element hinzu, sondern überschreiben den bestehenden Eintrag.
- ✓ Numerisch indizierte und assoziative Arrays schließen sich gegenseitig nicht aus. Fügen Sie einem assoziativen Array einen Wert ohne konkreten Schlüssel hinzu, fügt PHP automatisch einen numerischen Index hinzu. Ebenso können Sie einem numerisch indizierten Array ein Schlüssel-Werte-Paar hinzufügen. In beiden Fällen entsteht eine Mischform aus beiden Array-Typen, die sowohl numerische Indexwerte als auch selbst definierte Schlüssel hat.

Beispiel: `array_indiz.php` (Auszug)

<code>\$staedte = array("Frankfurt", "Berlin", "Bern");</code>
<code>\$staedte[] = "Graz";</code>
<code>\$staedte[] = "Rom";</code>

ergibt:

Index	0	1	2	3	4
Wert	Frankfurt	Berlin	Bern	Graz	Rom

Indexwerte manuell vergeben

Bei assoziativen Arrays werden Schlüsselwerte stets vom Benutzer eingegeben und mit dem Wert durch die Zeichenfolge `=>` zugewiesen. Bei indizierten Arrays können Sie ebenfalls Indexeinträge selbst vergeben. Sie sind nicht an die automatisch vergebenen Indexwerte gebunden.

Beispiel: `array_indiz_manuell.php`

Hier wurde die Zuweisung der Städtenamen mit der `array()`-Anweisung realisiert.

```
<?php
① $staedte = array(1 => "Frankfurt", "Berlin", "Bern");
② $staedte[34] = "Graz";
③ $staedte[] = "Rom";
④ $staedte[5] = "Hamburg";
⑤ echo "<pre>";
print_r($staedte);
echo "</pre>";
?>
```

- ① Das Array `$staedte` wird über die `array()`-Funktion angelegt und mit Werten gefüllt. Sie weisen einen Indexwert zu, indem Sie den Indexwert zusammen mit den Zeichen `=>` vor den gewünschten Eintrag schreiben. In diesem Fall wird dem Eintrag Frankfurt der Index 1 zugewiesen. Da es sich um den ersten Eintrag im Array handelt, wird der Indexwert 0 nicht vergeben. Bei einer automatischen Indizierung wird die Nummerierung einfach nach dem höchsten vorhandenen Wert fortgesetzt, also keine eventuell vorhandenen Lücken aufgefüllt. Berlin erhält den Index 2, Bern den Index 3.
- ② Der Eintrag Graz erhält manuell den Indexwert 34.
- ③ Der Indexwert für den Eintrag Rom wird automatisch vergeben. PHP sucht den größten Indexwert im Array (34) und setzt die Index-Nummerierung mit dem nächsthöheren Wert 35 fort.
- ④ Hier wird ein Index vergeben, der kleiner ist als der höchste Index im Array. Dieser Index wird für den Eintrag im Array verwendet. In der Ausgabe ⑤ können Sie allerdings beobachten, dass der Eintrag **ans Ende des Arrays angefügt**, und **nicht** – wie vielleicht erwartet – mit dem Index in das Array eingesortiert wird. Für die Sortierung der Array-Elemente stehen unterschiedliche Array-Funktionen zur Verfügung (weiter unten).
- ⑤ Die Funktion `print_r()` gibt den Inhalt der Array-Variablen `$staedte` mit allen Einträgen und dazugehörigen Indexwerten aus. Die Array-Variable wird in runden Klammern in der Funktion `print_r($staedte)` geschrieben. So können Sie bei Arrays überprüfen, welche Indexwerte den Einträgen zugeordnet sind. Das HTML-Tag `<pre>` sorgt dafür, dass das Array zeilenweise ausgegeben wird.

```
Indexwerte teilweise manuell vergeben

Array
(
    [1] => Frankfurt
    [2] => Berlin
    [3] => Bern
    [34] => Graz
    [35] => Rom
    [5] => Hamburg
)
```

Ausgabe des Array-Inhaltes (Beispieldatei "array_indiz_manuell.php")

6.6 Daten aus eindimensionalen Arrays extrahieren

Daten an Variablen übergeben

Mit einer `foreach()`-Schleife sind Sie in der Lage, die einzelnen Werte der Array-Elemente auszulesen. Dabei wird jedes Element in einer neuen Variablen zwischengespeichert. Ein Vorteil der `foreach()`-Schleife ist, dass Sie weder die Schlüssel des Arrays noch die Anzahl der Array-Einträge kennen müssen. Die Schleife durchläuft das Array bis zum letzten Eintrag.

```

①   foreach($feld as $wert) {
      Anweisungsblock;
    }

②   foreach($feld as $index => $wert) {
      Anweisungsblock;
    }
  
```

- ✓ Die Funktion `foreach()` erwartet die Angabe des assoziativen oder indizierten Arrays, dessen Elemente durchlaufen werden sollen.
- ✓ In der ersten Variante ① werden mithilfe von `foreach` bei jedem Schleifendurchlauf der Variablen `$wert` nacheinander die Werte der Array-Elemente zugewiesen. Diese Variante wird häufig für indizierte Arrays eingesetzt, da hier die automatisch vergebenen Indexwerte oft ohne Bedeutung sind und nicht weiter verarbeitet oder ausgegeben werden sollen.
- ✓ Mithilfe der zweiten Variante ② können Sie zusätzlich den Index jedes Array-Elements auslesen. Bei jedem Durchlauf wird der Schlüssel des Array-Eintrags der Variablen `$index` zugewiesen, der dazugehörige Array-Wert der Variablen `$wert`. Diese Variante wird häufig bei assoziativen Arrays verwendet, wenn der Schlüssel für die weitere Verarbeitung benötigt oder ausgewertet wird. Aber auch bei indizierten Arrays kann diese Variante eingesetzt werden.

Beispiel: `foreach.php`

Verschiedene Länder der Welt und deren Hauptstädte werden in einem assoziativen Array gespeichert. Zur Ausgabe sollen das Land (= Schlüssel) und die dazugehörige Hauptstadt (= Wert) in einer Tabelle angezeigt werden:

```

① <?php
    $hauptstaedte = array("Schweiz" => "Bern",
                          "Frankreich" => "Paris",
                          "Deutschland" => "Berlin");
    $hauptstaedte["Polen"] = "Warschau";
    $hauptstaedte["Italien"] = "Rom";
    $hauptstaedte["Spanien"] = "Madrid";
    echo "<table border='1'>";
    echo "<tr><th>Land</th><th>Hauptstadt</th></tr>";

    ④   foreach ($hauptstaedte as $land => $stadt) {
        ⑤     echo "<tr><td>$land</td><td>$stadt</td></tr>";
      }

    ⑥   echo "</table>";
  ?>
  
```

Beispieldatei "foreach.php"

- ① Das assoziative Array `$hauptstaedte` wird über die `array`-Funktion neu erstellt und mit Werten gefüllt.
- ② Dem assoziativen Array `$hauptstaedte` werden weitere Elemente hinzugefügt.

- ③ Der HTML-Tabellenkopf und die Kopfzeile der Tabelle werden **vor** der `foreach`-Schleife erstellt, der Tabellenfuß **nach** der Schleife ⑥.
- ④ Mithilfe einer `foreach`-Schleife können Sie auf jedes Element des Arrays zugreifen. Die Schleife wird so lange ausgeführt, bis alle Elemente durchlaufen wurden. In jedem Schleifendurchlauf wird der Variablen `$land` der jeweilige Schlüssel und der Variablen `$stadt` der Wert des Arrays an der durch den Schlüssel festgelegten Position zugeordnet.
- ⑤ Die Werte werden in die Tabelle eingetragen und über den Ausgabebefehl `echo` am Bildschirm ausgegeben.

Land	Hauptstadt
Schweiz	Bern
Frankreich	Paris
Deutschland	Berlin
Polen	Warschau
Italien	Rom
Spanien	Madrid

Ausgabe von Werten eines assoziativen Arrays (Beispieldatei "foreach.php")

6.7 Mehrdimensionale indizierte Arrays erstellen

Grundlagen zu mehrdimensionalen Arrays

Im folgenden Beispiel sehen Sie mehrere Angaben (*Vorname*, *Nationalität* und *Alter*), die jeweils einen Eintrag in einer Array-Variablen bilden. Damit ist dieses Array ein mehrdimensionales Array. Bei jeder der Angaben sind Mehrfachwerte möglich. Mehrdimensionale Arrays sind verschachtelte Arrays, man spricht hier von äußerem und inneren Arrays.

Mehrdimensionale Arrays können sowohl numerisch indiziert als auch assoziativ sein. Eine Mischform aus numerisch indizierten und assoziativen Arrays ist ebenfalls möglich.

Index 1	Index 2	Wert
0	0	Oliver
	1	spanisch
	2	37 Jahre
1	0	Maria
	1	deutsch
	2	23 Jahre
2	0	Oliver
	1	englisch
	2	46 Jahre

In dieser Tabelle wird bereits die Arbeitsweise mehrdimensionaler Arrays ersichtlich. Um beispielsweise an die Informationen von *Maria* zu gelangen, suchen Sie die Zeile mit der Angabe *Maria* und lesen die einzelnen Werte dieser Zeile aus: *Maria*, *deutsch*, *23 Jahre*.

Mit mehrdimensionalen indizierten Arrays arbeiten

Ein mehrdimensionales indiziertes Array hat statt **eines** Indexes – in Abhängigkeit von der Anzahl der Verschachtelungen – **mehrere** Indizes.

Syntax eines mehrdimensionalen indizierten Arrays

Die bereits erwähnten Informationen sollen im Folgenden als mehrdimensionales Array realisiert werden.

Auf eine bestimmte Angabe in einem mehrdimensionalen indizierten Array zugreifen

```
$person = array(
    array("Oliver",
          "Spanisch",
          "37 Jahre"
        ),
    array("Maria",
          "Deutsch",
          "23 Jahre"
        ),
    array("Oliver",
          "Englisch",
          "46 Jahre"
        )
);

$vorname1      = $person[1][0];
$nationalitaet1 = $person[1][1];
$alter1        = $person[1][2];

echo "$vorname1 ist $alter1 alt und spricht $nationalitaet1.<hr>";
```

Ausschnitt aus der Beispieldatei "mehrdimensional.php"



Ausgabe des oben gezeigten Ausschnitts aus der Beispieldatei "mehrdimensional.php"

Ein mehrdimensionales indiziertes Array erweitern

```
$person[3][0] = "Johanna";
$person[3][1] = "schwedisch";
$person[3][2] = "19 Jahre";
```

Oder alternativ:

```
$person[] = array("Johanna", "schwedisch", "19 Jahre");
```

Beides Ausschnitte aus der Beispieldatei "mehrdimensional.php"

6.8 Mit mehrdimensionalen assoziativen Arrays arbeiten

Syntax eines mehrdimensionalen assoziativen Arrays

Die assoziativen Arrays besitzen zum Ansprechen der einzelnen Elemente einen Schlüssel. Um ein mehrdimensionales Array anzulegen, wird jedem Schlüssel ein weiteres Array mit Schlüssel-Wert-Paaren übergeben (Sonderfall zweidimensional). Dabei wird jedem Schlüssel auf erster Ebene ein weiteres Array zugewiesen. In diesen verschachtelten Arrays werden dann jeweils mehrere Schlüssel mit Werten angegeben.

Beispiel: *mehrdimensional.php*

```
$land = array(
    "Spanien" => array("Hauptstadt" => "Madrid",
                         "Sprache" => "Spanisch",
                         "Waehrung" => "Euro",
                         "Flaeche" => "504645 qkm"
                     ),
    "England" => array("Hauptstadt" => "London",
                         "Sprache" => "Englisch",
                         "Waehrung" => "Pfund Sterling",
                         "Flaeche" => "130395 qkm"
                     ),
    "Portugal" => array("Hauptstadt" => "Lissabon",
                         "Sprache" => "Portugiesisch",
                         "Waehrung" => "Euro",
                         "Flaeche" => "92345 qkm"
                     )
);
```

Hiermit ist jeder Array-Eintrag mit entsprechenden Länderinformationen über einen Schlüssel mit aussagekräftiger Bezeichnung – Spanien, England oder Portugal – direkt ansprechbar.

Auf ein bestimmtes Element in einem mehrdimensionalen assoziativen Array zugreifen

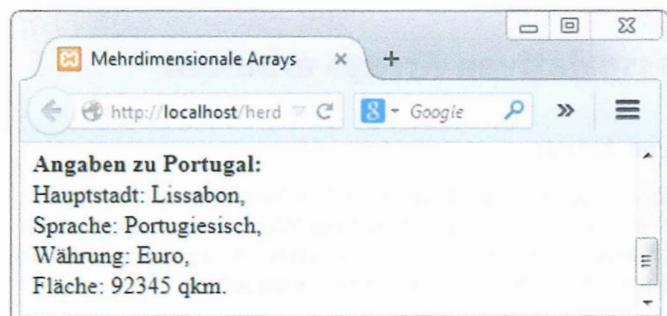
Bei numerisch indizierten Arrays greifen Sie mit dem numerischen Index auf die einzelnen Array-Einträge zu. Wenn Sie ein bestimmtes Array-Element in einem assoziativen Array direkt ansprechen wollen, müssen Sie die Elemente über den Schlüssel ansprechen. Mit dem ersten Schlüssel (hier Portugal) erhalten Sie das verschachtelte Array mit dem entsprechenden Index, mit dem zweiten Schlüssel (hier Hauptstadt usw.) ermitteln Sie dann den zugehörigen Wert.

Die Schreibweise sieht folgendermaßen aus: Sie geben den Variablennamen des Arrays an, dahinter zwei doppelte Pärchen von eckigen Klammern `[]`. Das erste Pärchen nimmt den ersten Schlüssel auf, das zweite Pärchen dann den Schlüssel des inneren Array, also `$wert["key-1"]["key-2"]`. Auch hier verwenden Sie Anführungszeichen für Zeichenketten-Schlüssel.

```
$hauptstadt2 = $land["Portugal"]["Hauptstadt"];
$sprache2   = $land["Portugal"]["Sprache"];
$waehrung2  = $land["Portugal"]["Waehrung"];
$flaeche2   = $land["Portugal"]["Flaeche"];

// Ausgabe der Angaben von Portugal
echo "<p><strong>Angaben zu Portugal:</strong><br>
      Hauptstadt: $hauptstadt2,<br>
      Sprache: $sprache2,<br>
      Währung: $waehrung2,<br>
      Fläche: $flaeche2.</p>";
```

Ausschnitt aus Beispieldatei "mehrdimensional.php"



Ausgabe des oben gezeigten Ausschnitts aus der Beispieldatei "mehrdimensional.php"

Ein mehrdimensionales assoziatives Array erweitern

```
$land["Ungarn"]["Hauptstadt"] = "Budapest";
$land["Ungarn"]["Sprache"] = "Ungarisch";
$land["Ungarn"]["Waehrung"] = "Forint";
$land["Ungarn"]["Flaeche"] = "93036 qkm";
```

Oder alternativ:

```
$land["Ungarn"] = array("Hauptstadt" => "Budapest",
                        "Sprache"      => "Ungarisch",
                        "Waehrung"     => "Forint",
                        "Flaeche"       => "93036 qkm");
```

Ausschnitt aus Beispieldatei "mehrdimensional.php"

6.9 Daten aus mehrdimensionalen Arrays extrahieren

Bei der Arbeit mit mehrdimensionalen Arrays kommt es häufig vor, dass Sie die kompletten Daten eines inneren Arrays benötigen. Mit einer `foreach`-Schleife können Sie alle Werte einer Array-Variablen auslesen. Bei der Verwendung einer `foreach`-Schleife müssen Sie keinen Index oder Schlüssel kennen. Bei jedem Schleifendurchlauf wird das aktuelle Element in einer von Ihnen angegebenen Variablen zwischengespeichert. Hierbei handelt es sich auch um eine Array-Variable mit dem aktuellen „Datensatz“.

Syntax

```
foreach($feld as $wert);
```

Bei jedem einzelnen Schleifendurchlauf wird das verschachtelte Array in der Variablen `$wert` gespeichert. Um nun die einzelnen Werte dieses Arrays auszulesen, wird ein noch nicht vorgestellte PHP-Funktion verwendet: die `list()`-Funktion.

Syntax

```
list($variable1, $variable2, ...) = $wert;
```

- ✓ `list()` dient dazu, die einzelnen Werte des inneren Arrays aufzunehmen und sie den Variablen zuzuweisen, die als Parameter in `list()` angegeben sind.
- ✓ In den runden Klammern der Funktion `list()` werden die Variablennamen angegeben, in denen die einzelnen Array-Werte gespeichert werden sollen. Diese können frei gewählt werden.
- ✓ Über dem Zuweisungsoperator `=` wird der Funktion `list()` die Array-Variable zugewiesen, im Beispiel `$wert`. Dies wiederholt sich beim `foreach()` bei jedem einzelnen Schleifendurchlauf.
- ✓ Nun werden die Werte der in `list()` angegebenen Variablen mit den einzelnen Einträgen des Arrays versehen. Die erste Variable erhält den Wert des ersten Eintrags aus dem Array `$wert`, die zweite Variable den des zweiten Eintrags aus dem Array usw.
- ✓ `list()` benötigt numerisch indizierte Arrays, die mit dem Schlüssel 0 beginnen und die nicht mit abweichenden Schlüsseln versehen worden sind.

Beispiel: `array_mehrdimensional_list.php`

Verschiedene Angaben zu Ländern werden in einem mehrdimensionalen Array gespeichert. Zur Ausgabe sollen der Name des Landes, die Hauptstadt, die Sprache und die Landeswährung in separaten Variablen abgelegt werden.

```


| Land     | Hauptstadt                                                                      | Sprache | Währung |
|----------|---------------------------------------------------------------------------------|---------|---------|
|          |                                                                                 |         |         |
| <b>①</b> | <b>&lt;?php</b>                                                                 |         |         |
|          | // Array \$staedte wird definiert und gefüllt                                   |         |         |
|          | \$staedte = <b>array</b> (                                                      |         |         |
|          | "Japan"      => <b>array</b> ("Tokio",      "Japanisch",      "Yen") ,          |         |         |
|          | "Niederlande" => <b>array</b> ("Amsterdam",  "Niederländisch",  "Euro") ,       |         |         |
|          | "Polen"       => <b>array</b> ("Warschau",   "Polnisch",       "Z&#322;oty") ,  |         |         |
|          | "Indien"      => <b>array</b> ("Neu Delhi",   "Indisch",       "Rupie") ,       |         |         |
|          | "Island"      => <b>array</b> ("Reykjavik",  "Isländisch",   "Krone") ,         |         |         |
|          | "Italien"     => <b>array</b> ("Rom",        "Italienisch",  "Euro") ,          |         |         |
|          | "Frankreich" => <b>array</b> ("Paris",       "Französisch",  "Euro") ,          |         |         |
|          | "Spanien"     => <b>array</b> ("Madrid",      "Spanisch",     "Euro") ,         |         |         |
|          | "England"     => <b>array</b> ("London",      "Englisch",     "Pfund Sterling") |         |         |
|          | ) ;                                                                             |         |         |
| <b>②</b> | <b>// Auslesen des gesamten Arrays mit foreach</b>                              |         |         |
| <b>③</b> | <b>foreach</b> (\$staedte <b>as</b> \$key => \$ausgabe) {                       |         |         |
| <b>④</b> | <b>list</b> (\$hauptstadt, \$sprache, \$waehrung) = \$ausgabe;                  |         |         |
| <b>④</b> | <b>echo</b> "<tr>" ;                                                            |         |         |
| <b>④</b> | <b>echo</b> "<td>" . \$key . "</td>" ;                                          |         |         |
| <b>④</b> | <b>echo</b> "<td>" . \$hauptstadt . "</td>" ;                                   |         |         |
| <b>④</b> | <b>echo</b> "<td>" . \$sprache . "</td>" ;                                      |         |         |
| <b>④</b> | <b>echo</b> "<td>" . \$waehrung . "</td>" ;                                     |         |         |
| <b>④</b> | <b>echo</b> "</tr>" ;                                                           |         |         |
|          | }                                                                               |         |         |
|          | <b>?&gt;</b>                                                                    |         |         |
|          | <b>&lt;/table&gt;</b>                                                           |         |         |


```

- ① Das assoziative Array \$staedte wird neu definiert und mit Werten gefüllt. Die Daten enthalten Informationen zur Hauptstadt eines Landes, zur Sprache und zur im Land verwendeten Währung. Der Name des Landes selbst wird als eindeutiger Schlüssel verwendet. Die verschachtelten Arrays selbst sind numerisch indizierte Arrays. Nur so kann die Funktion **list()** einwandfrei arbeiten.
- ② Mithilfe einer **foreach**-Schleife durchlaufen Sie das Array, bis das letzte Array-Element erreicht ist. Die **foreach()**-Schleife speichert dabei in jedem Durchlauf das verschachtelte Array in der Variablen \$ausgabe. Im ersten Schleifendurchlauf enthält die Variable \$ausgabe nur das Werte-Array des zuerst definierten Landes Japan (Tokio, Japanisch, Yen), im zweiten Schleifendurchlauf die Werte der Niederlande etc.
- ③ Über den Befehl **list()** werden die Daten zum aktuell abgerufenen Land (Hauptstadt, Sprache und Währung) in die entsprechenden Variablen \$hauptstadt, \$sprache und \$waehrung überführt.
- ④ Über den Ausgabebefehl **echo** werden die Werte der Variablen am Bildschirm ausgegeben. Bei der Variablen \$key handelt es sich um den Schlüssel (Landesnamen), der in ② durch die **foreach()**-Schleife zugewiesen wurde.

The screenshot shows a web browser window with the title "Mehrdimensionale Arrays". The address bar displays "http://localhost/herdt_ph". The page content is a table with four columns: Land, Hauptstadt, Sprache, and Währung. The data rows are:

Land	Hauptstadt	Sprache	Währung
Japan	Tokio	Japanisch	Yen
Niederlande	Amsterdam	Niederländisch	Euro
Polen	Warschau	Polnisch	Złoty
Indien	Neu Delhi	Indisch	Rupie
Island	Reykjavik	Isländisch	Krone
Italien	Rom	Italienisch	Euro
Frankreich	Paris	Französisch	Euro
Spanien	Madrid	Spanisch	Euro
England	London	Englisch	Pfund Sterling

Ausgabe von Werten eines mehrdimensionalen assoziativen Arrays
(Beispieldatei "array_mehrdimensional_list")

6.10 Den passenden Array-Typ verwenden

Der passende Array-Typ hängt von den Daten ab, die Sie in der Array-Variablen speichern wollen. Die folgende Tabelle soll Ihnen bei der Auswahl des Array-Variablentyps helfen:

Daten	Anzahl an Informationen pro Eintrag	Bevorzugter Array-Typ
Einfache Liste (z. B. Namen, Länder, Automarken etc.)	1	✓ eindimensional/indiziert
Wertepaare (z. B. Länder - Hauptstädte, Artikel - Preis etc.)	2	✓ eindimensional/assoziativ, sofern ein Wert eindeutig ist (= Schlüssel), sonst ✓ mehdimensional/indiziert oder assoziativ
Mehrere evtl. mehrfach verschachtelte Werte (z. B. Mitarbeiter und ihre Stammdaten, Artikelkategorien - Artikel - Artikeldetails etc.)	> 2	✓ mehdimensional/indiziert oder assoziativ

6.11 Weitere Informationen zu Arrays in PHP

Arrays begegnen Ihnen in PHP nicht nur, wenn Sie selbst Array-Variablen definieren. PHP arbeitet intern ebenfalls mit Arrays, und zwar bei folgenden Aktionen, die in den nächsten Kapiteln des Buches besprochen werden:

- ✓ Die in einem HTML-Formular übermittelten Daten speichert PHP in einer Array-Variablen.
- ✓ Wenn Sie mit Sessions arbeiten, werden die zur Session gehörigen Daten automatisch in einem Array abgelegt.
- ✓ Bei Abfragen aus Datenbanken befinden sich die Ergebnisse ebenfalls in einer Array-Variablen.

Weitere Array-Funktionen

PHP kennt annähernd 100 Funktionen für den Umgang mit Arrays. Für nahezu jede Fragestellung liefert PHP eine passende Funktion. Folgende Aufgabengebiete werden abgedeckt:

- ✓ Auslesen
- ✓ Auswerten (u. a. Dopplungen finden)
- ✓ Sortieren (vorwärts, rückwärts, vorwärts nach Schlüssel, rückwärts nach Schlüssel)
- ✓ Suchen
- ✓ Teilen und Zusammensetzen
- ✓ Verändern (u. a. Füllen, Hinzufügen, Löschen)
- ✓ mehrere Arrays zusammenführen
- ✓ neues Array aus den Array-Schlüsseln generieren

Eine Auswahl wichtiger Array-Funktionen wird in der nachfolgenden Tabelle vorgestellt:

Array-Funktion	Beschreibung
array_flip(\$feld)	Im Array werden Indizes mit Werten vertauscht.
array_key_exists(\$wert, \$feld)	Prüfung, ob ein Schlüssel in einem Array vorhanden ist.
array_keys(\$feld)	Liefert die Indizes des angegebenen Arrays zurück.
array_merge(\$feld1, \$feld2...)	Fügt die Elemente mehrerer Arrays zu einem Array zusammen.
array_push(\$feld, \$werte)	Das Array wird um den oder die angegebenen Werte am Ende des Arrays erweitert.
array_search(\$wert, \$feld)	Das Array wird nach dem angegebenen Wert durchsucht.
array_sum(\$feld)	Addiert die Werte des Arrays und liefert das Ergebnis zurück.
array_unique(\$feld)	Es wird ein neues Array erstellt, aus dem doppelte Werte des angegebenen Arrays gelöscht wurden.
array_values(\$feld)	Alle Werte des Arrays werden zurückgeliefert.
count(\$feld)	Gibt die Anzahl der Elemente des Arrays zurück.
sort(\$feld) rsort(\$feld)	Sortiert die Werte des angegebenen Arrays aufsteigend bzw. absteigend.
ksort(\$feld) krsort(\$feld)	Sortiert das angegebene Array nach Schlüssel aufsteigend bzw. absteigend.

Beispiele zu Array-Funktionen finden Sie im Kapitel 10.7.



Einen Überblick über alle Array-Funktionen finden Sie auf der Webseite von [php.net](http://docs.php.net/manual/de/ref.array.php):

6.12 Übung

Arrays in der Theorie

Übungsdatei: –

Ergebnisdatei: **kapA\Antworten (Theorie).doc**

- ① Erklären Sie die Begriffe „eindimensionales“ und „mehrdimensionales Array“ sowie „indiziertes“ und „assoziatives Array“.
- ② Welche Formen des Zugriffs auf Array-Werte kennen Sie? Nennen Sie die Unterschiede.

Mit eindimensionalen Arrays arbeiten

Übungsdatei: –

Ergebnisdatei: *kennzeichen.php*

- ③ Erstellen Sie eine neue PHP-Datei unter dem Namen *kennzeichen.php*. Definieren Sie eine assoziative Arrayvariable, die Sie mit der folgenden Liste von Autokennzeichen und den dazugehörigen Städten füllen:

HH	Hamburg
B	Berlin
S	Stuttgart

- ④ Ergänzen Sie die Liste nach der Erstellung mit den Elementen:

F	Frankfurt
HB	Bremen

Kennzeichen	Stadt
HH	Hamburg
B	Berlin
S	Stuttgart
F	Frankfurt am Main

Beispiel-Ergebnisdatei "kennzeichen.php"

- ⑤ Löschen Sie mithilfe des Befehls `unset()` das Element *Bremen* aus dem Array und definieren Sie den Eintrag *Frankfurt* neu – er soll nun *Frankfurt am Main* heißen.
- ⑥ Listen Sie mithilfe von `foreach` alle Elemente der Tabelle auf und vergeben Sie zusätzlich Überschriften für die Tabelle.

Mit mehrdimensionalen Arrays arbeiten

Übungsdatei: –

Ergebnisdatei: *uebung_mehrdimensional.php*

- ⑦ Erstellen Sie eine neue Datei unter dem Namen *uebung_mehrdimensional.php*. Erstellen Sie ein mehrdimensionales indiziertes Array mit folgenden Inhalten und geben Sie die Daten anschließend in Tabellenform (mit Überschrift) auf dem Bildschirm aus:

Beginn	Disziplin	Ort	Bemerkung
09:30 Uhr	Diskuswurf	Nebenplatz	Jugendmeisterschaften
10:00 Uhr	5-km-Lauf	Stadion - Laufbahn	Offener Lauf
11:00 Uhr	Halbmarathon	Waldgebiet	Teilnahme ab 18 Jahren
12:00 Uhr	Stabhochsprung	Stadion - Stabhochsprunganlage	Nur Frauen

Beginn	Disziplin	Ort	Bemerkung
09.30 Uhr	Diskuswurf	Nebenplatz	Jugendmeisterschaften
10.00 Uhr	5 km-Lauf	Stadion - Laufbahn	Offener Lauf
11.00 Uhr	Halbmarathon	Waldgebiet	Teilnahme ab 18 Jahren
12.00 Uhr	Stabhochsprung	Stadion - Stabhochsprunganlage	Nur Frauen

Beispiel-Ergebnisdatei "uebung_mehrdimensional.php"

7 Mit Formularen arbeiten

In diesem Kapitel erfahren Sie

- ✓ welche Methoden der Datenübergabe möglich sind
- ✓ was beim Aufbau von Formularen zu beachten ist
- ✓ welche Formularelemente welche Werte übertragen
- ✓ wie Sie mit PHP Formulardaten auswerten
- ✓ wie Sie Formulare in PHP einsetzen

Voraussetzungen

- ✓ HTML-Kenntnisse
- ✓ Grundlagen PHP
- ✓ Variablen und Kontrollstrukturen
- ✓ Arrays

7.1 Interaktion mit PHP

Einsatz von Formularen

Formulare sind die Schnittstelle zwischen Benutzer und PHP. Immer, wenn ein Benutzer interagieren möchte, wird er Daten in ein Formular eintragen. Über Formulare können Daten an den Webserver gesendet werden. PHP kann die Eingaben auswerten und entsprechend die nächste Webseite dynamisch und individuell aufbauen. Unabhängig davon, ob Sie ein Login-, Bestell- oder Kontaktformular verwenden, es handelt sich immer um die Übertragung von Nutzereingaben und deren anschließende Verarbeitung durch PHP.

Formularauswertung mit PHP

Mit PHP können Sie interaktive Webseiten erstellen, bei denen Benutzereingaben aus Formularen durch PHP ausgewertet werden:

Nachdem Sie ein Formular aufgerufen haben, können Sie Daten in das Formular eingeben. Wenn das Formular abgesendet wird, werden die eingegebenen Daten an den Webserver übertragen und stehen PHP für die Verarbeitung zur Verfügung. Abhängig von den Eingaben kann PHP individuelle HTML-Ergebnisseiten generieren und sendet diese an Ihren Browser zurück.

Methoden der Datenübertragung

Das **Hypertext Transfer Protocol (HTTP)**, englisch für Hypertext-Übertragungsprotokoll) ist ein Kommunikationsprotokoll, welches für die Übertragung von Daten zwischen Browser und Webserver verantwortlich ist bzw. die Regeln für den Austausch von Daten festlegt.

HTTP sieht zwei Methoden vor, um Daten an einen Webserver zu senden: **POST** und **GET**. Diese beiden Methoden unterscheiden sich z. B. dadurch, wie Formulardaten an den Server übertragen werden oder in der Menge der Daten, die übertragen werden können.

Auf welche Weise Eingaben eines HTML-Formulars an den Webserver gesendet werden, definieren Sie selbst. Das HTML-Element `form` sieht das Attribut `method` vor. Bei der Angabe von `<form method="POST">` legen Sie fest, dass Formulardaten per POST-Methode übertragen werden, entsprechend `<form method="GET">` für die GET-Methode. Die Werte `POST` bzw. `GET` sind nicht Case-sensitiv, `<form method="post">` und `<form method="get">` sind ebenfalls gültige Schreibweisen. Lassen Sie das `method`-Attribut leer oder weg (absichtlich oder versehentlich), verwenden Browser den Standardwert `GET` für `method`, Daten werden dann per GET-Methode übertragen.

Die GET-Methode

Die Eingaben werden nach dem Absenden des Formulars an die URL angehängt. Der URL folgt zuerst das Zeichen `?` als Trennung von der eigentlichen URL und den Eingaben des Formulars. Hinter dem `?` folgen dann alle Daten aus dem Formular. Diese bestehen aus dem Schlüssel (dem Wert des Formularelement-Attributs `name`) und der eigentlichen Eingabe des Nutzers, getrennt durch ein `=`-Zeichen (z. B. `vorname=Max`). Die einzelnen Schlüssel-Wert-Paare werden dann jeweils durch ein „kaufmännisches Und“ `&` voneinander getrennt.

Eine URL nach Absenden eines GET-Formulars könnte wie folgt aussehen:

`http://localhost/antwort.php?vorname=Max&nachname=Mustermann&kennwort=123abc`

Eine solche URL könnte auch als Link auf einer Webseite verwendet werden. Links verwenden also die HTTP-GET-Methode zur Datenübertragung. Der Webserver selbst kann nicht unterscheiden, ob die Werte aus einem Formular mit der GET-Methode oder von einem Link mit Parametern stammen.



Merkmale von GET

- ✓ Angabe von `method="GET"` im einleitenden HTML-Tag `<form>`
- ✓ Formulardaten werden sichtbar und unverschlüsselt in der URL übermittelt
- ✓ Daten sind in der Adresszeile des Browsers veränderbar, ohne dass das Formular erneut ausgefüllt bzw. abgesendet werden muss.
- ✓ Der Aufruf des Skripts mit Angabe der Daten kann als Lesezeichen gespeichert werden.
- ✓ Die Übertragungsmenge gegenüber POST ist wesentlich geringer. Die Begrenzung der Datenmenge wird durch den Browser bestimmt und kann von Browser zu Browser variieren.

Die POST-Methode

Bei der POST-Methode werden Formulardaten als Datenblock bzw. als Datenstrom nach dem HTTP-Header gesendet. In der Adressleiste des Browsers werden keine Werte angezeigt, die Formulardaten werden für den Nutzer nicht sichtbar übertragen. Die Menge der Daten, die übertragen werden können, ist bei der POST-Methode wesentlich größer als bei der GET-Methode. Außerdem ermöglicht ausschließlich die POST-Methode den Upload von Dateien (`<input type="file">`), die Übertragung von Dateien ist mit der GET-Methode nicht möglich.

Merkmale von POST

- ✓ Angabe von `method="POST"` im einleitenden HTML-Tag `<form>`
- ✓ Formulardaten werden direkt an die Webadresse versendet.
- ✓ Daten können nicht in der Adresszeile des Browsers manipuliert werden.
- ✓ Formulardaten werden nicht in der Protokolldatei des Servers gespeichert.
- ✓ Formulardaten sind nicht im Verlauf des Browsers sichtbar.
- ✓ Die Datenmenge, die übertragen werden kann, ist wesentlich größer als bei der GET-Methode. Per Standard können 8 MB übertragen werden, der Wert kann angepasst werden (über die `php.ini` über den Parameter `max_post_size`, vgl. Installationshinweise im Download).
- ✓ Upload von Dateien ist nur über die POST-Methode möglich (hierzu wird im `form`-Tag das Attribut `ENCTYPE="multipart/form-data"` benötigt).

Unterschiede zwischen GET und POST

Der wesentliche Unterschied liegt in der Übertragung in der URL im Gegensatz zu der als Datenstrom. Bei der Übertragung in der URL können Daten in der URL verändert werden, diese bietet die Möglichkeit der Manipulation. Die Möglichkeit, GET-URLs als Favorit zu speichern oder diese per E-Mail zu versenden, birgt die Gefahr, dass diskrete Eingaben wie z. B. Passwörter für andere Nutzer sichtbar werden.

<input type="hidden">Felder (versteckte Formular-Felder im HTML für interne Daten zur Weiterverarbeitung) werden bei der GET-Methode in der URL für den Nutzer direkt sichtbar. Allerdings sind hidden-Felder auch bei der POST-Methode im HTML-Quelltext auszulesen, was die Möglichkeit der Manipulation bietet.

Ein weiterer Unterschied ist das Verhalten der Browser: Wird eine Webseite nach dem Absenden eines GET-Formulars neu geladen (z. B. per F5-Taste), werden die Formulardaten erneut an den Webserver gesendet, hier kann es zur mehrfachen Verarbeitung der gleichen Daten kommen (was z. B. bei einer Bank-Überweisung fatal wäre). Bei der POST-Methode erscheint im Browser ein Dialogfenster mit der Frage *Möchten Sie die Formulardaten erneut senden?* Erst wenn Sie diese Frage bestätigen, wird das Formular nochmal versendet. So erkennen Sie bereits beim Reload einer Seite, welche HTTP-Methode im form-Tag definiert wurde.



Aufgrund der Sichtbarkeit der übertragenen Daten kann die POST-Methode als sicherer als die GET-Methode bewertet werden. Aus diesem Grund wird in diesem Buch vorwiegend die POST-Methode in Formularen verwendet.



Allerdings: Die Wahl der POST-Methode bietet nur eine kleinere Verbesserung der Sicherheit. Auch POST-Formulare können manipuliert werden. Ein Seitenreload kann dazu führen, dass gleiche Eingaben mehrfach an den Webserver gesendet werden. Diese Risiken müssen über eine durchdachte Logik im PHP abgefangen werden.

7.2 Formulare mit PHP auswerten

Formulardaten eingeben

Beispiel: *formular.html*

Erstellen Sie zunächst eine HTML-Datei, in der ein einfaches Formular mit Text-Eingabefeldern integriert ist. In diesem Formular soll der Nutzer seinen Vor- und Nachnamen sowie seinen Wohnort angeben. Im Formular sollen Eingaben in Textfelder erfolgen.

```
<!DOCTYPE html>
<html>
  <head>
    <title>einfaches Formular</title>
  </head>
  <body>
    <h1>Anmeldung</h1>
    <p>Bitte füllen Sie die nachfolgenden Eingabefelder aus: </p>
    <form action="formular_auswertung.php" method="POST">
      <p>Vorname: <input type="text" name="vorname"></p>
      <p>Nachname: <input type="text" name="nachname"></p>
      <p>Wohnort: <input type="text" name="ort"></p>
      <p><input type="submit" value="Abschicken">
        <input type="reset" value="Zurücksetzen"></p>
    </form>
  </body>
</html>
```

- ① Im HTML wird das Formular über das `<form>`-Tag definiert. Das Skript, das die Formulardaten auswerten und verarbeiten soll, wird über das Attribut `action` festgelegt. In diesem Fall werden die Daten an das PHP-Skript `formular_auswertung.php` gesendet. Die POST-Methode zur Übermittlung der Daten wird mittels `method="POST"` festgelegt. Wichtig ist, dass alle Formularfelder innerhalb des öffnenden und schließenden `form`-Tags liegen. Auch die Schaltfläche *Abschicken* muss innerhalb der `form`-Tags stehen.
- ② Es folgen die einzeiligen Texteingabefelder vom `input`-Type `text`. Falls Sie das `type`-Attribut weg oder leer lassen, stellen alle Browser automatisch ein einzeiliges Eingabefeld dar. Die Felder werden dabei über das HTML-Attribut `name` als `vorname`, `nachname` und `ort` gekennzeichnet. Nach dem Absenden des Formulars sind die `name`-Attribute die Schlüssel des `$_POST`- bzw. `$_GET`-Arrays, welche für die weitere Verarbeitung zur Verfügung stehen.
- ③ Es folgen die Standardschaltflächen zum *Abschicken* und *Zurücksetzen* der Formulardaten. Die Schaltfläche *Zurücksetzen* ist optional und dient lediglich der leichteren Bedienung des Formulars durch den Benutzer, falls dieser seine Eingaben mit einem Klick löschen möchte.

Einzeilige Eingabefelder (auch vom Type `email`, `number`, `password` usw.), Selectboxen, mehrzeilige Textfelder sowie Radiobuttons und Checkboxen eines HTML-Formulars müssen durch das Attribut `name` eindeutig gekennzeichnet sein. Der Wert von `name` des Formularfeldes wird später als Schlüssel im Array `$_POST` bzw. `$_GET` verwendet.



Beispielformular "formular.html"

Formulardaten übertragen

Mit Klick auf die Schaltfläche *Abschicken* werden die eingegebenen Formulardaten an den Webserver gesendet. Im Formular-Attribut `action` geben Sie das PHP-Skript an, an welches die Daten zur Verarbeitung gesendet werden sollen (`formular_auswertung.php`). Dieses Skript soll nun erstellt werden.

Formulardaten auswerten

Nach dem Absenden des Formulars stellt der Webserver die Übertragung von Formulardaten fest und stellt diese für das PHP-Skript je nach Übertragungsmethode in dem superglobalen Array `$_POST` bzw. `$_GET` zur Verfügung. Superglobal bedeutet, das Array ist allgemein in PHP verfügbar (mehr dazu im folgenden Kapitel). Dabei ist der Schlüssel des Arrays der Wert des HTML-Attributs `name`, der Wert selbst ist dann die Eingabe des Nutzers.

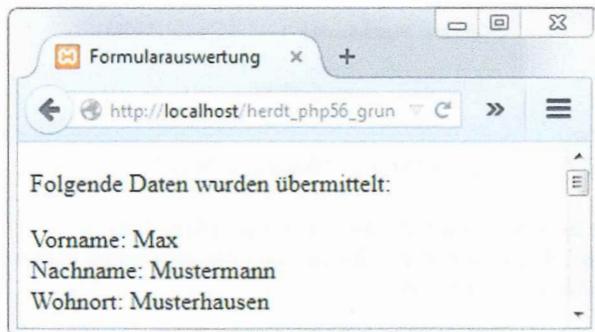
Darüber hinaus hält PHP die übertragenen Werte in dem von PHP definierten Array `$GLOBALS` bereit. Je nach Übertragungsmethode können Sie entweder auf die Arrays `$GLOBALS["_GET"]` bzw. `$GLOBALS["_POST"]` zugreifen. Zusätzlich können Sie die Array-Variable `$_REQUEST` verwenden. In diesem Array stehen die Formulardaten unabhängig von der Übertragungsmethode zur Verfügung. Allerdings sollten Sie immer das `$_POST`- bzw. `$_GET`-Array ansprechen. Das ist guter Stil und ein Aspekt sicherer Programmierung. Damit stellen Sie sicher, dass Formulardaten auch aus dem Formular stammen, das Sie selbst programmiert haben.

Die übermittelten Daten sprechen Sie über die Array-Variable und den Schlüssel an, wie Sie es im Kapitel 6 gelernt haben: `$_POST["Schlüssel"]`. Der Wert des HTML-`input`-Attributs `name` (z. B. `name="vorname"`) wird als Schlüssel der Array-Variablen `$_POST` verwendet, z. B. `$_POST["vorname"]`. Die Eingabe im Formular bildet den dazugehörigen Wert der Array-Variablen, z. B. `$_POST["vorname"] = "Max"`.

Beispiel: *formular_auswertung.php*

Sie erstellen ein PHP-Skript, das die Eingaben des HTML-Formulars ausliest und zur Kontrolle am Bildschirm ausgibt.

```
<?php
echo "<p>Folgende Daten wurden übermittelt:</p>";
echo " <p>Vorname: " . $_POST["vorname"] . "<br>";
echo "Nachname: " . $_POST["nachname"] . "<br>";
echo "Wohnort: " . $_POST["ort"] . "</p>";
?>
```



Ausgabe der Beispieldatei "formular_auswertung.php"

Ausgabe der übertragenen Formulardaten

PHP bietet unterschiedliche Funktionen, mit der Sie sich übermittelte Formulardaten im Browser anzeigen lassen können:

```
print_r(Variable);
var_dump(Variable 1[, Variable 2,...]);
```

Erweitern Sie das PHP-Skript um folgende Zeilen. Sowohl `print_r()` als auch `var_dump()` dienen der Darstellung von Array-Variablen im Browser. Das `pre`-Tag, welches Sie über den `echo`-Befehl ausgeben, dient der zeilenweisen Darstellung der Arrays. Ohne das `pre`-Tag werden die Array-Daten in einer Zeile nacheinander und schwerer zu lesen dargestellt.

```
echo "<pre>";
print_r($_POST);
var_dump($_POST);
echo "</pre>";
```

Erweiterung der Beispieldatei "formular_auswertung.php" um die angegebenen Zeilen

Die Ausgabe von `print_r()` sehen Sie in nebenstehender Abbildung: Die Schlüssel des `$_POST`-Arrays werden in Klammern angezeigt (sie entsprechen den name-Attributen der Formularelemente), daneben die Eingaben, die der Nutzer gemacht hat.

`var_dump()` liefert umfangreichere Informationen über das `$_POST`-Array. Neben Schlüssel und Wert wird angezeigt, wie viele Einträge das `$_POST`-Array hat, wie viele Zeichen jede einzelne Eingabe hat und von welchem Datentyp der Wert eines Array-Eintrags ist (allerdings handelt es sich im Falle von Formulardaten auch um den Datentyp `string`, selbst wenn Ganz- oder Fließzahlen eingegeben wurden).

Verschiedene Formularelemente

Die wichtigsten Formularelemente in HTML sind die Elemente `input`, `select` und `option`, `textarea` und `button`. Das `input`-Element ist dabei hervorzuheben, da erst durch den Wert des `type`-Attributs das Eingabefeld zu einem speziellen Feld wird, z. B. `type="radio"` wird zum Radiobutton, `type="checkbox"` zur Checkbox, `type="file"` zum Upload-Button, `type="email"` wird zum E-Mail-Feld (Browser validieren die E-Mail, auf mobilen Geräten wird die Tastatur mit den relevanten Zeichen angezeigt), `type="submit"` wird zum Absende-Button.

Mit HTML5 sind weitere `type`-Attributwerte hinzugekommen, die Formulare speziell auf die zu erwartenden Eingaben anpassbar machen, um die Bedienbarkeit zu verbessern (z. B. `type="date"`, erzeugt ein Eingabefeld, welches einen Auswahlkalender anzeigt. Allerdings wird dies noch nicht von allen Browsern unterstützt).

Beispiel: `form-elemente.html`

In dieser HTML-Datei finden Sie eine Reihe von unterschiedlichen Formular-Elementen. Obwohl die Menge aller Formular-Elemente überschaubar ist, verändert sich die Darstellung besonders beim `type`-Attribut für das `input`-Feld. Aber nicht nur die Ansicht im Browser ist verschieden, auch bei der Vergabe des `name`-Attributs und bei Übergabe der Eingaben an die Zielseite sind ein paar Dinge zu beachten.

```

① <form action="form-elemente.php" method="get">
②   <p>E-Mail: <input type="email" name="email"></p>
③   <p>Passwort: <input type="password" name="password"></p>
④   <p>Datum: <input type="date" name="datum"></p>
⑤   <p>Farben: <br>
        <input type="checkbox" name="gelb"> gelb
        <input type="checkbox" name="blau"> blau
        <input type="checkbox" name="gruen"> gruen
        <input type="checkbox" name="rot" value="#f00"> rot </p>
⑥   <p>Ampel: <br>
        <input type="radio" name="ampel" value="gruen"> grün
        <input type="radio" name="ampel" value="gelb"> gelb
        <input type="radio" name="ampel" value="rot"> rot </p>
⑦   <p>Speisen: <br>
        <input type="checkbox" name="speisen[]"> Pizza
        <input type="checkbox" name="speisen[]"> Nudeln
        <input type="checkbox" name="speisen[]" value="salat"> Salat </p>
⑧   <p> Eis: <select name="eissorte">
        <option></option>
        <option>Schoko</option>
        <option>Vanille</option>
        <option>Nuss</option>
    </select> </p>

```

```

Informationen über Variablen mit print_r():
Array
(
    [vorname] => Max
    [nachname] => Mustermann
    [ort] => Musterhausen
)

Informationen über Variablen mit var_dump():
array(3) {
    ["vorname"]=>
        string(3) "Max"
    ["nachname"]=>
        string(10) "Mustermann"
    ["ort"]=>
        string(12) "Musterhausen"
}

```

Anzeige der Daten, die aus dem Formular übertragen werden

```

⑦ <p> Gemüse: <select name="gemuese">
    <option></option>
    <option value="sorte-1">Bohnen</option>
    <option value="sorte-2">Erbsen</option>
    <option value="sorte-3">Blumenkohl</option>
  </select> </p>
⑧ <p> Obst: <select name="obst[]" multiple>
    <option>Apfel</option>
    <option>Birne</option>
    <option>Pflaume</option>
    <option>Orange</option>
  </select> </p>
⑨ <p>Nachricht:<br><textarea name="memo"></textarea></p>
⑩ <p><input type="submit" value="Abschicken" name="senden"></p>
⑪ </form>

```

- ① Die Übertragungsmethode wurde hier bewusst mit der GET-Methode definiert. Schauen Sie sich nach Absenden des Formulars die URL an, z. B. finden Sie dort die Eingabe des Passworts im Klartext. Achten Sie darauf, dass alle Formularfelder einschließlich des Absende-Buttons innerhalb des öffnenden und schließenden form-Tags liegen.
- ② Hier werden drei input-Elemente mit unterschiedlichen type-Attributten hinterlegt. Optisch sehen alle drei aus wie ein einziges Eingabefeld. Allerdings unterscheiden Sie sich in der Bedienung durch den Nutzer. Ein input-Element vom type="email" überprüft je nach Browser die Eingabe auf eine gültige Syntax, mit dem type="date" wird je nach Browser ein Kalenderauswahlfeld angezeigt. Bei allen drei Feldern ist das name-Attribut vergeben, welches als Schlüssel im \$_GET-Array wieder zu finden ist. Bei allen drei Feldern wird auf jeden Fall ein Schlüssel übergeben, auch wenn der Nutzer keine Eingabe vorgenommen hat.
- ③ Über das type-Attribut checkbox erzeugen Sie hier vier Checkboxen. Als name-Attribut vergeben Sie auch hier einen Wert, den Sie als Schlüssel im \$_GET-Array wieder finden. Beachten Sie: Bei der vierten Checkbox ist zusätzlich der value="#f00" angegeben. In dem Fall, und falls die Checkbox ausgewählt wird, wird mit dem Schlüssel rot auch der Wert #f00 übergeben. Bei den anderen drei Boxen fehlt ein value-Attribut. In diesen Fällen und bei Auswahl der Boxen ist der Wert zum Schlüssel im \$_GET-Array lediglich on. Mit value zu arbeiten ist z. B. sinnvoll, wenn Sie ein Produkt zur Auswahl anbieten, als value dann aber die Produkt-ID verwenden wollen, und nicht mit der Produktbezeichnung, die zur Ansicht im Browser dient. Beachten Sie zusätzlich: Ist eine Checkbox nicht ausgewählt, werden weder Schlüssel noch ein möglicher Wert übertragen.
- ④ Über den Wert radio für das type-Attribut definieren Sie hier Radiobuttons. Damit diese miteinander korrespondieren, also nur ein Radiobutton von den dreien auswählbar ist, müssen alle dasselbe name-Attribut haben. Um die Auswahl im PHP-Code erkennen zu können, muss bei einem Radiobutton immer auch ein value vergeben sein. Auch für Radiobutton gilt: Wenn der Nutzer keine Auswahl trifft, werden weder Schlüssel noch Wert übertragen.

Verschiedene Formularelemente

Bitte füllen Sie die nachfolgenden Eingabefelder aus:

E-Mail: _____

Passwort: *****

Datum: _____

Farben:

gelb blau gruen rot

Ampel:

grün gelb rot

Speisen:

Pizza Nudeln Salat

Eis: _____

Gemüse:

Apfel
Birne
Pflaume

Obst: Orange

Nachricht:

Abschicken

Ausgabe der Beispieldatei
"form-elemente.html"

- ⑤ Hier werden weitere Checkboxen hinterlegt. Im Unterschied zu den Checkboxen in ③ wird nicht für jede Checkbox ein individuelles name-Attribut angegeben, sondern über den gleichen Wert speisen und die eckigen Klammern die Checkboxen gruppiert. Im \$_GET-Array wird die Auswahl dieser Checkboxen in einem inneren indizierten Array mit dem Schlüssel speisen zusammengefasst. Auch hier gilt, wenn kein value vergeben wurde, erscheint im \$_GET-Array lediglich ein on zu dem nicht aussagekräftigen Schlüssel des indizierten Array (Der Eintrag \$_GET["speisen"] [0] entspricht der ersten Checkbox im HTML-Code mit dem name-Attribut speisen[]). Erst ein value liefert einen verwertbaren Wert.
- ⑥ In dieser Zeile wird eine Auswahlbox mit dem HTML-Element select definiert. Die einzelnen Auswahlmöglichkeiten werden über jeweils ein option-Tag hinterlegt. Der ausgewählte Eintrag erscheint mit dem name des select-Elements als Schlüssel im \$_GET-Array. Bei Auswahlboxen wird immer ein Schlüssel übertragen, auch wenn der Nutzer keine Auswahl getroffen hat.
- ⑦ Vergleichbar zu ⑥, allerdings haben hier die option-Tags value-Attribute. In dem Fall wird nicht der Wert übertragen, den der Nutzer sieht, sondern der Wert von value, den Sie hinterlegt haben.
- ⑧ Hier wird eine weitere Auswahlbox definiert, im Unterschied zu den vorherigen hat das select-Element das Attribut multiple. Dies erlaubt dem Nutzer, mehrere Optionen aus der Auswahlbox zu wählen. Damit im \$_GET-Array alle ausgewählten Einträge übermittelt werden, muss das name-Attribut des select-Elements eckige Klammern haben, damit die Werte als Array übertragen werden. Wird als name ein Schlüssel ohne die Klammern angegeben, wird nur die letzte ausgewählte Option als Wert übertragen.
- ⑨ Hier wird ein mehrzeiliges Eingabefeld definiert. Die HTML-Syntax unterscheidet sich von den meisten anderen Formularelementen, da nicht nur ein öffnendes textarea-Tag benötigt wird, sondern immer auch ein schließendes. Bei textarea-Elementen wird ein Schlüssel übertragen, auch wenn der Nutzer keine Eingabe vorgenommen hat.

HTML5 bietet neue Formularelemente, die zum Teil mit einer implizierten Validierung einhergehen, z. B. werden E-Mail-Adressen auf Gültigkeit überprüft. Als PHP-Entwickler dürfen Sie sich nicht auf diese Validierung im Browser verlassen. Einerseits beherrschen noch nicht alle Browser HTML5 oder nur im begrenzten Umfang, andererseits sind die Validierungen im Browser sehr minimalistisch. Die Eingabe von aaa@bbb wird als gültige E-Mail-Adresse erkannt. Die Korrektheit der Eingaben muss immer serverseitig mit PHP überprüft werden.



Formulare mit Checkboxen und Radiobuttons

Checkboxen und Radiobuttons bieten eine Besonderheit, die Sie bei der Auswertung mit PHP berücksichtigen müssen: Diese name-Attribute werden aus dem Formular nur übermittelt, wenn sie ausgewählt bzw. angeklickt wurden. Ein anderes Verhalten wird bei der Übertragung von einzeiligen oder mehrzeiligen Eingabefeldern und Auswahlboxen deutlich: Diese name-Attribute werden als Schlüssel immer übermittelt, der Wert bleibt jedoch leer, falls kein Eintrag vorgenommen wurde.

Beispiel: **formular-2.html**

Es wird zunächst eine HTML-Datei erstellt, in der ein einfaches Formular mit einer Gruppe von Checkboxen und einer Gruppe von Radiobuttons erstellt wird.

```

Übertragene Daten
Array
(
    [email] => demo@demo.xx
    [password] => 123abc
    [datum] => 28.02.15
    [blau] => on
    [gruen] => on
    [rot] => #f00
    [ampel] => rot
    [speisen] => Array
        (
            [0] => on
            [1] => salat
        )

    [eissorte] => Nuss
    [gemuese] => sorte-3
    [obst] => Array
        (
            [0] => Birne
            [1] => Pflaume
            [2] => Orange
        )

    [memo] => Test Eintrag
    [senden] => Abschicken
)

```

[Zurück zur Eingabe](#)

Ausgabe des \$_GET-Array in der Datei
"form-elemente.php"

```

① <form action="formular_auswertung-2.php" method="POST">
    <p>Interessen:
        <input type="checkbox" name="interesse[]" value="Kultur"> Kultur
        <input type="checkbox" name="interesse[]" value="Musik"> Musik
        <input type="checkbox" name="interesse[]" value="Natur"> Natur
        <input type="checkbox" name="interesse[]" value="Sport"> Sport
    </p>
    <p>Zahlungsart:
        <input type="radio" name="zahlung" value="bar"> bar
        <input type="radio" name="zahlung" value="Scheck"> Scheck
        <input type="radio" name="zahlung" value="Überweisung"> Überweisung
    </p>
    <p><input type="submit" name="absenden" value="Abschicken">
        <input type="reset" value="Zurücksetzen"></p>
</form>

```

- ① Um Checkboxen für die Auswertung mit PHP vorzubereiten, muss im Formular jede Gruppe von Checkboxen den gleichen Namen – und zusätzlich folgende eckige Klammern **[]** – erhalten. Der Eingabename `interesse` wird damit als Array-Variable gekennzeichnet, die mehrere Werte aufnehmen kann.
- ② Im Formular wird eine Gruppe von Radiobuttons bereitgestellt. Hier ist für alle drei Radiobuttons dasselbe `name`-Attribut vergeben, damit diese miteinander korrespondieren. Für jeden Radiobutton wird ein `value`-Attribut vergeben, dessen Wert bei der Auswahl übergeben wird.



Formular, aus dem Daten übertragen werden

Beispiel: `formular_auswertung-2.php`

```

<?php
    echo "<pre>";
    print_r($_POST);
    echo "</pre>";
② if (!empty($_POST["interesse"])) {
    echo "<p>Folgende Interessen wurden angegeben:<br>";
    echo implode(", ", $_POST["interesse"]) . "</p>";
}
③ ?>

```

- ① Mit `print_r()` wird ausgegeben, welche Daten aus dem Formular übermittelt wurden.
- ② Es erfolgt die Prüfung, ob eine Checkbox im Formular ausgewählt wurde. Falls der Nutzer eine Auswahl getroffen hat, fällt diese Prüfung positiv aus, der folgende Anweisungsblock wird dann ausgeführt.
- ③ Über die Array-Funktion `implode()` werden die Werte aus der Variablen `$_POST["interesse"]` ausgelesen, mit Komma getrennt zusammengeführt und als Zeichenkette ausgegeben.

```

Array
(
    [interesse] => Array
        (
            [0] => Kultur
            [1] => Musik
            [2] => Natur
            [3] => Sport
        )

    [absenden] => Abschicken
)

Folgende Interessen wurden angegeben:
Kultur, Musik, Natur, Sport

```

Anzeige der Beispieldatei "formular_auswertung-2.php"

Da im Formular kein Radiobutton ausgewählt wurde, fehlt die Variable `$_POST["zahlung"]` in der Liste der übermittelten Variablen. Die Schaltfläche zum Absenden des Formulars hingegen wird in dieser Liste aufgeführt.

Prüfung auf Existenz und Inhalt übergebener Schlüssel und Werte

Um Formulardaten flexibel auswerten zu können, können Sie mit PHP abfragen, ob

- ✓ Schlüssel vorhanden sind, also übertragen wurden oder
- ✓ Werte leer sind oder einen bestimmten Wert haben.

Mit `isset()` prüfen Sie die Existenz einer in Klammern angegebenen Variablen. Existiert die Variable, wird TRUE zurückgeliefert, ansonsten FALSE. `isset()` wird häufig in Verbindung mit der `if`-Anweisung verwendet, damit ein Anweisungsblock, der die Eingaben verarbeitet, nicht mit leeren Werten arbeitet, in einen Fehler läuft und gegebenenfalls zur Anzeige einer Warnung im Browser führt. Zur Prüfung mit `isset()` eignen sich folgende Formularelemente:

- ✓ Checkboxen

- ✓ Radiobuttons

```

isset(Variablen);
if(isset(Variablen)) {
    Anweisungsblock;
}

```

```

empty(Variablen);
if(!empty(Variablen)) {
    Anweisungsblock;
}

```

- ✓ Submit-Schaltflächen

Mit `empty()` prüfen Sie, ob eine angegebene Variable einen Wert enthält. Ist die geprüfte Variable nicht leer und hat sie einen von 0 unterschiedlichen Wert, liefert `empty()` FALSE zurück, ansonsten TRUE.

Text-Eingabefelder werden bei der Übertragung von Formulardaten immer übermittelt, auch wenn kein Wert eingetragen wurde. In diesen Fällen hilft die Überprüfung mit `empty()`. Dies gehört zu den gängigen Methoden, um zu prüfen, ob Pflichtfelder in Formularen gefüllt wurden.

Leerzeichen in Text-Eingabefeldern werden übertragen und sind als Werte im `$_POST-` bzw. `$_GET-`Array gespeichert. Eine Prüfung mit `empty()` schlägt hier fehl, auch wenn Sie im Browser keinen Eintrag sehen. Die Prüfung eines Leerzeichens beantwortet `empty()` mit FALSE. Hier empfiehlt es sich, die Prüfung um `trim()` zu erweitern. Diese PHP-Funktion entfernt Leerzeichen und Zeilenumbrüche am Anfang und Ende von Zeichenketten. `if (empty(trim($_POST["nachname"])))` liefert TRUE, auch wenn dort ausschließlich Leerzeichen eingegeben wurden (mehr zu `trim()` siehe Kapitel 10.6).



Mehrere Absende-Schaltflächen

Mit PHP können Sie prüfen, welche Submit-Schaltfläche gedrückt wurde, die ein Formular abgesendet hat. Zu diesem Zweck integrieren Sie in Ihr Formular beliebig viele Submit-Schaltflächen und geben den Schaltflächen eindeutige Namen, z. B.:

```
<input type="submit" name="eins">
<input type="submit" name="zwei">
```

In der auswertenden Datei können Sie gezielt abfragen, welche der Schaltflächen gedrückt wurde. Da eine Submit-Schaltfläche das Formular umgehend übermittelt, kann immer nur genau eine Schaltfläche angeklickt worden sein. Formulare mit mehreren Submit-Schaltflächen sind eine Möglichkeit der Umsetzung, wenn dasselbe Formular verschiedene Aktionen auslösen kann.

```
[gemeinsame Anweisungen]
if(isset($_POST["eins"])){
    Anweisungsblock, wenn Schaltfläche eins gedrückt wurde
}
if(isset($_POST["zwei"])){
    Anweisungsblock, wenn Schaltfläche zwei gedrückt wurde
}
[gemeinsame Anweisungen]
```

Beispiel: *form_multi.html*

Es wird zunächst eine HTML-Datei erstellt, die mehrere Submit-Schaltflächen zum Absenden des Formulars hat. In der auswertenden Datei wird geprüft, welche Schaltfläche betätigt wurde, sodass eine entsprechende Aktion ausgeführt werden kann.

①	<h1>Berechnungen mit zwei Zahlen</h1> <p>Bitte geben Sie zwei Zahlen ein: </p> <form action="form_multi-auswertung.php" method="POST"> <p>Erste Zahl: <input type="text" name="zahl1"></p> <p>Zweite Zahl: <input type="text" name="zahl2"></p> <p><input type="submit" name="mal" value="Zahlen multiplizieren"> <input type="submit" name="plus" value="Zahlen addieren"></p> </form>
---	---

Beispieldatei "form_multi.html"

- ① Über das `action`-Attribut im `form`-Tag wird die Ziel-PHP-Datei festgelegt, an welche die Formulardaten übermittelt werden sollen.
- ② Es werden zwei Submit-Schaltflächen in das Formular integriert. Über die Schaltfläche mit der Bezeichnung `mal` sollen die beiden eingegebenen Zahlen miteinander multipliziert werden.
- ③ Mit der Schaltfläche `plus` sollen die Zahlen hingegen addiert werden.

Beispiel: *form_multi-auswertung.php*

①	<?php echo "<h1>Rechenergebnis</h1>"; if (isset(\$_POST["mal"])) { \$ergebnis = \$_POST["zahl1"] * \$_POST["zahl2"]; echo "<p>" . \$_POST["zahl1"] . " mal " . \$_POST["zahl2"] . " ist gleich \$ergebnis.</p>"; } if (isset(\$_POST["plus"])) { \$ergebnis = \$_POST["zahl1"] + \$_POST["zahl2"]; echo "<p>" . \$_POST["zahl1"] . " plus " . \$_POST["zahl2"] . " ist gleich \$ergebnis.</p>"; } ?>
---	--

- ① Die Überschrift wird über eine echo-Anweisung ausgegeben.
- ② Mithilfe der Funktion `isset()` wird geprüft, ob die Variable `$_POST["mal"]` übermittelt wurde. Wenn die Prüfung TRUE ergibt, wurde die Schaltfläche **Zahlen multiplizieren** zum Absenden des Formulars verwendet. Der folgende Anweisungsblock wird ausgeführt.
- ③ Es wird geprüft, ob die Variable `$_POST["plus"]` vorhanden ist. Wenn die Prüfung TRUE ergibt, wurde die Schaltfläche **Zahlen addieren** verwendet. Der folgende Anweisungsblock wird ausgeführt, die Werte werden miteinander addiert und ausgegeben.

The left screenshot shows a form titled "Berechnungen mit zwei Zahlen". It contains two input fields: "Erste Zahl: 45" and "Zweite Zahl: 262". Below the inputs are two buttons: "Zahlen multiplizieren" and "Zahlen addieren". The right screenshot shows the result page titled "Rechenergebnis" with the text "45 plus 262 ist gleich 307."

Anzeige der Beispieldateien "form_multi.html" und "form_multi-auswertung.php" nach Absenden des Formulars über die Schaltfläche "Zahlen addieren"

Dieses Beispiel setzt voraus, dass der Nutzer eine Schaltfläche drückt. Ein Formular kann jedoch auch über die Return- bzw. Enter-Taste abgesendet werden (wenn sich der Cursor in einem Eingabefeld befindet). Beim Absenden durch Bedienen der Return-Taste wird nur der Wert einer Submit-Schaltfläche übermittelt. Hier verhalten sich die verschiedenen Browser teilweise unterschiedlich.



Formular und Auswertung in derselben PHP-Datei

In der Praxis werden oftmals PHP-Dateien verwendet, die nicht nur das Formular, sondern auch den Code zur Verarbeitung des Formulars beinhalten. Im `action`-Parameter des HTML-Tags `<form>` ist keine fremde Datei angegeben, sondern der Verweis „auf sich selbst“. In größeren Projekten können Sie so einige Seiten einsparen, außerdem können Sie Hinweise auf Fehler, die Sie bei der Auswertung der Formulardaten ermittelt haben, direkt in das Formular einbauen, was die umständliche Übergabe der Fehler und das erneute Laden der Ausgangsdatei überflüssig macht. Die Datei muss die Dateinamenerweiterung `*.php` haben.

Um mit einem Formular auf dieselbe Datei zu verweisen, in dem das Formular hinterlegt ist, haben Sie über das `action`-Attribut des HTML-Tags `<form>` folgende Möglichkeiten. Notieren Sie im `action`-Attribut:

- ✓ den eigenen Dateinamen, z. B. `formular.php`
- ✓ die PHP-Variable `$_SERVER["PHP_SELF"]` (`$_SERVER` gehört zu den superglobalen Variablen, vergleichbar zu den `$_POST`-, `$_GET` oder `$GLOBALS`-Variablen und liefert Informationen über den Webserver. Der Eintrag `PHP_SELF` enthält den Namen der Datei, die aufgerufen ist)
- ✓ `[]`, also `action="?"` (nicht empfohlen)
- ✓ Leerwert, also `action=""` (nicht empfohlen)

Beispiel: `selbstverweis.php`

Es wird eine PHP-Datei mit einem Formular erstellt. Mit dem Formular können Sie die Hintergrundfarbe der Datei steuern. Durch Absenden des Formulars wird die Seite selbst mit der gewählten Hintergrundfarbe aufgerufen.

```

① <body bgcolor="<?php if (isset($_POST["hintergrund"])) echo
$_POST["hintergrund"]; ?>">
    <h1>Hintergrundfarbe wählen</h1>
    Wählen Sie eine Hintergrundfarbe für die Datei aus:
    <form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="POST">
        <p><input type="radio" name="hintergrund" value="#FFFF00"> gelb
           <input type="radio" name="hintergrund" value="#8FEC95"> hellgrün
           <input type="radio" name="hintergrund" value="#FFA000"> orange
           <input type="radio" name="hintergrund" value="#FFFFFF"> weiß</p>
        <p><input type="submit" name="absenden" value="Absenden"></p>
    </form>
    <?php
        if (isset($_POST["absenden"]) && isset($_POST["hintergrund"])) {
            echo "<p><em>Ihre Auswahl wird als Hintergrundfarbe angezeigt.</em></p>";
        }
    ?>
</body>

```

- ① Im HTML-Tag `<body>` wird über PHP der Wert für den Parameter `bgcolor` gesetzt und damit die Hintergrundfarbe für diese Datei festgelegt. Dies geschieht in Abhängigkeit von der aus dem Formular übermittelten Variable `hintergrund`. Damit es nicht zu einer PHP-*notice* im Browser kommt, wird zusätzlich über `isset()` abgefragt, ob die Variable auch gesetzt ist.
- ② Im `action`-Parameter des HTML-Tags `<form>` wird angegeben, dass das Formular „an sich selbst“ gesendet werden soll. Dieselbe Datei mit dem HTML-Formular wird auch zur Auswertung der übertragenen Formulardaten verwendet.
- ③ Es wird geprüft, ob die Variablen `absenden` und `hintergrund` aus dem Formular übermittelt wurden. Die Prüfungen sind verknüpft: Nur wenn beide Prüfungen `TRUE` zurückliefern (also das Formular abgesendet UND eine Farbe ausgewählt wurde), wird die anschließende `echo`-Anweisung ausgeführt.



Anzeige der Beispieldatei "selbstverweis.php"

7.3 Übung

HTML-Formular und Zugriff durch PHP (Theorie)

Übungsdatei: --

Ergebnisdatei: *kap01\Antworten (Theorie).doc*

- ① Nennen Sie die Unterschiede der beiden Übertragungsmethoden POST und GET.
- ② Erklären Sie, wie Sie in einem PHP-Skript auf die übermittelten Daten des HTML-Formulars zugreifen können. Was ist diesbezüglich bei der Erstellung des Formulars zu beachten?

Ein Formular für eine Umfrage erstellen

Übungsdatei: --

Ergebnisdateien: *uebung_formular.html*,
uebung_auswertung.php

- ③ Erstellen Sie das nachstehende Formular für die Umfrage und speichern Sie es unter dem Namen *uebung_formular.html*.

Übung (Kapitel 7) x +

http://localhost/herdt_php56_grundlagen/uebt C Google

Bitte füllen Sie das Formular komplett aus

Persönliche Daten

Vorname: _____

Nachname: _____

Wohnort: _____

Was wir wissen wollen

Wie wohnen Sie? Einfamilienhaus Eigentumswohnung Mehrfamilienhaus

Welche TV-Sendungen sehen Sie gern? Dokumentationen Nachrichten Spielfilme Sport

Haben Sie noch eine Nachricht für uns?

Daten speichern **Zurücksetzen**

Ausgabe "uebung_formular.html"

- ④ Geben Sie nach dem Absenden des Formulars eine Bestätigungsseite aus. Diese Bestätigungsseite soll die vorher eingegebenen Daten zeigen. Speichern Sie die Datei unter dem Namen *uebung_auswertung.php*.

Auswertung des Formulars

Vielen Dank für die Teilnahme an unserer Umfrage. Sie haben folgende Daten übermittelt:

Vorname: Max
 Nachname: Mustermann
 Ort: Musterhausen
 Wohnung: Eigentumswohnung
 Beliebte TV-Sendungen: Dokumentationen, Nachrichten, Spielfilme
 Nachricht: keine

Ausgabe der Ergebnisdatei "uebung_auswertung.php"

Informationen über ein Formular anfordern

Übungsdatei: --

Ergebnisdatei: uebung_formular2.php

- ⑤ Erstellen Sie eine neue PHP-Datei (*uebung_formular2.php*) mit dem nachstehenden Formular. Nach dem Absenden des Formulars soll die Auswertung der Formulardaten in derselben Datei vorgenommen werden. Diese Bestätigungsseite soll die vorher eingegebenen Daten zeigen.

Bewerbung, Newsletter oder Infomaterial

Bitte nennen Sie uns Ihr Anliegen:

Anrede: Herr Frau

Vorname: Max

Nachname: Mustermann

Mailadresse: max@mustermann.com

Herzlichen Dank, Herr Mustermann, für Ihre Bewerbungsanfrage. Unsere Personalabteilung wird per Mail - an Ihre Adresse max@mustermann.com - Kontakt zu Ihnen aufnehmen.

Ausgabe der Beispiellösung (Datei "uebung_formular2.php")

8 Funktionen

In diesem Kapitel erfahren Sie

- ✓ welche Vorteile benutzerdefinierte Funktionen haben
- ✓ wie Sie mit benutzerdefinierten Funktionen arbeiten
- ✓ was der Unterschied zwischen einer lokalen, einer globalen und einer superglobalen Variablen ist
- ✓ wie Sie Dateien mit `include` und `require` einbinden

Voraussetzungen

- ✓ Variablen und Kontrollstrukturen

8.1 Funktionen erstellen und aufrufen

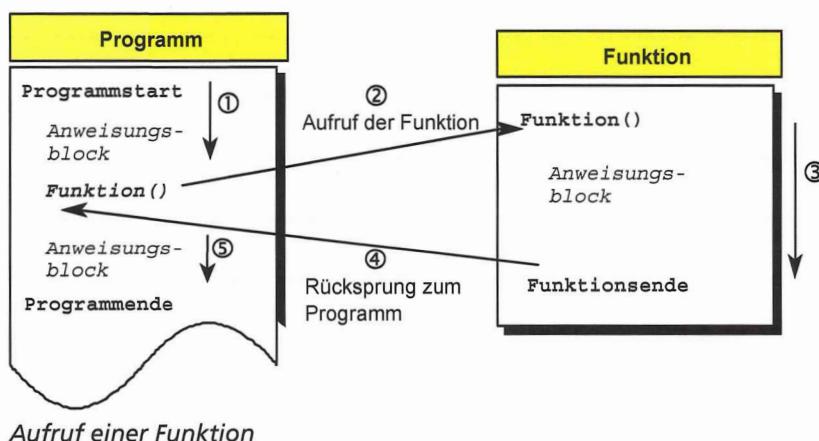
Was sind Funktionen?

Funktionen sind eigenständige Programmteile, die vom Skript beliebig oft aufgerufen und abgearbeitet werden können. Funktionen beinhalten Anweisungen, die innerhalb des Programms oder innerhalb eines Projekts mit mehreren Programmdateien mehrmals benötigt werden. Anstatt die Anweisungen mehrfach im Programm zu codieren, wird die entsprechende Funktion einmalig definiert und an den gewünschten Stellen aufgerufen, um die Anweisungen der Funktion dort auszuführen.

PHP bietet eine Vielzahl vordefinierter Funktionen, um bestimmte Standardaufgaben zu lösen. Funktionen, die Sie selbst zur Lösung Ihrer Aufgaben erstellen, werden benutzerdefinierte Funktionen genannt.

Vorteile von Funktionen

- ✓ Immer wiederkehrende Abläufe werden nur einmal beschrieben und können danach beliebig oft ausgeführt werden.
- ✓ Der Programmcode wird mithilfe von Funktionen strukturiert, lässt sich leichter nachvollziehen, ist dadurch übersichtlicher und einfacher zu pflegen.
- ✓ Änderungen am Programm lassen sich schneller und einfacher durchführen, da eine Änderung nur an einer Stelle nötig ist (und zwar in der Funktion) und sich die Änderungen auch auf die Bereiche des Quellcodes auswirken, die diese Funktion verwenden.



Eine Funktion wird erst ausgeführt, wenn sie im Programm aufgerufen wird. Dies geschieht über den Namen der Funktion, gefolgt von einem runden Klammernpaar `()`. In den runden Klammern können je nach Definition ein oder mehrere Parameter stehen, die der Funktion übergeben werden und dort verarbeitet werden. Unabhängig davon, wo eine Funktion im PHP-Code steht, wird der Funktionsblock bei der zeilenweisen Abarbeitung eines PHP-Skripts übersprungen. Die Funktion wird ausschließlich durch expliziten Aufruf ausgeführt.

Das eigentliche PHP-Skript wird bis zum Aufruf einer Funktion von oben nach unten abgearbeitet ①. Der Funktionsaufruf ② erzwingt einen Sprung in die angegebene Funktion. Jetzt werden die Anweisungen der Funktion abgearbeitet ③. Ist das Ende der Funktion erreicht, wird zum Programm zurückgesprungen ④. Das Programm wird weiter zeilenweise ausgeführt ⑤.

An welcher Stelle im Programm die Funktion definiert wird, spielt keine Rolle. Daher ist es trotz der zeilenweisen Abarbeitung des Skripts möglich, eine Funktion aufzurufen, die weiter oben steht oder erst weiter unten definiert wird. Dennoch wird empfohlen, alle verwendeten Funktionen zu bündeln, entweder am Anfang oder am Ende eines Skripts, oder diese in separate Dateien auszulagern (dazu mehr am Ende dieses Kapitels). Dies ist übersichtlich und trennt die Definitionen vom Programmablauf.



Eine Funktion erstellen

Syntax und Beschreibung der `function`-Anweisung

- ✓ Das reservierte Schlüsselwort `function` leitet eine Funktion ein.
- ✓ `name` ist die Bezeichnung der Funktion, diese ist von Ihnen frei (nach bestimmten Regeln) wählbar und sollte einen Bezug dazu haben, welche Aufgabe die Funktion erfüllt; z. B. können Sie eine Funktion, die das Quadrat einer Zahl berechnet, `quadratzahl` oder `berechnung_quadrat` nennen.
- ✓ Für den Namen einer Funktion gelten folgende Regeln:
 - ✓ Das erste Zeichen muss entweder ein Buchstabe oder ein Unterstrich sein.
 - ✓ Der Name darf nur aus Buchstaben und Ziffern bestehen, aber keine Umlaute oder "ß" und keine Sonderzeichen außer dem Unterstrich "_" enthalten.
 - ✓ Der Name kann Groß- oder Kleinbuchstaben enthalten, wobei der PHP-Interpreter bei Funktionsnamen **nicht** zwischen Groß- und Kleinschreibung unterscheidet.
 - ✓ Besteht ein Funktionsname aus mehreren Begriffen, können die einzelnen Begriffe durch den Unterstrich `_` getrennt werden, z. B. `berechne_quadrat_zahl()`, oder die CamelCase-Schreibweise verwendet werden, z. B. `berechneQuadratZahl()`. Verwenden Sie zur Vermeidung von Fehlern eine einheitliche Schreibweise.
 - ✓ Der Name darf nicht identisch sein mit einem sogenannten reservierten Wort (z. B. Befehl aus PHP).
- ✓ In den runden Klammern werden die Bezeichnungen der einzelnen Parameter (auch Übergabewerte genannt) angegeben, für die beim Aufruf der Funktion Werte übergeben werden können.
- ✓ Eine Funktion kann keinen, einen oder mehrere Parameter besitzen.

```
function name([Parameter]) {
    Anweisungsblock;
}
```

Eine Funktion mit `return`-Anweisung erstellen

Syntax und Beschreibung der `return`-Anweisung

- ✓ Mit der `return`-Anweisung kann von der Funktion ein Wert in den normalen Programmablauf zur Weiterverarbeitung zurückgeliefert werden.
- ✓ Sobald die `return`-Anweisung ausgeführt wird, wird eine Funktion verlassen und es erfolgt eine Rückkehr an die aufrufende Stelle. Eventuell folgende Programmzeilen innerhalb der Funktion werden nicht mehr ausgeführt.
- ✓ Die `return`-Anweisung kann ohne Rückgabewert auch ausschließlich zum Verlassen einer Funktion verwendet werden.

```
function name([Parameter]) {
    Anweisungsblock;
    return [$wert];
}
```



Wenn Sie kein `return` definiert haben, geben Funktionen trotzdem einen Wert zurück, und zwar `NULL`. Dieser ist notwendig, damit die aufrufende Stelle weiß, dass die Abarbeitung der Funktion beendet ist.

Eine Funktion aufrufen

Syntax und Beschreibung eines Funktionsaufrufs

- ✓ Eine Funktion können Sie von jeder beliebigen Stelle im PHP-Code aufrufen.
- ✓ Eine Funktion wird mit ihrem Namen und folgendem runden Klammernpaar `()` aufgerufen.
- ✓ Die runden Klammern nach dem Funktionsnamen können Parameter zur Übergabe von Werten enthalten. Falls keine Werte übergeben werden, bleiben die Klammern leer.
- ✓ Ob und welche Parameter übergeben werden können, wird durch die Funktionsdefinition bestimmt.
- ✓ Sie können eine Funktion auch aus einer anderen Funktion heraus aufrufen.
- ✓ Eine Funktion kann auch sich selbst aufrufen (rekursiver Aufruf).

```
<?php
    Anweisungsblock
    funktionsname();
    Anweisungsblock
?>
```

8.2 Mit Funktionen arbeiten

Je nach Verwendungszweck einer Funktion kann eine Funktion mit oder ohne Parameter sowie mit oder ohne Rückgabewert erforderlich sein.

Funktionen ohne Parameter

Bei einer Funktion ohne Parameter werden bei jedem Aufruf dieselben Anweisungen ausgeführt.

Beispiel: `funktionen-1.php`

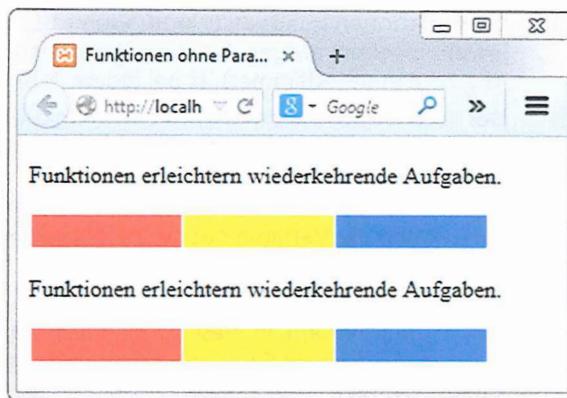
```
<?php
    // Funktionsdefinitionen
    ① function textausgabe() {
        echo "<p>Funktionen erleichtern wiederkehrende Aufgaben.</p>";
    }

    ② function farbtabelle() {
        echo "<table>
            <tr>
                <td width='100' style='background-color:#FF0000'>&nbsp;</td>
                <td width='100' style='background-color:#FFFF00'>&nbsp;</td>
                <td width='100' style='background-color:#0000FF'>&nbsp;</td>
            </tr>
        </table>";
    }

    // normaler Programmablauf
    ③ textausgabe();
    ④ farbtabelle();
    ③ textausgabe();
    ④ farbtabelle();

?>
```

- ① Die Funktion `textausgabe()` wird definiert, in der ein einfacher Text ausgegeben wird.
- ② Es erfolgt die Definition der Funktion `farbtabelle()`, die eine einfache Tabelle mit verschiedenfarbigem Hintergrund für jede Zelle ausgibt.
- ③ Die Funktion `textausgabe()` wird aufgerufen.
- ④ Die Funktion `farbtabelle()` wird ausgeführt.



Anzeige der Beispieldatei "funktionen-1.php"

Funktionen mit einem oder mehreren Parametern

Bei einer Funktion mit einem oder mehreren Parametern werden bei jedem Aufruf der Funktion mit den Parametern gleiche Aufgaben ausgeführt. Hierbei kann ein Parameter eine Zeichenkette, ein Zahlenwert, eine Konstante oder eine Variable sein, aber auch Arrays oder Objekte können Funktionen übergeben werden.

Bei Funktionen mit mehreren Parametern werden diese in der **Reihenfolge** des Aufrufs übergeben. Der erste Parameter im Funktionsaufruf wird dem ersten Parameter in der Funktionsdeklaration übergeben, der zweite dem zweiten usw. Standardmäßig muss die **Anzahl** der Parameter im Funktionsaufruf der in der Funktionsdeklaration entsprechen. Lediglich beim Einsatz von optionalen Parametern (Vorgabewerten, weiter unten) können diese im Funktionsaufruf weggelassen werden.



Beispiel: funktionen-2.php

Im folgenden Skript werden mathematische Berechnungen mithilfe von Funktionen mit jeweils einem Parameter durchgeführt.

```
<?php

③ function quadrat($zahl) {
    $ergebnis = $zahl * $zahl;
    echo "<p>Die Quadratzahl von $zahl ist: <strong>$ergebnis</strong>.</p>";
}

④ function wurzel($zahl) {
    $ergebnis = sqrt($zahl);
    echo "<p>Die Wurzel aus $zahl ist: <strong>$ergebnis</strong>.</p>";
}

① quadrat(9);
② wurzel(64);
① quadrat(5);
② wurzel(49);
?>
```

- ①/② Die Funktionen `quadrat()` und `wurzel()` werden jeweils zweimal aufgerufen. Der übergebene Parameter (die Zahl in der Klammer) ist bei jedem Aufruf ein anderer.
- ③ Der Wert in der Klammer wird beim Funktionsaufruf der Variablen `$zahl` zugewiesen und gespeichert. Diese Zuweisung geschieht bei jedem Funktionsaufruf erneut. Mithilfe der Variablen `$zahl` wird beim Aufruf beider Funktionen die Variable `$ergebnis` berechnet und am Bildschirm ausgegeben.
- ④ Aus einer Funktion heraus können andere Funktionen aufgerufen werden, in diesem Fall die vordefinierte mathematische PHP-Funktion `sqrt()` zur Berechnung der Quadratwurzel.

```
Die Quadratzahl von 9 ist: 81.  
Die Wurzel aus 64 ist: 8.  
Die Quadratzahl von 5 ist: 25.  
Die Wurzel aus 49 ist: 7.
```

Anzeige der Datei "funktionen-2.php"

Beispiel: `funktionen-3.php`

Wird eine Funktion mit mehreren Parametern aufgerufen, wird der erste Wert an den ersten Parameter, der zweite Wert an den zweiten Parameter übergeben usw.

```
<?php  
①     function honorar_berechnen($dozent, $stundenzahl, $honorarsatz) {  
    $honorar = $stundenzahl * $honorarsatz;  
    echo "<p>$dozent war $stundenzahl Stunden im Einsatz und hat $honorar Euro  
    verdient.</p>";  
}  
  
②     honorar_berechnen("Peter Schmidt", 15, 18);  
②     honorar_berechnen("Arndt Hoffmann", 38, 15.5);  
②     honorar_berechnen("Petra Meyer", 27, 23);  
?>
```

- ① Die Funktion `honorar_berechnen()` wurde mit 3 Parametern versehen: `$dozent`, `$stundenzahl`, `$honorarsatz`.
- ② Beim Aufruf der Funktion werden durch Komma getrennt die drei geforderten Parameter übergeben.

```
Peter Schmidt war 15 Stunden im Einsatz und hat 270 Euro verdient.  
Arndt Hoffmann war 38 Stunden im Einsatz und hat 589 Euro verdient.  
Petra Meyer war 27 Stunden im Einsatz und hat 621 Euro verdient.
```

Anzeige der Beispieldatei "funktionen-3.php"

 Wenn Sie an eine Funktion weniger Parameter übergeben als in der Funktionsdefinition vereinbart, erhalten Sie eine entsprechende Fehlermeldung (Ausnahme: Funktionen mit optionalen Parametern). Übergeben Sie einer Funktion mehr Parameter, als im Funktionskopf definiert sind, kommt es zu keiner Fehlermeldung, die überschüssigen Werte sind dann keinem Parameter im Funktionskopf zugewiesen. Allerdings können die überzähligen Werte über die PHP-Funktion `func_get_args()` ermittelt werden.

Optionale Parameter

Die Parameter im Kopf der Funktion, die dort mit Variablennamen angegeben sind, müssen der Funktion auch in dieser Anzahl übergeben werden. Fehlt ein Wert im Funktionsaufruf, gibt PHP eine Meldung der Kategorie `warning` aus.

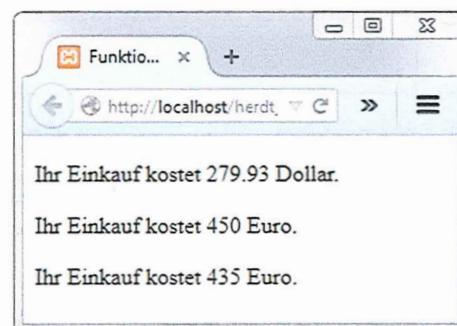
Sie können eine Variable in der Funktionsdeklaration jedoch mit einem Standardwert versehen. Damit deklarieren Sie diesen Parameter zu einem **optionalen Parameter**. Der Standardwert wird auch als **Vorgabewert** bezeichnet. PHP arbeitet dann wie folgt: Wird im Funktionsaufruf ein Wert für den optionalen Parameter übergeben, verwendet die Funktion den Übergabewert für diese Variable, der Standardwert wird dann ignoriert. Wird hingegen kein Wert für diesen Parameter übergeben, verwendet die Funktion den Standardwert. Damit kommt es zu keiner Fehlermeldung, wenn ein optionaler Parameter nicht übergeben wird.

Zu beachten ist dabei die Reihenfolge von nicht-optionalen und optionalen Parametern. Parameter, die auf jeden Fall einen Wert erwarten, sollten in der Funktionsdeklaration zuerst angegeben werden, erst dahinter die optionalen Parameter.

Beispiel: *optional.php*

①	<?php function berechne(\$anzahl, \$preis = 45, \$waehrung = "Euro") { echo "<p>Ihr Einkauf kostet " . (\$anzahl * \$preis) . " \$waehrung.</p>"; }
②	berechne(7, 39.99, "Dollar"); // normaler Aufruf der Funktion
③	berechne(10); // Parameter 2 und 3: Standardwerte werden verwendet
④	berechne(15, 29); // Parameter 3: Standardwert wird verwendet

- ① Die Funktion `berechne()` wurde mit 3 Parametern versehen: `$anzahl`, `$preis`, `$waehrung`. Die letzten beiden Parameter sind mit Standardwerten versehen worden, es handelt sich also um optionale Parameter, die dann verwendet werden, wenn im Funktionsaufruf der zweite und/oder dritte Parameter nicht angegeben wird.
- ② Die Funktion wird mit allen geforderten Parametern aufgerufen.
- ③ Die Funktion wird einmal nur mit einem Parameter, einmal mit zwei Parametern aufgerufen. PHP interpretiert die angegebenen Parameter automatisch als **erste** Parameter der abzuarbeitenden Argumentliste. Da alle weiteren Parameter über Standardwerte verfügen, kann die Funktion in beiden Fällen ohne Fehlermeldung ausgeführt werden.



Anzeige der Beispieldatei "optional.php"

Wertübergabe per Wert oder per Referenz

Wenn Sie eine Variable als Parameter an eine Funktion übergeben, dann erhält die Funktion eine **Kopie** der übergebenen Variablen. Die Übergabe der Parameter als Kopie wird auch als **call-by-value** bezeichnet. Wenn der Wert einer übergebenen Variablen innerhalb der Funktion geändert wird, wird nur die lokale Kopie geändert. Eine Veränderung der Kopie hat keinerlei Rückwirkung auf den Parameter, der vom Haupt-PHP-Skript übergeben wurde.



Manchmal ist jedoch eine Veränderung der übergebenen Variablen im übergebenden Programm erwünscht, beispielsweise um den Preis einer Bestellung abhängig zu machen von der vorgegebenen Menge, um dann im PHP-Skript selbst mit dem veränderten Wert weiterzuarbeiten. Um dies zu erreichen, übergeben Sie den Parameter als **Referenz** auf das Original. Sie übergeben also nicht, wie bei dem call-by-value-Verfahren, den Inhalt bzw. den Wert einer Variablen, sondern einen sogenannten **Verweis** auf die Variable. Diese Vorgehensweise wird **call-by-reference** genannt. Eine Veränderung hat, anders als bei der Übergabe als Kopie, Rückwirkung auf das Original, da nicht die Kopie, sondern die Variable selbst verarbeitet wird. Eine Referenz auf das Original erhalten Sie, indem Sie in der Funktionsdefinition vor den Variablennamen ein `&` setzen.

Die **Parameterübergabe per Referenz** kann **nur** in der **Funktionsdeklaration** festgelegt werden. Bis zur PHP-Version 5.3 konnte das Zeichen `&` auch im Funktionsaufruf notiert werden, um einen Wert per Referenz zu übergeben. Ab PHP 5.4 führt der Einsatz des Zeichens `&` im Funktionsaufruf zum **fatal error**.



Beispiel: by_reference.php

```

① <?php
    function quadrat($zahl) {
        echo "<p>Die Quadratzahl von $zahl ist: ";
        $zahl = $zahl * $zahl;
        echo $zahl . "</p>";
    }

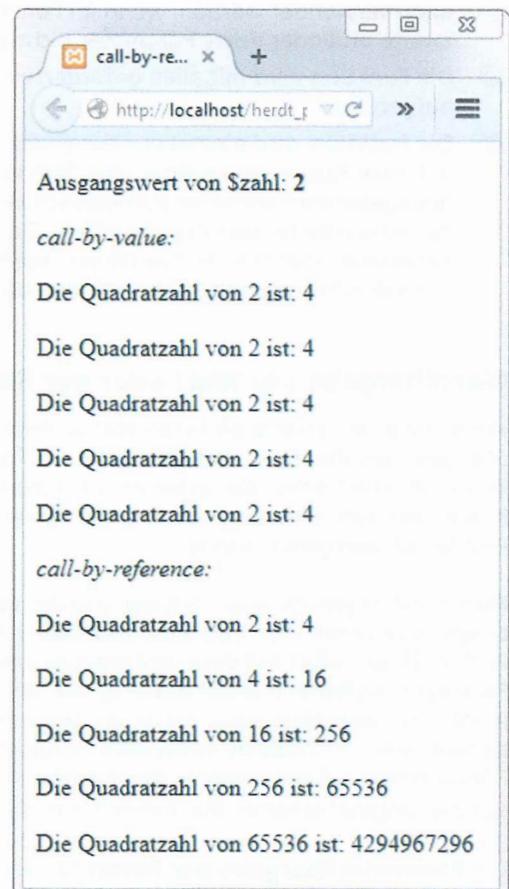
② function quadrat_referenz(&$zahl) {
    echo "<p>Die Quadratzahl von $zahl ist: ";
    $zahl = $zahl * $zahl;
    echo $zahl . "</p>";
}

$zahl = 2;
echo '<p>Ausgangswert von $zahl: <strong>' . $zahl . '</strong></p>';
echo "<em>call-by-value:</em>";
for ($i = 1; $i <= 5; $i++) { // Schleife: 5-malige Ausführung
    quadrat($zahl); // call-by-value
}
echo "<p><em>call-by-reference:</em></p>";
for ($i = 1; $i <= 5; $i++) {
    quadrat_referenz($zahl); // call-by-reference
}

?>

```

- ① Die Funktion `quadrat()` wird definiert. Sie berechnet auf Basis des Übergebewertes `$zahl` die Quadratzahl und gibt sie aus. Diese Funktion arbeitet ohne Parameterübergabe per Referenz, also per `call-by-value`.
- ② Die Funktion `quadrat_referenz()` berechnet die Quadratzahl von `$zahl` auf exakt die gleiche Weise, arbeitet aber mit `call-by-reference`, d. h., Parameter werden als Referenz weiterverarbeitet. Dies erkennen Sie am Zeichen `&` vor der Variablen `$zahl` in der Funktionsdefinition.
- ③ Die Funktion `quadrat()` wird in einer Schleife fünfmal aufgerufen und jeweils die Quadratzahl von 2 berechnet (`call-by-value`). Alle fünf Funktionsaufrufe ergeben das gleiche Ergebnis 4. Der Wert der Variablen `$zahl` bleibt 2. Innerhalb der Funktion wird nur mit einer Kopie der Variablen gearbeitet, die das Original nicht verändert.
- ④ Wird allerdings die Funktion `quadrat_referenz()` mit einem Parameter aufgerufen, handelt es sich um einen `call-by-reference`-Aufruf. In diesem Fall wird auch die Variable `$zahl` durch die Berechnung verändert. Da die Funktion auch hier fünfmal aufgerufen wird, sehen Sie die Auswirkungen im Ergebnis des Funktionsaufrufs (4, 16, 256, 65536 und 4294967296).



Anzeige der Beispieldatei
"by_reference.php"

Variadische Funktionen (variadic functions) mit . . . verwenden

Variadisch beschreibt die Eigenschaft von Funktionen, dass die Anzahl der Parameter in der Funktionsdeklaration (Funktionskopf) nicht zwingend festgelegt werden müssen. Selbst wenn Sie eine Funktion mit drei Parametern deklariert haben, können Sie fünf Parameter übergeben, ohne dass dies zum PHP-Fehler führt. Über die Funktion `func_get_args()` können nicht deklarierte Variablen ermittelt werden.

Mit PHP 5.6 wird dieses Prinzip erweitert. Über den Einsatz des Zeichens für die Auslassungspunkte . . . (drei Punkte, auch Ellipse genannt, in anderen Programmiersprachen Splat-Operator genannt) können Funktionen jetzt mit einer beliebigen Anzahl von Parametern aufgerufen werden. Ein Parameter im Funktionskopf, dem die Auslassungszeichen vorangestellt sind, z. B. `function(...$args)`, nimmt alle Übergabeparameter in einem Array auf. Aber auch beim Aufruf einer Funktion können die Auslassungspunkte verwendet werden, z. B. `meinFunktionsAufruf(...$param)`. Hier kann die Funktion mit einem Array aufgerufen werden, im Funktionskopf werden dann die einzelnen Arrayeinträge auf die dort deklarierten Parameter verteilt.

Zugriff auf variable Parameter

Beispiel: variadic-1.php (Auszug)

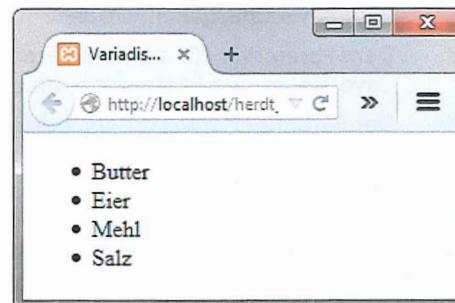
```

① <?php
    // Funktion mit Parameter für variable Anzahl von Parametern
    function zeigeZutaten(...$args) {
        echo "<ul>";
        foreach ($args as $val) {
            echo "<li>$val</li>";
        }
        echo "</ul>";
    }

③     zeigeZutaten("Butter", "Eier", "Mehl", "Salz");
?>

```

- ① Die Funktion `zeigeZutaten()` wird aufgebaut. Als Parameter wird die Variable `$args` deklariert. Dieser Variable wird das Auslassungszeichen . . . vorangestellt. Damit werden übergebene Parameter als Array aufgenommen.
- ② Mit einer `foreach`-Schleife wird das Array durchlaufen und jeder einzelne Arrayeintrag als Listeneintrag ausgegeben.
- ③ Im Funktionsaufruf `zeigeZutaten()` werden hier vier Parameter übergeben. Hier könnten beliebig viele Parameter übergeben werden, alle werden von der Funktionsvariable `$args` als Array aufgenommen.



In der Beispieldatei `variadic-1.php` finden Sie eine zweite Funktion: `zeigeZutatenAlt()`. Dort wird die Alternative für ältere PHP-Versionen gezeigt: Dort ist kein Parameter im Funktionskopf deklariert. Der Funktionsaufruf findet ebenfalls mit vier Parametern statt, diese werden in der Funktion über `func_get_args()` ermittelt und im Array `$args` gespeichert und ebenfalls über eine `foreach`-Schleife ausgelesen und die Einträge angezeigt.

Beim Zugriff auf variable Parameter und der Verwendung des Auslassungszeichens . . . ist Folgendes zu beachten:

- ✓ Das Auslassungszeichen . . . wird einer Variablen vorangestellt.
- ✓ Eine Funktion mit einer Variablen mit dem Auslassungszeichen . . . kann weitere Parameter in der Funktionsdeklaration haben. Diese müssen zuerst deklariert werden.
- ✓ Die Variable mit dem Auslassungszeichen . . . muss als **letzter** Parameter im Funktionskopf deklariert werden. Dieser wird von den anderen Parametern mit einem Komma getrennt. Folgen nach der Variablen für variable Parameter weitere Variablen, meldet PHP einen Fehler vom Typ *fatal error*.
- ✓ Die Variable für variable Parameter kann **nicht** mit einem optionalen Parameter deklariert werden.
- ✓ Diese Variable kann nur per *call-by-value* übergeben werden. Die Übergabe per *call-by-reference* führt ebenfalls zum *fatal error*.

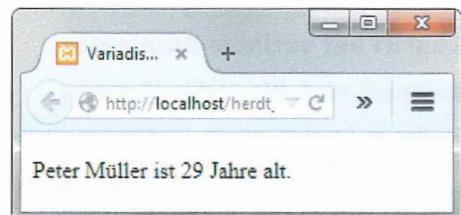
Übergabe einer Parameterliste

Beispiel: *variadic-2.php* (Auszug)

```
<?php
    // Funktion mit Parameter für variable Anzahl von Parametern
    function personInfo($name, $vorname, $alter) {
        echo "<p>$vorname $name ist $alter Jahre alt.</p>";
    }

    $person = ['Müller', 'Peter', 29];
    personInfo(...$person);
?>
```

- ① Die Funktion `personInfo()` wird aufgebaut. Die Parameterdeklaration im Funktionskopf entspricht dem Standardaufbau von Funktionen. Es werden drei Variablen deklariert, die jeweils einen Parameter erwarten.
- ② Die übergebenen Parameter werden als Zeichenkette auf dem Bildschirm ausgegeben.
- ③ Hier wird ein Array über die Kurzschreibweise definiert. Dieses enthält drei Einträge.
- ④ Die Funktion `personInfo()` wird mit dem zuvor erstellten Array aufgerufen. Wichtig beim Funktionsaufruf ist das der Arrayvariablen vorangestellte Auslassungszeichen `...`. Daran erkennt PHP, dass dieser Parameter nicht an einen einzelnen Parameter im Funktionskopf übergeben werden soll, sondern dass es sich um eine Parameterliste als Array handelt, die auf die einzelnen Parameter im Funktionskopf verteilt werden soll.



Bei der Verwendung des Auslassungszeichens `...` zur Übergabe einer Parameterliste ist Folgendes zu beachten:

- ✓ Werden mehrere Parameter an die Funktion übergeben, darf die Übergabeliste **nur** als letzter Parameter übergeben werden.
- ✓ Dem Parameter für die Parameter wird das Auslassungszeichens `...` voran gestellt und von etwaigen davorstehenden Parametern mit einem Komma getrennt.
- ✓ Der Parameter erwartet ein indiziertes Array oder ein traversables Objekt (wird in diesem Buch nicht weiter vertieft). Eine Zeichenkette oder ein assoziatives Array führen zu einem *fatal error*.



Bevor Sie ein PHP-Skript schreiben, dass später bei einem professionellen Provider laufen soll, stellen Sie sicher, dass dieser PHP 5.6 unterstützt, wenn Sie das Auslassungszeichen `...` für den variablen Zugriff auf Parameter verwenden möchten. Das erspart Ihnen die Fehlersuche und späteres Anpassen von fertigen Skripten, wenn der Provider noch nicht diese PHP-Version anbietet.

Funktionen mit Rückgabewerten

Wenn Sie innerhalb einer Funktion **ein** Ergebnis ermitteln und dies zurückliefern möchten, verwenden Sie die `return`-Anweisung.



Allerdings ist hier zu beachten, dass eine `return`-Anweisung nicht nur den Rückgabewert festlegt, sondern auch die Ausführung der Funktion sofort beendet. Alle Anweisungen nach einer `return`-Anweisung werden nicht mehr ausgeführt und der vorhandene Rückgabewert wird an das Programm übergeben. `return` kann auch ohne einen konkreten Rückgabewert verwendet werden. Auch dies beendet sofort die Abarbeitung der Funktion, dahintergelegene Anweisungen innerhalb der Funktion werden nicht mehr ausgeführt.

Beispiel: funktionen-4.php

```

① <?php
②     function addiere($zahl1, $zahl2) {
③         $ergebnis = $zahl1 + $zahl2;
④         return $ergebnis;
⑤     }
⑥
⑦     addiere(4, 3); // nur Ausführung der Funktion, keine Ausgabe
⑧         //auf dem Bildschirm
⑨     echo "<hr>";
⑩     $summe = addiere(30, 19); // Rückgabewert zur Weiterverarbeitung
⑪         // in einer Variablen gespeichert
⑫     echo $summe;
⑬
⑭     echo "<hr>";
⑮     echo addiere(17, 4); // direkte Ausgabe des Rückgabewertes
⑯ ?>

```

- ① Die Funktion `addiere()` hat die beiden Parameter `$zahl1` und `$zahl2`.
- ② Die beiden Parameter werden innerhalb der Funktion addiert und das Ergebnis in der Variablen `$ergebnis` gespeichert.
- ③ Der Wert der Variablen `$ergebnis` wird mithilfe der Anweisung `return` an die aufrufende Stelle zurückgegeben.
- ④ Die Funktion wird mit den Werten 4 und 3 aufgerufen. Die Funktion wird ausgeführt, das Ergebnis 7 an das Programm zurückgegeben, vom aufrufenden Programm aber nicht verwendet, also weder in einer Variablen gespeichert noch per `echo` ausgegeben.
- ⑤ Das Ergebnis eines weiteren Funktionsaufrufs 49 wird der Variablen `$summe` zugewiesen und in der folgenden Programmzeile ausgegeben. Das Speichern des Rückgabewertes einer Funktion in einer Variablen wird häufig verwendet, wenn mit diesem Wert weitere Operationen durchgeführt werden sollen.
- ⑥ Der Funktionsaufruf gibt das Ergebnis 21 zurück. Das Ergebnis wird durch die vorangestellte `echo`-Anweisung sofort ausgegeben. Diese Methode wird oft verwendet, wenn eine Ausgabe, aber keine Weiterverarbeitung des Wertes vorgenommen werden soll.



Anzeige der Datei "funktionen-4.php"

Mehrere Rückgabewerte

PHP kann nur **einen** Wert mit `return` zurückgeben. Wenn Sie mehrere Werte aus einer Funktion zurückgeben möchten, können Sie diese Werte in einem Array zusammenfassen. Um die Elemente des zurückgelieferten Arrays dann Variablen zuzuweisen, verwenden Sie die `list()`-Funktion.

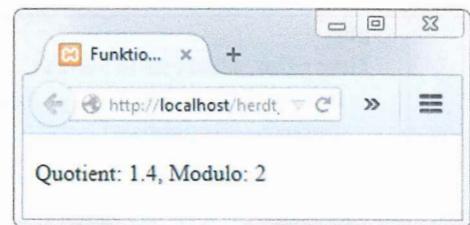
Beispiel: multi_return.php

```

① <?php
②     function berechnung($zahl1, $zahl2) {
③         $quotient = $zahl1 / $zahl2;
④         $modulo = $zahl1 % $zahl2;
⑤         $werte = array($quotient, $modulo);
⑥         return $werte;
⑦     }
⑧
⑨     list($wert1, $wert2) = berechnung(7, 5);
⑩     echo "<p>Quotient: " . $wert1 . ", Modulo: " . $wert2."</p>";
⑪ ?>

```

- ① Die Funktion `berechnung()` hat die beiden Parameter `$zahl1` und `$zahl2`. In der Funktion wird der Quotient zweier Zahlen berechnet und zusätzlich der Restwert einer Ganzzahlendivision dieser Zahlen.
- ② Die beiden Ergebnisse der Berechnungen werden als Elemente eines Arrays an die Variable `$werte` übergeben.
- ③ Mithilfe der `return`-Anweisung wird die Array-Variable `$werte` aus der Funktion an die aufrufende Stelle zurückgegeben.
- ④ Mit der Funktion `list()` wird der Rückgabewert, also die Array-Variable, neuen Variablen zugewiesen.
- ⑤ Die Variablen werden ausgegeben.



Anzeige der Datei "multi_return.php"

8.3 Der Gültigkeitsbereich von Variablen

Variablen, die Sie außerhalb einer Funktion im PHP-Programm einsetzen, sind standardmäßig innerhalb von Funktionen nicht definiert, also dort unbekannt. Das Gleiche gilt für Variablen innerhalb von Funktionen, auf die Sie nicht im normalen Programmablauf zugreifen können.

Der Gültigkeitsbereich von Variablen (auch Namensraum genannt) ergibt sich also automatisch aus dem Zusammenhang, in dem sie definiert wurden.

- ✓ **Lokale Variablen** sind nur innerhalb der Funktion gültig, in der sie definiert wurden. Wurde die Variable im normalen Programmablauf definiert (außerhalb einer Funktion), steht sie auch nicht innerhalb von Funktionen zur Verfügung. Dieses Verhalten ist der Standard.
- ✓ Lokale Variablen können zu **globalen Variablen** werden, indem Sie innerhalb einer Funktion mit dem Schlüsselwort `global` die Variablen bekannt machen, z. B. `global $zahl`. Damit heben Sie für diese Variable die Grenze zwischen Funktion und Programmablauf auf. Eine als `global` definierte Variable steht sowohl in der Funktion als auch außerhalb der Funktion zur Verfügung. Das bedeutet zugleich, dass Veränderungen an der Variablen, sowohl im Haupt-Skript als auch innerhalb einer Funktion, die Variable immer beeinflussen.
- ✓ **Superglobale Variablen** sind von PHP vordefinierte Variablen, die Ihnen an jeder Stelle des Programms zur Verfügung stehen. Sie erkennen diese Variablen daran, dass sie mit den Zeichen `$_` beginnen und vollständig aus Großbuchstaben bestehen, z. B. `$_POST`. Superglobale Variablen können Sie nicht selbst definieren. Auch wenn Sie sich an die beschriebene Konvention zur Benennung dieser Variablen halten, erstellen Sie damit keine superglobalen Variablen.

Im folgenden Beispiel werden die verschiedenen Typen von Variablen gegenübergestellt. Hierzu wird ein Bestellformular für Äpfel erstellt. Anhand des PHP-Codes im Auswertungsprogramm wird gezeigt, wann welche Variablen gültig sind.

Beispiel: `formular_funktion.html`

	<h1>Apfelkauf</h1>
	<p>Bitte geben Sie die gewünschte Menge ein und wählen Sie eine Apfelsorte:</p>
①	<form action="funktion_var.php" method="post">
②	<p>Menge (in kg): <input type="text" name="menge"></p>
③	<p>Apfelsorte: <input type="radio" name="sorte" value="Jonagold"> Jonagold <input type="radio" name="sorte" value="Gala"> Gala <input type="radio" name="sorte" value="Elstar"> Elstar</p> <p><input type="submit" value="Abschicken"></p>
	</form>

- ① Mithilfe der Methode POST werden nach Absenden des Formulars folgende Variablen an das Auswertungsprogramm *funktion_var.php* übergeben:
- ② die Variable \$menge, die vom Benutzer eingetragen wird, und
- ③ die Variable \$sorte, die der Benutzer mithilfe eines Optionsfeldes auswählt.

Lokale, gl... +

http://localhost/herdt_p

Apfelkauf

Bitte geben Sie die gewünschte Menge ein und wählen Sie eine Apfelsorte:

Menge (in kg):

Apfelsorte: Jonagold Gala Elstar

Abschicken

Anzeige Datei "formular_funktion.html"

Beispiel: *funktion_var.php*

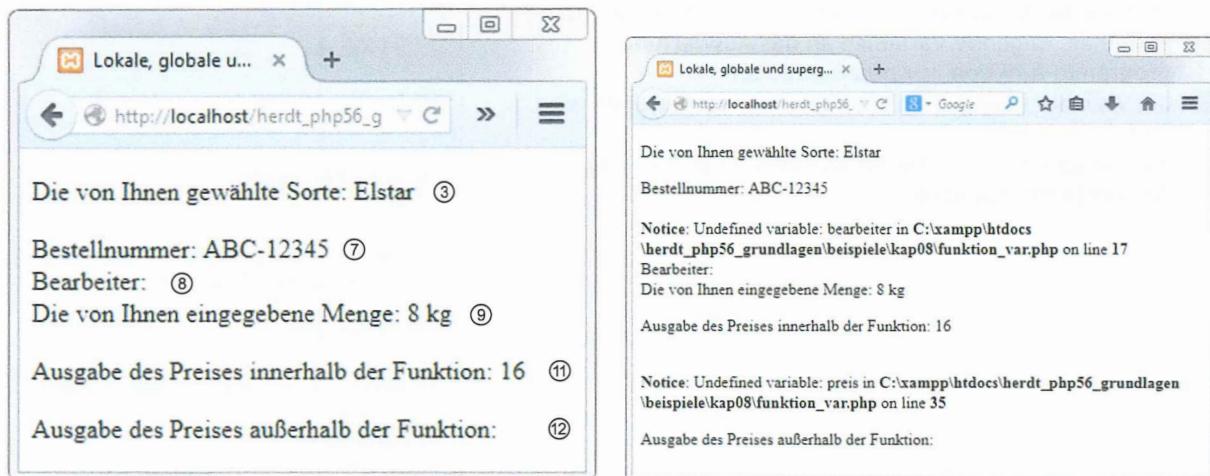
```

① <?php
    error_reporting(E_ERROR | E_PARSE); // Nur einfache Fehler melden

② $bestellnummer = "ABC-12345";
    $bearbeiter = "Maria Schulze";

⑤ function bestellung() {
    global $bestellnummer; // globale Variable
    echo "<p>Bestellnummer: " . $bestellnummer . "<br>";
    echo "Bearbeiter: " . $bearbeiter . "<br>";
        // lokale Variable - funktioniert hier nicht!
    echo "Die von Ihnen eingegebene Menge: " . $_POST["menge"] . " kg</p>";
    switch ($_POST["sorte"]) {
        case "Jonagold":
            $preis = $_POST["menge"] * 1.50;
            break;
        case "Gala":
            $preis = $_POST["menge"] * 1.65;
            break;
        case "Elstar":
            $preis = $_POST["menge"] * 2.00;
            break;
    }
    echo "<p>Ausgabe des Preises innerhalb der Funktion: $preis</p>";
}

③ echo "<p>Die von Ihnen gewählte Sorte: " . $_POST["sorte"] . "</p>";
bestellung();
echo "<p>Ausgabe des Preises außerhalb der Funktion: $preis</p>";
        // lokale Variable der Funktion - funktioniert hier nicht!
?>
```



Anzeige des Beispielprogramms "funktion_var.php"

Links ist das Error-Reporting für *notice* deaktiviert. Rechts ist das Standard-Error-Reporting von PHP belassen, die *notice*-Meldungen weisen bereits darauf hin, an welchen Stellen die Variablen unbekannt sind.

- ① *notice*-Meldungen werden standardmäßig ausgegeben. (Bis PHP 5.3 gehörte *notice* nicht zu den Standardmeldungen.) In der rechten Abbildung wird durch die Fehlermeldungen bereits deutlich, welche Variablen wo nicht bekannt sind. Um die Anzeige in der linken Abbildung zu erhalten, sind Fehlermeldungen per `error_reporting(E_ERROR | E_PARSE)` auf schwerwiegende Fehler reduziert.
- ② Die Variablen `$bestellnummer` und `$bearbeiter` werden mit einem Wert belegt.
- ③ Der Wert des **superglobalen Arrays** `$_POST["sorte"]` wird außerhalb der Funktion `bestellung()` ausgegeben.
- ④ Die Funktion `bestellung()` wird aufgerufen. Da Funktionsblöcke erst durchlaufen werden, wenn die Funktion aus dem Skript aufgerufen wird, werden die Anweisungen innerhalb der Funktion erst ausgeführt, wenn `bestellung()` ⑤ aufgerufen wurde.
- ⑥ Die Variable `$bestellnummer` wird innerhalb der Funktion `bestellung()` ⑤ mithilfe des Schlüsselwortes `global` bekannt gemacht.
- ⑦ + ⑧ Die Variablen `$bestellnummer` und `$bearbeiter` sollen innerhalb der Funktion ausgegeben werden. Da die Variable `$bearbeiter` innerhalb der Funktion **nicht** mithilfe des Schlüsselwortes `global` bekannt gemacht wurde, wird sie bei der Ausgabe nicht angezeigt ⑧. Sie gilt als nicht definiert.
- ⑨ Der Wert des **superglobalen Arrays** `$_POST["menge"]` wird innerhalb der Funktion `bestellung()` ausgegeben.
- ⑩ Innerhalb der Funktion `bestellung()` wird der Wert der **lokalen Variablen** `$preis` per switch-Anweisung in Abhängigkeit von der Apfelsorte berechnet.
- ⑪ Die **lokale Variable** `$preis` wird innerhalb der Funktion ausgegeben.
- ⑫ Die **lokale Variable** `$preis` ⑪ soll auch außerhalb der Funktion ausgegeben werden. Dies funktioniert allerdings nicht, da es sich um eine lokale Variable handelt, die innerhalb dieser Funktion definiert wurde, aber nicht mit dem Schlüsselwort `global` bekannt gemacht wurde.

8.4 PHP-Dateien einbinden mit `include()` und `require()`

Bislang wurde in allen Übungen mit einer einzelnen PHP-Datei gearbeitet. Wie Sie gelesen haben, optimieren Funktionen den Quelltext dadurch, dass wiederkehrende Aufgaben in Funktionen gekapselt werden und von beliebigen Stellen im Skript aufgerufen werden können. Ein weiterer Schritt, PHP-Code zu verbessern, ist, Funktionen in separate Dateien auszulagern, damit diese nicht nur in einer Datei, sondern von beliebig vielen Dateien verwendet werden können.

Um ausgelagerte Funktionen verwenden zu können, muss die Datei mit den darin enthaltenen Funktionen mit der eigentlichen Skript-Datei verknüpft werden. Mithilfe der `include()`- bzw. `require()`-Anweisung können Sie ausgelagerte PHP-Abschnitte, wie z. B. eigene PHP-Funktionen in Ihr PHP-Programm einbinden.

Mit include() und require() arbeiten

Die `include()`- bzw. `require()`-Anweisung bindet eine Datei, deren Name (und relative oder absolute Pfadangabe) als Argument übergeben wird, in den aktuellen Programmcode ein.

Unterschiede zwischen include() und require()

Beide Anweisungen funktionieren ähnlich. Ein wichtiger Unterschied zwischen den beiden Anweisungen zeigt sich, wenn eine Datei fehlerhaft eingebunden wurde:

- ✓ Wenn beispielsweise die angegebene Datei nicht vorhanden ist, erzeugt `include()` eine Warnung. Das Skript wird weiter ausgeführt. Befindet sich in der Datei kein PHP-Code, der für die weitere Abarbeitung des Haupt-Skripts notwendig ist, kann diese auch mit einem fehlerhaften `include()` bis zum Ende laufen.
- ✓ Der Befehl `require()` beendet bei einer fehlenden Datei das Skript sofort mit einer Fehlermeldung.

Syntax und Bedeutung der include() - und require() -Anweisungen

- ✓ Die `include()`- und `require()`-Befehle erwarten als Parameter die einzubindende Datei. Befindet sich die Datei nicht im selben Ordner, muss der Pfad angegeben werden.
- ✓ Der Aufruf erfordert nicht zwingend die runden Klammern. Dateiname und Pfad können auch in Anführungszeichen hinter dem Schlüsselwort `include` bzw. `require` angegeben werden.
- ✓ Beinhaltet die eingebundene Datei keinen PHP-Code, wird der Inhalt unverändert an den Browser weitergegeben und somit am Bildschirm dargestellt.
- ✓ Auch Ausgaben per `echo` oder `print_r` in einer eingebundenen Datei werden, soweit keine anderen Maßnahmen ergriffen wurden, direkt im Browser ausgegeben.
- ✓ Eingebundene Dateien, die per `return` einen Rückgabewert liefern, können wie Funktionen mit Rückgabewert behandelt werden. Entweder wird der Rückgabewert direkt per `echo` `include("Dateiname")` ausgegeben oder er wird in einer Variablen gespeichert: `$variable = include("Dateiname")` und kann dann per `echo $variable` ausgegeben werden.
- ✓ Jede Funktionsdefinition in eingebundenen Dateien steht in der aufrufenden Datei zur Verfügung. Die Vorstellung, die eingebundene Datei wird in die aufrufende Datei an die Stelle des Aufrufs "hineinkopiert", erleichtert das Verständnis der Funktionsweise.

```
include("Dateiname");
require("Dateiname");

include ("Pfad/Dateiname");
require ("Pfad/Dateiname");

include "Dateiname";
require "Dateiname";
```

Wenn `include()` oder `require()` ausgeführt werden, wird die Abarbeitung des Haupt-Skripts so lange unterbrochen, bis die fremde Datei geladen und verarbeitet wurde. Aus diesem Grund muss der PHP-Code der eingebundenen Dateien vollständig von gültigen PHP-Tags `<?php ... ?>` umschlossen sein.



Skripte, die in einen PHP-Code eingebunden werden sollen, können mit der Dateierweiterung `.inc` abgespeichert werden. Je nach Serverkonfiguration wird bei `.inc`-Dateien der PHP-Code jedoch nicht geparsst, sondern der Inhalt direkt im Browser ausgegeben, wenn die Datei direkt im Browser aufgerufen wird, was je nach Inhalt eine Sicherheitslücke darstellen kann. Erst wenn eine `.inc`-Datei in PHP eingebunden ist, wird sie regulär wie PHP geparsst.



Alternativ bietet sich die Dateiendung `.inc.php` an. Daran können Sie erkennen, dass eine solche Datei zur Einbindung vorgesehen ist, sie wird aber beim direkten Aufruf im Browser geparsst. Eine ungewollte Ausgabe des PHP-Codes im Browser wird so vermieden.

Nur einmal laden: include_once() und require_once()

Die Funktionen `include_once()` und `require_once()` entsprechen im Wesentlichen den Funktionen `include()` und `require()`. PHP prüft allerdings, ob die angegebene Datei bereits eingebunden wurde. Ist das korrekt, wird die Datei nicht nochmals eingebunden.

Durch den Einsatz dieser Funktionen können Sie Fehler vermeiden, die z. B. durch Mehrfachdeklaration namengleicher Funktionen oder nochmalige Wertzuweisungen von Variablen hervorgerufen werden.

Datei mit require() einbinden

Beispiel: *require.php*

Zur Veranschaulichung der Arbeitsweise der `require()`-Anweisung wird ein häufiges Verfahren erläutert. Sie erstellen eine selbst definierte einfache Funktion `summe()` in einer separaten Datei. Außer der Funktion enthält diese Datei keinen anderen PHP-Code. Die Datei wird dann von einer zweiten Datei über die `require()`-Anweisung eingebunden, sodass auch dort die Funktion `summe()` zur Verfügung steht.

```
<h1>require: Verwenden einer Funktion aus einer anderen Datei</h1>
<?php
echo "<p>Ein fremdes PHP-Skript wird eingebunden und die dort enthaltene
Funktion ausgeführt:</p>";
require ("eigene_funktion.inc.php");
summe(2000, 1376);
echo "<hr><p>Hier kann das Skript weitere Anweisungen enthalten.</p>";
?>
```

- ① Die `require()`-Anweisung bindet die Datei *eigene_funktion.inc.php* ein.



Damit die Funktion `summe()` in der ausgelagerten Datei zur Verfügung steht, muss die Datei vor dem Funktionsaufruf eingebunden werden. Wird `require()` erst hinter dem Funktionsaufruf verwendet, kennt PHP den ausgelagerten Code noch nicht und es kommt zum Fehler.

Datei: *eigene_funktion.inc.php*

Innerhalb der Datei *eigene_funktion.inc.php* wurde eine einfache Funktion definiert, die nach der Einbindung in der aufrufenden Datei zur Verfügung steht.

```
<?php
function summe($zahl1, $zahl2) {
    $ergebnis = $zahl1 + $zahl2;
    echo "<p>Summenbildung: $zahl1 + $zahl2 = " . $ergebnis . "</p>";
}
?>
```

require: Verwenden einer Funktion aus einer anderen Datei

Ein fremdes PHP-Skript wird eingebunden und die dort enthaltene Funktion ausgeführt:

Summenbildung: 2000 + 1376 = 3376

Hier kann das Skript weitere Anweisungen enthalten.

Anzeige der Beispieldatei "require.php"

8.5 Andere Dateitypen einbinden

Sie können auch andere Dateien, z. B. reine Text- oder HTML-Dateien, in Ihr Programm einbinden.

Innerhalb eines PHP-Skripts soll beispielsweise eine einfache Textdatei eingebunden werden. Der Inhalt der Textdatei wird sofort am Bildschirm ausgegeben. Zuvor wird eine Textdatei mit folgendem Inhalt gespeichert.

Beispiel: *einfueg.txt*

```
<strong>Dies ist der Inhalt der eingebundenen Textdatei!</strong>
```

Beispiel: *include.php*

```
<h1>Eine Textdatei über den Befehl <em>include</em> einbinden</h1>
<?php
    echo "<p>Es folgt die eingebundene Text-Datei:</p><hr>";
    include("einfueg.txt");
    echo "<hr><p>Hier kann das Skript weitere Anweisungen enthalten.</p>";
?>
```

Inmitten des PHP-Codes wird über den Befehl `include()` die Textdatei eingebunden. Da diese keinen weiteren PHP-Code, sondern nur Text mit HTML-Befehlen beinhaltet, wird diese von PHP direkt an den Browser weitergeleitet.



Anzeige der Beispieldatei "include.php"

8.6 Übung

Mit Funktionen arbeiten

Übungsdatei: -

Ergebnisdatei: *funktion_uebung.php*

- ① Erstellen Sie eine PHP-Datei *funktion_uebung.php*, in der Sie zwei Funktionen definieren: `addiere()` und `multiplizierte()`. Beide Funktionen sollen Berechnungen gemäß ihren Namen durchführen: Zahlen addieren bzw. multiplizieren.
- ② In den Funktionsaufrufen sollen zwei Parameter eingegeben werden müssen, ein definierter dritter Parameter ist optional. (Überlegen Sie, welche Standardwerte Sie für den dritten Parameter vergeben müssen, damit die Funktionen richtige Berechnungen durchführen.) Eine Ausgabe des Ergebnisses soll zusammen mit einem kurzen Text, welche Berechnung mit welchen Zahlen vorgenommen wurde, direkt in den Funktionen erfolgen.
- ③ Erweitern Sie Ihr Programm um vier Funktionsaufrufe:
Funktionsaufrufe 1 und 2: `addiere()`, `multiplizierte()`
mit jeweils drei Parametern: 8, 4 und 2.
Funktionsaufrufe 3 und 4: `addiere()`, `multiplizierte()`
mit jeweils zwei Parametern: 8 und 4.



Anzeige der Beispiellösung
"funktion_uebung.php"

Funktionen über ein Formular aufrufen

Übungsdatei: -

Ergebnisdateien: *funktion.inc.php, form_uebung.php*

- ④ Kopieren Sie die Funktionsdefinition `addiere()` und speichern Sie diese in eine andere Datei mit dem Namen *funktion.inc.php*.
- ⑤ Erstellen Sie ein Formular mit dem Dateinamen *form_uebung.php* mit Eingabefeldern für drei Zahlen und Schaltflächen zum Absenden und Zurücksetzen des Formulars.
- ⑥ Die Datei soll sich beim Absenden des Formulars selbst aufrufen. Die Funktion `addiere()` soll in die Datei *form_uebung.php* eingebunden und von hier aus aufgerufen werden, nachdem das Formular versendet wurde.

Anzeige der Beispiellösung
"form_uebung.php"

9 Mit Daten aus externen Dateien arbeiten

In diesem Kapitel erfahren Sie

- ✓ wie der Zugriff auf Dateien erfolgt
- ✓ wie Sie Daten aus externen Dateien einlesen können
- ✓ wie Sie Daten in externe Dateien schreiben können

Voraussetzungen

- ✓ Mit verschiedenen Dateitypen arbeiten
- ✓ Arbeiten mit Kontrollstrukturen und Arrays

9.1 Externe Dateien nutzen

Beispiel für die Nutzung von Daten aus externen Dateien

Wenn Sie Daten, die die Benutzer in ein Formular eingegeben haben, z. B. in einer Excel-Tabelle auswerten oder bearbeiten möchten, können Sie diese außerhalb des PHP-Skripts in einer externen Datei speichern. So können Sie beispielsweise auf diese Datei zugreifen, die Daten lesen, aber auch neue Daten hinzufügen bzw. die Daten ersetzen.

Beispiele für die sinnvolle Nutzung von externen Dateien sind:

- ✓ Protokolldateien schreiben:
Um z. B. die Anzahl der Aufrufe einer Seite zu protokollieren, können Sie Daten in einer eigenen Datei speichern.
- ✓ Dynamische Daten in eine Webseite einbauen:
Beispielsweise können Sie einen Hinweis auf das Angebot des Tages in einer externen Textdatei speichern. So brauchen Sie, um die Daten zu verändern, nur die externe Datei zu bearbeiten.
- ✓ Eingaben aus einem Formular speichern, falls Sie keine Datenbank einsetzen können bzw. wollen.

Wichtig bei der Bearbeitung von externen Dateien ist die Art der Datei. Hierbei kann PHP unterscheiden zwischen dem Zugriff auf Textdateien und binäre Dateien.

Textdateien (*.txt) oder Textdateien im Tabellenformat (*.csv) beinhalten Zeilen unterschiedlicher Länge. Sie werden sowohl beim Schreiben als auch beim Lesen zeilenweise bearbeitet.

Im folgenden Kapitel wird der Zugriff auf Textdateien beschrieben.

9.2 Dateien öffnen, lesen und schließen

Dateien mit fopen() öffnen

Bevor Sie auf die Daten in Dateien zugreifen können, müssen Sie die entsprechende Datei öffnen. Dazu können Sie die Funktion fopen() verwenden.

Syntax und Bedeutung der Funktion fopen()

```
①      ②  
fopen("Pfad/Dateiname", "Modus");  
$handle = fopen("user.txt", "r")  
③
```

Zusätzlich zur Angabe der zu öffnenden Datei ① wird der Modus ② angegeben, in dem die Datei geöffnet werden soll:

- 'r' Die Datei wird nur zum Lesen (r = *read*) geöffnet, der Dateizeiger wird auf den Anfang der Datei positioniert.
- 'r+' Die Datei wird zum Lesen und Schreiben geöffnet und der Dateizeiger wird auf den Anfang der Datei gesetzt.
- 'w' Öffnet eine Datei zum Schreiben (w = *write*). Existiert die Datei nicht, wird sie angelegt. Der Dateizeiger wird auf den Anfang der Datei gesetzt und die Länge der Datei auf 0 Byte. Eventuell vorhandene Daten werden gelöscht.
- 'w+' Hier handelt es sich um dieselbe Option wie die Option 'w', nur wird hier die Datei zum Lesen und Schreiben geöffnet.
- 'a' Die Datei wird zum Schreiben geöffnet, der Dateizeiger wird auf das Ende der Datei gesetzt. Neue Daten ergänzen vorhandene Daten (a = *append*).
- 'a+' Wie Option 'a', die Datei wird zusätzlich zum Lesen geöffnet.
- 'x' Erzeugt und öffnet eine Datei zum Schreiben und setzt den Dateizeiger auf den Dateianfang. Falls die Datei bereits vorhanden ist, liefert `fopen()` FALSE zurück.
- 'x+' Erzeugt und öffnet eine Datei zum Lesen und Schreiben, sonst wie 'x'.
- 'c' Öffnet eine Datei zum Schreiben, falls die Datei nicht existiert, wird sie erzeugt. Wenn die Datei existiert, wird sie im Gegensatz zu 'w' nicht gekürzt, vorhandene Daten bleiben erhalten.
- 'c+' Öffnet die Datei zum Lesen und Schreiben, ansonsten wie 'c'.

Der Rückgabewert der Funktion `fopen()` ist ein Zeiger auf die geöffnete Datei (sogenanntes Handle) und wird für weitere Zugriffe auf die Datei genutzt. Im Beispiel wird der Dateizeiger in der Variablen \$handle ③ gespeichert. Die Variable \$handle hat den Datentyp *resource*.

Scheitert das Öffnen der angegebenen Datei z. B. wegen fehlender Schreib- und Lese-Berechtigungen, gibt die Funktion `fopen()` den Wert FALSE zurück.



Prüfung auf Existenz einer Datei mit `file_exists()`

Mit der Funktion `file_exists()` können Sie prüfen, ob eine Datei oder ein Verzeichnis existiert.

Syntax und Bedeutung der Funktion `file_exists()`

```
file_exists(Name);
```

Wurde die angegebene Datei oder das angegebene Verzeichnis gefunden, liefert die Funktion TRUE zurück, anderenfalls FALSE.

Dateien mit `fgets()` lesen

Mithilfe der Funktion `fgets()` können Sie – von der Position des Dateizeigers – eine Zeile einer Datei auslesen.

Syntax und Bedeutung der Funktion `fgets()`

```
fgets(Dateizeiger[, Modus]);
$zeile = fgets($handle);
```

- ✓ Im ersten Parameter (Dateizeiger) finden Sie die Angabe zur Datei, aus der gelesen werden soll. Hierbei wird der Dateizeiger im Beispiel \$handle ① aus der vorher geöffneten Datei verwendet.
- ✓ `fgets()` liest die angegebene Datei zeilenweise aus, wobei Sie optional im Parameter Modus die Länge angeben können. Lassen Sie diesen Parameter weg, wird automatisch so lange gelesen, bis das Ende der Zeile erreicht ist.
- ✓ `fgets()` liest bis zu dem ersten Auftreten eines Zeilenumbruchs bzw. der maximalen Zeilenlänge, dem Ende der Datei (EOF = *End of File*) oder bis zu der im Parameter Modus angegebenen Länge.

- ✓ Jeder Aufruf von `fgets()` bewegt den Dateizeiger eine Zeile weiter, bis das Ende der Datei erreicht ist. Wenn Sie `fgets()` erneut aufrufen, wird die nächste Zeile gelesen. Wollen Sie eine Datei komplett auslesen, bietet sich die Verwendung von `fgets()` in einer Schleife an.
- ✓ Der Rückgabewert der Funktion `fgets()` ist eine Zeichenkette. Im Beispiel wird der Rückgabewert in der Variablen `$zeile` gespeichert.

Prüfung auf Dateiende mit `feof()`

Wenn Sie z. B. eine Datei mit einer Schleife Zeile für Zeile auslesen wollen, ist es sinnvoll, mithilfe dieser Funktion zu prüfen, ob das Ende der Datei erreicht wurde.

Syntax und Bedeutung der Funktion `feof()`

```
feof(Dateizeiger);
```

Ob der Dateizeiger am Ende der Datei steht, prüfen Sie mithilfe der Funktion `feof(Dateizeiger)`. Sie liefert `TRUE` zurück, wenn der Dateizeiger am Ende der Datei steht, anderenfalls `FALSE`.

Dateien mit `fclose()` schließen



Nachdem Sie eine Datei geöffnet und die Daten ausgelesen haben, müssen Sie die Datei wieder schließen, um sie für andere Prozesse wieder nutzbar zu machen.

Syntax und Bedeutung der `fclose()`-Anweisung

```
fclose(Dateizeiger);
fclose($handle);
```

- ✓ Mit der Funktion `fclose()` schließen Sie die Datei, auf die der Dateizeiger weist. Der Dateizeiger muss gültig sein. Die Datei, auf die gezeigt wird, muss zuvor mit der Funktion `fopen()` geöffnet worden sein.
- ✓ Wurde die Datei erfolgreich geschlossen, liefert die Funktion `fclose()` den Wert `TRUE` zurück, sonst `FALSE`.

Beispiel zu externe Datei öffnen, lesen und schließen: `fgets.php`

```
<?php
① $handle = fopen("protokoll.txt", "r");
② if ($handle != FALSE) {
    $i = 1; // Zähler für Zeilennummer
    ③ while (!feof($handle)) { // Schleife bis zum Dateiende
        ④ $zeile = fgets($handle); // liest aktuelle Zeile
        echo "<em>Zeile $i:</em> " . $zeile . "<br>";
        ⑤ $i++; // nach jedem Schleifendurchlauf Zähler erhöhen
    }
    ⑥ fclose($handle);
} else {
    echo "Beim Öffnen der Datei trat ein Fehler auf.";
}
?>
```

- ① Die Datei `protokoll.txt` wird zum Lesen geöffnet. `fopen()` gibt einen Dateizeiger (*Handle*) zurück, dieser wird in der `resource`-Variablen `$handle` gespeichert.
- ② In der `if`-Anweisung wird geprüft, ob die Datei geöffnet werden konnte. Falls ja, wird der `if`-Zweig ausgeführt, anderenfalls der `else`-Zweig.

- ③ Um den kompletten Inhalt der Datei auszulesen, wird eine while-Schleife verwendet, die so lange durchlaufen wird, bis das Ende der auszulesenden Datei erreicht ist. Ob der Dateizeiger am Ende der Datei steht, prüfen Sie mit der Funktion `feof()`. Solange das Ende der Datei nicht erreicht ist, gibt `feof() FALSE` zurück, die while-Schleife wird weiter ausgeführt. Am Ende der Datei gibt `feof() TRUE` zurück, damit wird die while-Schleife dann beendet.
- ④ Die aktuelle Zeile wird ausgelesen und in der Variablen `$zeile` gespeichert. Nach dem Auslesen der Zeile wird der Dateizeiger beim Einsatz der Funktion `fgets()` automatisch auf die nächste Zeile gesetzt, die dann im nächsten Durchlauf der Schleife ausgelesen wird.
- ⑤ Die Variable `$zeile` wird per `echo` ausgegeben.
- ⑥ Die Datei `protokoll.txt` wird nach der while-Schleife geschlossen.

```
Zeile 1: 18.02.2015 16:07:01 Dienst wurde erfolgreich gestartet.  
Zeile 2: 18.02.2015 19:30:31 Statusabfrage: Alles in Ordnung!  
Zeile 3: 18.02.2015 22:47:13 Dienst wurde erfolgreich beendet.
```

Anzeige der Beispieldatei "fget.php"

9.3 Weitere Möglichkeiten zum Lesen von Dateien

Dateien mit `readfile()` oder `file()` lesen

Während die Funktion `fgets()` den Inhalt einer Datei zeilenweise ausliest, können Sie mit den Befehlen `readfile()` und `file()` den gesamten Inhalt einer Datei auf einmal wiedergeben.

Der Unterschied zwischen den beiden Funktionen `readfile()` und `file()` liegt in der Ausgabe der Daten:

- ✓ `readfile()` liest den Inhalt der Datei vollständig aus und sendet das Ergebnis ohne weitere Bearbeitung und Weiterverarbeitungsmöglichkeit direkt an den Browser. Zeilenumbrüche z. B. werden nicht dargestellt, wenn sie nicht als HTML-Tags in der eingelesenen Datei vorhanden sind.
- ✓ `file()` hingegen liest den vollständigen Inhalt der Datei in ein Array ein. Hierbei wird jeweils eine Zeile zu einem Eintrag im Array.

```
readfile("Datei");
file("Datei");
```

Beispiel `readfile.php` und `file.php`

<pre><h1>Auslesen einer Textdatei</h1> <?php readfile("protokoll.txt"); ?></pre>	<pre><h1>Auslesen einer Textdatei</h1> <?php \$feld = file("protokoll.txt"); echo "<p>"; \$i = 1; foreach (\$feld as \$zeile) { echo "Zeile " . \$i++ . ": "; echo \$zeile . "
"; } echo "</p>"; ?></pre>
--	--



Beispieldatei "readfile.php"



Beispieldatei "file.php"

Dateien mit `file_get_contents()` lesen

Die Funktion `file_get_contents()` ist die flexibelste Methode, Inhalte einer Datei einzulesen. Dabei werden die Inhalte in einem String gespeichert. Eine Weiterverarbeitung ist so problemlos möglich.

Beim Einlesen von Dateien besteht häufig das Problem, dass in der Datei befindliche Zeilenumbrüche zwar erkannt werden, aber z. B. durch fehlende HTML-Tags in Textdateien nicht als Zeilenwechsel bei der Ausgabe dargestellt werden können. PHP bietet für diese Zwecke die Hilfsfunktion `nl2br()` (gesprochen: "new line to break", z. B. "Wandle Zeilenumbrüche in
-Tags um"). Die Funktion erfüllt dann genau diese Aufgabe, sie erkennt Zeilenwechsel in externen Dateien und wandelt diese in HTML-Tags für einen Zeilenumbruch
 um. Der zweite Parameter ist optional. Wird dieser nicht angegeben, werden einfache
-Tags verwendet. Übergeben Sie `true`, wird ein
-Tag mit schließendem Backslash verwendet. Diese benötigen Sie, wenn Sie XHTML-konformes HTML generieren müssen.

Beispiel: `file_get_contents.php`

```
<h1>Auslesen einer Textdatei mit <em>file_get_contents()</em></h1>
<?php
①    $inhalt = file_get_contents("protokoll.txt");
②    echo nl2br($inhalt, FALSE); // nl2br: Zeilenumbrüche erkennen und
        // in <br />-Befehle umwandeln
    ?>
```

- ① Über die Funktion `file_get_contents()` wird der Inhalt der Datei `protokoll.txt` in einen String eingelesen und unter dem Variablennamen `$inhalt` gespeichert. Da HTML5 das Tag `
` in seiner alten Schreibweise zulässt, wird hier der zweite Parameter auf `FALSE` gesetzt.
- ② Die Variable `$inhalt` wird auf dem Bildschirm ausgegeben. Die Funktion `nl2br()` wird verwendet, um Zeilenumbrüche, die in der eingelesenen Datei vorhanden sind, in
-Tags umzuwandeln und so auch Zeilenumbrüche in der Ausgabe darzustellen.



Anzeige der Beispieldatei "file_get_contents.php"

Inhalt einer Datei in eine Zeichenkette einlesen

Sie können die gesamte Datei in einen String einlesen. Hierbei lesen Sie eine Datei über die Funktion `file()` in ein Array ein. Jede Zeile aus der Datei erzeugt einen Eintrag in diesem indizierten Array. Sie verwenden dann die Funktion `implode()`, die Array-Elemente zu einer Zeichenkette verbindet.

Beim Aufruf der Funktion `implode()` können Sie als Trennzeichen ein Zeichen oder eine Zeichenkette angeben, anhand deren das Array zu einem String zusammengefügt wird. Dies kann auch ein HTML-Tag sein, z. B. für die zeilenweise Darstellung ein
-Tag.

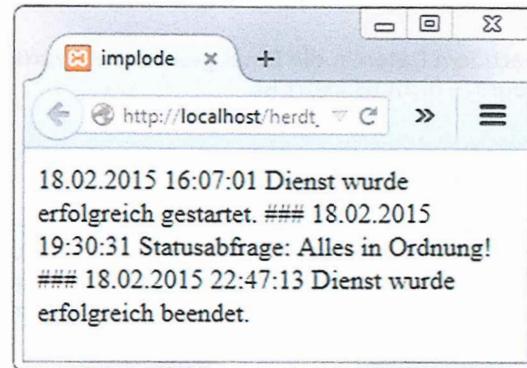
```
implode(Trennzeichen, Array);
```

Beispiel: *implode.php*

```

① <?php
②   if (file_exists("protokoll.txt")) {
      $inhalt = implode(" ### ", file("protokoll.txt"));
      echo $inhalt;
    }
?>
```

- ① Mithilfe der Funktion `file_exists()` wird überprüft, ob die angegebene Datei existiert. Die Funktion `file_exists()` erwartet als einzigen Parameter den Namen einer Datei auf dem Webserver. Dateien auf anderen Webservern (URL) können nicht auf ihre Existenz überprüft werden, da man dort nicht über ausreichende Zugriffsrechte verfügt.
- ② Der Befehl `implode()` verbindet das über die Funktion `file()` eingelesene Array aus der Datei `protokoll.txt` zu einer Zeichenkette. Als Trennzeichen wird hierbei die Zeichenfolge " ### " verwendet.



Anzeige der Beispieldatei "implode.php"

In den Beispielen liegen die zu öffnenden Dateien im selben Verzeichnis wie die PHP-Dateien. Liegen die Dateien in einem anderen Verzeichnis, können Sie diese mit relativen Pfaden (z. B. `fopen("../files/protokoll.txt", "r")`) ansprechen. Am zuverlässigsten sind allerdings absolute Pfade (z. B. "C:\\xampp\\htdocs\\herdt_php56_grundlagen\\beispiele\\kap09\\");), die sich auf das Betriebssystem beziehen. Im Falle von Windows müssen die Backslashes escaped werden. Der Zugriff über eine URL, z. B. `http://localhost/herdt_php56_grundlagen/beispiele/kap09/` funktioniert nicht. Beim Provider können Sie den Pfad über die superglobale `$_SERVER`-Variable auslesen.



9.4 In Dateien schreiben

Dateien zum Schreiben öffnen

Wollen Sie Daten in eine Datei schreiben, müssen Sie die Datei zuerst öffnen. Beim Öffnen der Datei können Sie bestimmen, ob Sie die bestehenden Daten der Datei überschreiben wollen oder der bestehenden Datei weitere Daten hinzufügen möchten.

(1) Dateien überschreiben

Wenn Sie eine Datei mit der Funktion `fopen()` und dem Modus 'w' für *write* öffnen, wird die Datei zum Schreiben geöffnet und der Dateizeiger auf den Anfang der Datei verschoben. Gleichzeitig wird die Länge der Datei auf 0 Byte gesetzt. Das bedeutet, dass vorhandene Inhalte gelöscht werden. Wenn die Datei nicht existiert, wird sie angelegt.

(2) Daten in Dateien hinzufügen

Um Daten fortlaufend in eine Datei zu schreiben, verwenden Sie den Befehl `fopen()` mit dem Modus 'a'. Die Datei wird zum Schreiben geöffnet und der Dateizeiger an das Ende der Datei gesetzt, sodass die neuen Daten hinzugefügt werden. Existiert sie nicht, legt PHP sie an.

Daten in Dateien schreiben

Um eine Zeichenkette in eine geöffnete Datei zu schreiben, verwenden Sie die Funktion `fputs()`.

Syntax und Bedeutung der Funktion fputs()

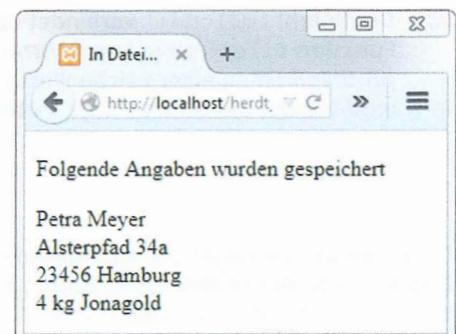
```
fputs(Dateizeiger, Zeichenkette[, Länge]);
```

- ✓ Mit der Funktion `fputs()` können Sie eine beliebige Zeichenkette in eine Datei schreiben.
- ✓ Die entsprechende Datei wird über den von `fopen()` zurückgegebenem Dateizeiger angesprochen.
- ✓ Ist der optionale Parameter `Länge` angegeben, wird das Schreiben nach der angegebenen Anzahl Bytes beendet. Geben Sie ihn nicht an, wird die gesamte Zeichenkette geschrieben.
- ✓ `fputs()` gibt bei Erfolg die Anzahl der geschriebenen Bytes zurück, andernfalls `FALSE`.

Nachdem Daten in die Datei geschrieben wurden, muss die Datei geschlossen werden, damit sie für andere Zugriffe nicht gesperrt ist.

Beispiel: *bestellung.php*

Die Daten, die der Benutzer in ein Bestellformular für Äpfel (Datei *bestellformular.html*) eingibt, sollen in eine *.csv-Datei (*bestellung_daten.csv*) geschrieben werden. Existiert die *.csv-Datei noch nicht, wird sie angelegt, andernfalls werden die Daten an das Ende der Datei geschrieben.



Anzeige nach Absendung des Formulars "bestellformular.html"

```
<?php
① $handle = fopen("bestellung_daten.csv", "a");
② if ($handle == FALSE) {
    echo "<p>Datei konnte nicht zum Schreiben geöffnet werden</p>";
    exit("<p>Das Programm wird beendet.</p>");
}
④ $name = $_POST["name"];
$strasse = $_POST["strasse"];
$ort = $_POST["ort"];
$sorte = $_POST["sorte"];
$menge = $_POST["menge"];
⑤ if (fputs($handle, utf8_decode("$name;$strasse;$ort;$sorte;$menge;\n")) ) {
    echo "<p>Folgende Angaben wurden gespeichert</p>";
    echo "<p>$name<br>$strasse<br>$ort<br>$menge kg $sorte</p>";
} else {
    echo "<p>Das Schreiben der Daten ist fehlgeschlagen.</p>";
}
⑥ fclose($handle);
?>
```

- ① Die Datei *bestellung_daten.csv* wird mit `fopen()` im Modus 'a' geöffnet, d. h. zum Schreiben und Anhängen neuer Daten. Der Dateizeiger wird an das Ende der Datei gesetzt, sodass die neuen Daten hinzugefügt werden.
- ② Es wird geprüft, ob das Öffnen der Datei erfolgreich war.

- ③ Wenn die Datei nicht erfolgreich geöffnet werden kann, z. B. weil Sie nicht über die nötigen Berechtigungen im Verzeichnis verfügen, wird mit `exit()` das Programm sofort beendet. Die Funktion `exit()` kann auch ohne Parameter aufgerufen werden. Wollen Sie eine Meldung über das Beenden des Skripts ausgeben, können Sie wie hier im PHP-Code `exit()` mit einer Zeichenkette aufrufen, welche vor dem Beenden des Skripts ausgegeben wird. Diese Zeichenkette kann auch HTML-Tags enthalten.
- ④ Die Variablen `$name`, `$strasse`, `$ort`, `$sorte` und `$menge` werden mit den übergebenen Formulardaten gefüllt.
- ⑤ Über die Funktion `fputs()` werden die Werte der Variablen in die geöffnete Datei geschrieben. Hierbei werden die Daten jeweils durch ein Semikolon voneinander getrennt und am Ende einer Zeile wird mithilfe von "`\n`" ein Zeilenvorschub erzeugt, um die neuen Daten in eine eigene Zeile zu schreiben. Im Erfolgs- bzw. im Fehlerfall wird eine entsprechende Meldung ausgegeben. Zusätzlich sehen Sie, dass die einzufügende Zeichenkette von der PHP-Funktion `utf8_decode()` umschlossen ist. Um für Windows eine Datei mit dem ISO-8859-1-Zeichensatz zu erstellen, müssen UTF-8-Zeichen konvertiert werden.
- ⑥ Die geöffnete Datei wird geschlossen.

*.csv-Dateien können Sie mit einem Programm wie z. B. *LibreOffice - Calc* oder *Microsoft Excel* öffnen und bearbeiten.



	A	B	C	D	E
1	Peter Schmidt	Mainufer 307	61234 Frankfurt/Main	Elstar	15
2	Arndt Hoffmann	Donauweg 66	87654 München	Gala	8
3	Petra Meyer	Alsterpfad 34a	23456 Hamburg	Jonagold	4

Ansicht der Datei "bestellung_daten.csv" in Microsoft Excel

Daten mit der Funktion `file_put_contents()` schreiben

`file_put_contents()` ist eine neuere Funktion in PHP, die das Schreiben einer Zeichenkette in eine Datei vereinfacht. Es handelt sich um eine Funktion, die die bei `fputs()` nötigen Vorgänge `fopen() → fputs() → fclose()` zusammenfasst. Es genügt der Aufruf der Funktion.

Syntax und Bedeutung der Funktion `file_put_contents()`

```
file_put_contents(Dateiname, Zeichenkette[, Flag]);
```

- ✓ Die Funktion ist identisch mit der Ausführung von 1) `fopen()`, 2) `fputs()` und 3) `fclose()`.
- ✓ Geben Sie den optionalen Parameter `Flag` nicht an, werden eventuell vorhandene Daten überschrieben. Durch Angabe von `FILE_APPEND` wird eine Zeichenkette bei Aufruf der Funktion am Ende der Datei angefügt.

Beispiel: `file_put_contents.php`

```
<?php
① $file = "text.txt";
② if (file_put_contents($file, "Zeichenkette Aufruf 1\r\n")) {
    echo "<p>Funktionsaufruf 1: erfolgreich.</p>";
}
③ if (file_put_contents($file, "Zeichenkette Aufruf 2\r\n", FILE_APPEND)) {
    echo "<p>Funktionsaufruf 2: erfolgreich.</p>";
}
?>
```

- ① Der Name der Datei, in die geschrieben werden soll, wird festgelegt und der Variablen \$file zugewiesen.
- ② file_put_contents() wird aufgerufen, die Datei text.txt wird angelegt und die angegebene Beispiel-Zeichenkette in die Datei geschrieben. Liefert die Funktion bei erfolgreichem Schreiben TRUE zurück, wird auf dem Bildschirm eine Meldung ausgegeben. Damit auch der Windows-Editor die Datei richtig anzeigen kann, wird in diesem Beispiel die Windows-Variante des Zeilenumbruchs "\r\n" verwendet. In Linux-Systemen reicht die Angabe von "\n" aus.
- ③ Beim nochmaligen Aufruf der Funktion file_put_contents() wird das optionale Flag FILE_APPEND angegeben. Dies bewirkt, dass die Zeichenkette der bestehenden Datei angefügt wird. Vorhandene Inhalte werden nicht überschrieben.

9.5 Zugriffszähler für eine Webseite

Eine Anwendung in Verbindung mit Dateizugriffen ist die Einbettung eines Counters (Zugriffszählers) in eine Webseite. Ein Counter zählt z. B. die Anzahl der Seiten-Zugriffe. Dabei wird bei jedem Zugriff der Wert des Zählers um eins erhöht. Das heißt, es wird auf eine bestehende Datei zugegriffen, der Inhalt, also die Zahl wird ausgelesen und um den Wert 1 erhöht. Das Ergebnis der Berechnung überschreibt dann den Inhalt der Datei.

Beispiel: counter.php

```

① <p>Der Stand des Zählers ist: <?php counter() ?></p>
<?php
② function counter() {
③     $name = "counter.txt";
④     $handle = fopen($name, "r+");
⑤     if ($handle) {
⑥         flock($handle, LOCK_EX);
⑦         $count = fgets($handle, 10);
⑧         fseek($handle, 0);
⑨         // Zähler erhöhen und ausgeben
⑩         echo "<strong>" . ++$count . "</strong>";
        fputs($handle, $count);
        fclose($handle);
    }
}
?>
```

- ① Im HTML-Text wird die Funktion counter() zur Anzeige der Besucherzahl aufgerufen.
- ② Die Funktion counter() wird definiert.
- ③ Der Name der Datei, in der die Anzahl der Besucher abgelegt werden soll, lautet counter.txt und wird der Variablen \$name zugewiesen.
- ④ Die Datei counter.txt wird über die Funktion fopen() zum Lesen und Schreiben geöffnet. Der Zugriff auf diese Datei erfolgt über den Dateizeiger \$handle. Der Dateizeiger wird mit dem Modus r+ an den Anfang der Datei gesetzt.
- ⑤ Damit während der Arbeit mit der Datei kein weiterer Zugriff und somit kein gleichzeitiges Überschreiben der Daten möglich ist, wird die Datei für andere Zugriffe (LOCK_EX) gesperrt. LOCK_EX steht für eine exklusive, nur schreibende Verriegelung einer Datei (vgl. folgenden Abschnitt).
- ⑥ Mit fgets() werden die ersten 10 Zeichen der Datei ausgelesen. Der Wert 10 wurde gewählt, da hiermit die Verarbeitung einer Zahl mit 10 Stellen (bis 9999999999) gewährleistet ist.

- ⑦ Damit der neue Wert den alten Wert in der Datei überschreiben kann, wird der Dateizeiger mittels `fseek()` und der Angabe `0` an den Anfang der Datei gesetzt (vgl. folgenden Abschnitt).
- ⑧ Nach dem Einlesen des Wertes wird er um eins erhöht und am Bildschirm fett formatiert ausgegeben.
- ⑨ Über die Funktion `fputs()` wird der neue Wert der Variablen `$count` in die Datei geschrieben.
- ⑩ `fclose()` schließt die Datei `counter.txt` und beendet zugleich die Zugriffssperre, die über `flock()` veranlasst wurde.



Ausgabe der Beispieldatei "counter.php"

Mit dem Befehl `flock()` können Sie eine Datei in der Zeit, in der das Skript in eine Datei schreibt, zusätzlich für weitere Zugriffe sperren.



Syntax und Bedeutung der Funktion `flock()`

```
flock(Dateizeiger, Modus);
```

- ✓ `flock()` sperrt den Zugriff auf die Datei, auf die der Dateizeiger verweist. Der Modus ermöglicht verschiedene Sperrzustände der Datei.
- ✓ Darf während Ihres Zugriffs die Datei von anderen nur gelesen werden, setzen Sie den Modus auf `LOCK_SH` (Shared Lock, Lesezugriff). Soll kein anderer Nutzer zeitgleich die Datei nutzen dürfen, setzen Sie die Option `LOCK_EX` (Exclusive Lock). `flock()` wartet, bis die Datei wie angegeben benutzt werden kann. Geben Sie `LOCK_NB` (No Block) an, kehrt die Funktion mit dem Wert `FALSE` zurück, wenn die Datei bereits von einem anderen Programm gesperrt ist. Möchten Sie die Verriegelung wieder freigeben, geben Sie `LOCK_UN` (Unlock) an.
- ✓ Die Funktion liefert bei Erfolg den Wert `TRUE` zurück bzw. `FALSE`, wenn ein Fehler auftrat. Ob die Zugriffssicherung funktioniert, erfahren Sie, indem Sie den Rückgabewert überprüfen.
- ✓ Die Sperre wird über die Funktion `fclose()` oder am Ende des Skripts automatisch aufgehoben.

Bei manchen Betriebssystemen ist die Funktion `flock()` auf Prozess-Ebene (Teil einer Applikation) implementiert. Hierbei können Sie sich nicht auf `flock()` verlassen, um Dateien vor dem Zugriff von anderen PHP-Skripten zu schützen. Alle PHP-Skripte laufen auf dem Server in einem gemeinsamen Prozess.



Zudem wird `flock()` nicht von allen Dateisystemen unterstützt. In diesem Fall gibt es in solchen Umgebungen immer `FALSE` zurück.

Dateizeiger mit `fseek()` an eine bestimmte Position setzen

Im vorher gezeigten Beispiel ist es notwendig, den Dateizeiger an den Anfang der Datei zu setzen, um den alten Wert des Zählers mit dem neuen Wert überschreiben zu können. Den Dateizeiger können Sie über `fseek()` positionieren.

```
fseek(Dateizeiger, Stelle [, Wie]);
```

- ✓ Hierbei geben Sie für den Parameter `Stelle` die Anzahl in Bytes, bezogen auf den Dateianfang, der durch den Dateizeiger festgelegt wird, an.
- ✓ Der optionale Parameter `Wie` legt bestimmte Bezugspunkte für die Ermittlung der Position fest:
Der Parameter `SEEK_CUR` ermöglicht die Verschiebung auf die aktuelle Position des Dateizeigers plus `Stelle`, `SEEK_END` vom Dateiende und `SEEK_SET` vom Dateianfang.
- ✓ Fehlt der Parameter `Wie`, ist standardmäßig die Option `SEEK_SET` gewählt.

9.6 Übung

Mit externen Dateien arbeiten

Übungsdatei: --

Ergebnisdatei: **kapA\Antworten (Theorie).doc**

- ① Sie möchten eine Datei zum Einlesen und Schreiben öffnen. Über die Funktion `fopen()` können Sie die verschiedenen Modi angeben. Erläutern Sie die Unterschiede der Optionen `r`, `w` und `a`.
- ② Eine Datei soll zum Lesen geöffnet werden und weitere Daten sollen an das Ende der Datei geschrieben werden. Nennen Sie den Modus, in dem die Datei geöffnet werden muss.

Umfrage mit Gewinnspiel

Übungsdatei: --

Ergebnisdateien: **formular_umfrage.html, umfrage.php, umfrage_daten.csv**

- ③ Sie möchten eine Umfrage durchführen. Hierfür erstellen Sie ein Formular, in das Daten eingegeben werden. Um die Daten auswerten zu können, schreiben Sie ein PHP-Skript, das die Daten an eine Tabelle (*.csv) übergibt und dort speichert.
- ④ Das Umfrageformular speichern Sie unter dem Namen `formular_umfrage.html`, das auswertende PHP-Skript unter dem Namen `umfrage.php` und die übergebenen Daten werden in der Datei `umfrage_daten.csv` gespeichert.

Umfrage mit Gewinnspiel

Wir freuen uns, dass Sie an unserer kleinen Umfrage zu unserer Webseite teilnehmen! Unter allen Teilnehmern verlosen wir drei Präsentkörbe.

Vorname und Nachname _____

Straße _____

PLZ und Ort _____

Bitte wählen Sie die Antwort aus, die für Sie am ehesten zutrifft:

Wie gefällt Ihnen unser Internetangebot?

- sehr gut
- gut
- nicht so gut
- gar nicht

Wie beurteilen Sie den Informationsgehalt unserer Webseite?

- sehr informativ
- die eine oder andere Information fehlt
- es fehlen sehr viele wichtige Informationen

Wie kommen Sie mit dem Bestellsystem zurecht?

- sehr gut
- gut
- nicht besonders gut
- gar nicht

Möchten Sie uns noch etwas mitteilen?

Formular "formular_umfrage.html"

10 Zeichenketten-Funktionen

In diesem Kapitel erfahren Sie

- ✓ wie Sie Zeichenketten formatiert ausgeben
- ✓ wie Sie nach Zeichenfolgen suchen
- ✓ wie Zeichenketten modifiziert werden

Voraussetzungen

- ✓ PHP-Grundlagen
- ✓ Arrays

10.1 Zeichenketten ausgeben

Daten liegen abhängig von ihrer Herkunft häufig als Rohdaten vor, haben keine besondere Formatierung und sind für die Ausgabe nicht immer geeignet. PHP bietet vielfältige Funktionen an, mit denen Sie Daten in eine bestimmte optische Form bringen können, die entweder besser lesbar ist oder bestimmten Konventionen (beispielsweise für die Darstellung von Fließkommazahlen, einer Telefonnummer oder eines Datums) der Sprache entsprechen.

Zeichenketten mit der Funktion `printf()` formatieren

Wenn Sie Daten für die Ausgabe im Browser formatieren möchten, verwenden Sie die Funktion `printf()`.

`printf` (*print formatted = formatierte Ausgabe*) ermöglicht Ihnen, durch die Angabe spezieller Optionen festzulegen, wie eine Zahl oder eine Zeichenkette ausgegeben werden soll. Ein Beispiel ist die Ausgabe von Fließkommazahlen. Bei Berechnungen erhalten Sie als Rückgabe mitunter eine Fließkommazahl mit mehreren Nachkommastellen. Sie möchten beispielsweise nur eine oder zwei Stellen nach dem Komma am Bildschirm ausgeben, z. B. 10.5 km oder 19.50 €.

Syntax der Funktion `printf()`

```
printf(Formatierung[, Argument1[, Argument2, ...]]);
```

- ✓ Der Parameter `Formatierung` ist eine Zeichenkette, worin Anweisungen für eine formatierte Ausgabe enthalten sind.
- ✓ Die `Formatierung` kann keine, eine oder mehrere Anweisungen beinhalten.
- ✓ Jede `Formatierungsanweisung` besteht aus einem Prozentzeichen gefolgt von einem oder mehreren Elementen, die eine bestimmte `Formatierung` festlegen.
- ✓ Jede einzelne `Formatierungsanweisung` in der Zeichenkette `Formatierung`, die mit dem % eingeleitet wird, wird von links nach rechts mit den einzelnen Parametern verarbeitet und dann an die definierte Stelle in die Zeichenkette eingesetzt.
- ✓ Innerhalb der `Formatierung` können auch einzelne Zeichen eingesetzt werden, die bei der Ausgabe mit auf dem Bildschirm erscheinen sollen, z. B. Leerzeichen, Buchstaben bzw. Zahlen oder Sonderzeichen wie beispielsweise ***.

Formatierungen angeben

Typangabe

Hiermit legen Sie fest, welchen Datentyp das auszugebende Argument haben soll. Folgende Typangaben kommen häufig vor (Auswahl):

Option	Erläuterung
b	Die Zeichenkette wird als Ganzzahl angesehen und soll in binärer (<i>binary</i>) Schreibweise ausgegeben werden.
c	Eine Ganzzahl soll als ASCII-Code dargestellt werden, z. B. 65 ergibt A.
d	Eine Ganzzahl erhält führende Füllzeichen, z. B. Nullen: 3:14 Uhr ergibt 03:14 Uhr.
f	Der Parameter soll als Fließkommazahl dargestellt werden.
s	Das Argument wird als Zeichenkette (<i>string</i>) angesehen und als solche ausgegeben.
x	Die Ganzzahl wird in hexadezimaler Form dargestellt. Hexadezimale Ziffern werden als Zahlen von 0 bis 9 und als Kleinbuchstaben a, b, c, d, e, f dargestellt.

Nachkommastellen

Diese Angabe ist nur auf Werte des Zahlentyps float anwendbar und legt fest, wie viele Stellen nach dem Komma ausgegeben werden sollen.

Beispiel:

`printf("%05.4f", 1.54321)` legt fest, dass 4 Stellen nach dem Komma ausgegeben werden: 1. 5432

Füllzeichen

Um die Zeichenkette nach rechts aufzufüllen, können Sie optional Füllzeichen angeben. Wenn Sie eine bestimmte Länge der auszugebenden Daten wünschen, diese aber nicht erreicht wird, können Sie den Rest mit dem Füllzeichen auffüllen. Standardmäßig ist dies bei Zeichenketten ein Leerzeichen oder bei Zahlen eine 0 (Null). Möchten Sie ein anderes Füllzeichen nutzen, geben Sie dies an, indem Sie ein einfaches Hochkomma `\` voranstellen.

Beispiel:

<code>printf("%05.2f", 3.6)</code>	füllt mit dem Zeichen 0 auf: 3.6 wird zu 03.60.
<code>printf("%'x5.2f", 3.6)</code>	füllt mit dem Zeichen x auf: 3.6 wird zu x3.60.

0	6	.	0	0
5	4	3	2	1

①

Mit `%` leiten Sie die Formatierungsoption ein. `05` bedeutet für den PHP Interpreter, dass insgesamt 5 Stellen von rechts ① mit dem Füllzeichen 0 aufgefüllt werden sollen. Das Zeichen `\` besagt, dass die nachfolgende Option `2f` nach der „Kommastelle“ angewendet werden soll. Hierbei legt die Zahl `2` fest, dass nach dem Komma zwei Zeichen angezeigt werden sollen. Ist nach dem Komma keines oder nur ein Zeichen zu erwarten, werden die freien Stellen mit dem Füllzeichen aufgefüllt. Bei Zahlen ist es standardmäßig das Zeichen 0. Die Typangabe `f` legt den Datentyp float fest.

Zusammengefasst bedeutet die Angabe `%05.2f` für den PHP-Interpreter: Fülle bei der Fließkommazahl vor dem Komma maximal 2 Stellen mit dem Zeichen 0 auf, falls Werte mit null oder einer Stelle vor dem Komma vorhanden sind. Längere Werte vor dem Komma werden nicht gekürzt. Erweitere zusätzlich die Nachkommastellen auf zwei Stellen. Bei einer Fließkommazahl werden die Stellen nach dem Komma mit dem Standardzeichen 0 aufgefüllt. Besitzt der Nachkommawert mehr als zwei Stellen nach dem Komma, wird mathematisch auf zwei Stellen gerundet.

Anzahl der Zeichen

Die Angabe bestimmt, wie lang die auszugebende Zeichenkette sein soll. Wird die Länge nicht erreicht, werden die fehlenden Zeichen mit dem angegebenen Füllzeichen ersetzt. Bei der Berechnung der Anzahl der Zeichen wird bei Fließkommazahlen auch der Dezimalpunkt als Zeichen gezählt.

Beispiel:

```
printf("%'x6.s", "a")
```

setzt die Länge der Zeichenkette auf 6 Zeichen. Enthält die Zeichenkette weniger Zeichen, wird mit dem festgelegten Zeichen **x** aufgefüllt: Die Zeichenkette "a" wird zu "**xxxxxa**".

Ausrichtung

Über die Ausrichtung können Sie bestimmen, ob die Ausgabe des ursprünglichen Strings (des Arguments) rechts oder links im Ergebnisstring ausgerichtet sein soll. Standardmäßig ist die Ausrichtung rechtsbündig, d. h., das Füllzeichen wird links aufgefüllt, also vor der eigentlichen Ausgabe. Möchten Sie das Füllzeichen hinter das Ursprungszeichen setzen, müssen Sie den „Schalter“ **-** angeben. Dieser wird direkt hinter dem **%** notiert.

Beispiel:

```
printf("%-'x6.s", "a")
```

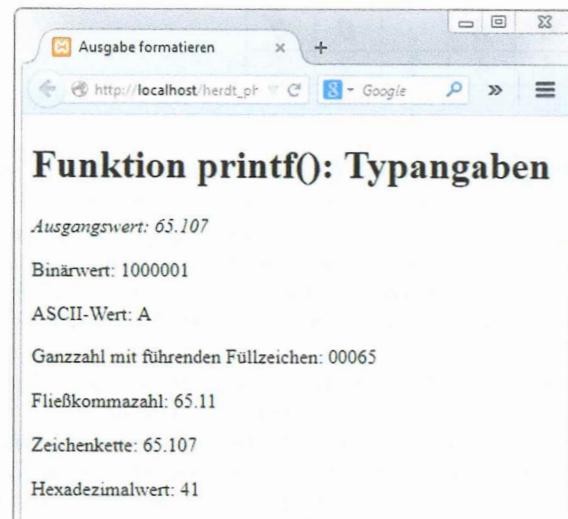
setzt die Länge der Zeichenkette auf 6 Zeichen. Enthält die Zeichenkette weniger Zeichen, wird mit dem festgelegten Zeichen **x** nach der Zeichenkette aufgefüllt: Die Zeichenkette "a" wird zu "**axxxxx**".

Beispiel: printf-1.php

Es werden verschiedene Typangaben anhand einer definierten Variablen demonstriert:

```
<?php
① $zahl = 65.107;
echo "<p><em>Ausgangswert: $zahl</em></p>";
② printf("<p>Binärwert: %b </p>", $zahl);
printf("<p>ASCII-Wert: %c </p>", $zahl);
printf("<p>Ganzzahl mit führenden Füllzeichen: %05d </p>", $zahl);
printf("<p>Fließkommazahl: %05.2f </p>", $zahl);
printf("<p>Zeichenkette: %s </p>", $zahl);
printf("<p>Hexadezimalwert: %x </p>", $zahl);
?>
```

- ① Der Variablen `$zahl` wird eine Fließkommazahl zugewiesen.
- ② Mit `printf()` werden verschiedene Typangaben der Funktion eingesetzt.
- ③ Möchten Sie die Variable `$zahl` als eine Fließkommazahl mit führenden Nullen (sofern der Wert der Variablen weniger Stellen umfasst) und zwei Nachkommastellen ausgeben (z. B. 65.11), so geben Sie die Formatierungsoption `%05.2f` an. Nachkommastellen werden mathematisch gerundet.



Ausgabe der Beispieldatei "printf-1.php"

Beispiel: printf-2.php

Im folgenden Beispiel werden mithilfe von `printf()` formatierte Zeichenketten ausgegeben.

```
<h1>Funktion printf(): Zeichenketten formatieren</h1>
<?php
① $zkette1 = "Beachten Sie unsere Sonderangebote!";
① $zkette2 = "";
① $zkette3 = "Nur noch bis Ende Juli!";
echo "<p>\$zkette1 = " . $zkette1 . "<br>";
echo "\$zkette2 = " . $zkette2 . "<br>";
echo "\$zkette3 = " . $zkette3 . "</p>";

echo "<p>Standardausgabe mit dem <em>echo</em>-Befehl:<br>";
echo "<strong> $zkette1 $zkette2 $zkette3 </strong></p>";
echo "<p>Zeichenketten mit '~' verbinden und 2. Zeichenkette füllen:<br>";
printf("<strong>%s~%'x6s~%s</strong></p>", $zkette1, $zkette2, $zkette3);
?>
```

- ① Drei Variablen werden Zeichenketten zugewiesen, wobei die Variable `$zkette2` eine leere Zeichenkette enthält.
- ② Die Formatierungen der drei Variablen werden über die Angabe von `%s~%'x6s~%s` in der Funktion `printf()` realisiert. Die einzelnen Variablen werden dabei über das entsprechende Zeichen `%` angesprochen. Die erste Formatierungsanweisung `%s` weist den Interpreter an, die Variable `$zkette1` als Variable vom Datentyp `string` zu betrachten. `%'x6s` setzt die Länge der Variablen `$zkette2` auf sechs Zeichen. Enthält die Variable weniger Zeichen, dann sollen sie mit dem festgelegten Zeichen `x` aufgefüllt werden. Die letzte Anweisung `%s` ist analog zur ersten Anweisung. Zwischen den einzelnen Formatierungsanweisungen wurde das Zeichen `~` gesetzt, um bei der Ausgabe die miteinander verbundenen Variablen zu kennzeichnen.



Ausgabe der Beispieldatei "printf-2.php"

Zahlen mit der Funktion `number_format()` formatieren

Geht es bei der formatierten Ausgabe nur um Zahlen, verwenden Sie die Funktion `number_format()`. Geeignet ist die Funktion vor allem dann, wenn Sie

- ✓ die Zahl in ein anderes Zahlenformat als das PHP-interne englische Zahlenformat umsetzen wollen,
- ✓ eine korrekte Rundung der Zahl benötigen oder
- ✓ Zeichen für die Gruppierung von Tausenderstellen und zur Trennung von Nachkommastellen definieren wollen.

Syntax der Funktion `number_format()`

```
number_format(Zahl, Stellen, "Nachkomma", "Tausender");
```

- ✓ Der Parameter `Zahl` gibt die Zahl an, die formatiert werden soll.
- ✓ Mit dem Parameter `Stellen` geben Sie die Anzahl der Nachkommastellen an. Auf die Zahl der Nachkommastellen wird gerundet.
- ✓ `Nachkomma` erwartet die Angabe des Dezimaltrennzeichens für die Nachkommastellen.
- ✓ Der Parameter `Tausender` legt das Tausendertrennzeichen fest.

 Die Funktion `number_format()` erwartet entweder zwei oder vier Parameter. Falls Sie ein Zeichen als Dezimaltrennzeichen angeben, müssen Sie auch immer ein Zeichen für das Tausendertrennzeichen hinterlegen.

Beispiel: `number_format.php`

```
<h1>Zahlen formatieren mit number_format()</h1>
<?php
    $zahl = 23456789.7583;
    echo "<p><em>Vorher: </em>$zahl</p>";
    echo "<p><em>Nachher: </em>" . number_format($zahl, 2, ',', '.') . "</p>"; // Ausgabe: 23.456.789,76
?>
```



Ausgabe der Beispieldatei "number_format.php"

10.2 Nach Zeichenketten suchen

Nach einer Zeichenkette (String) suchen

Mit den Funktionen `strstr()` und `stristr()` können Sie nach dem ersten Vorkommen eines Strings in einer Zeichenkette suchen. Zurückgeliefert werden die Zeichen ab Vorkommen des Strings bis zum Ende des Suchstrings.

Syntax und Bedeutung der Funktion `strstr()`

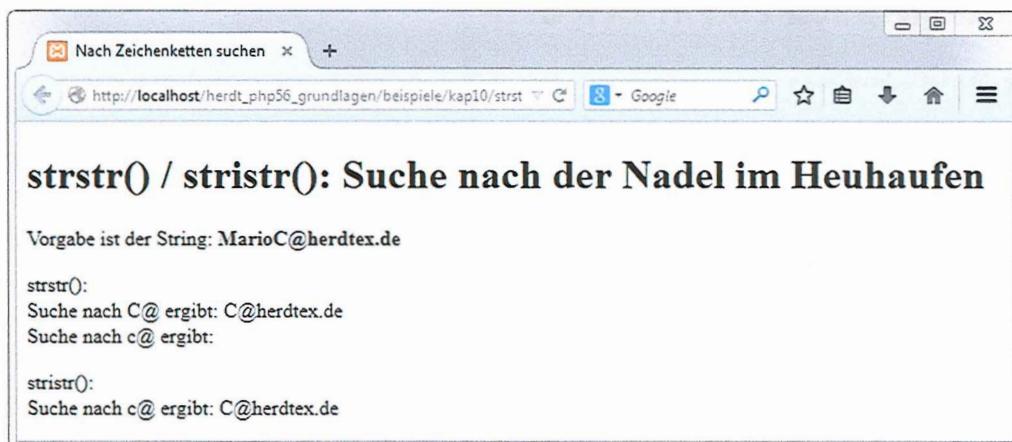
- ✓ Der erste Parameter `String` gibt die Zeichenkette an, die durchsucht werden soll. `strstr (String, Teilstring [, true/false]);`
- ✓ Die Zeichenkette, nach der gesucht werden soll, wird im zweiten Parameter `Teilstring` übergeben. `stristr(String, Teilstring [, true/false]);`

- ✓ Die Funktion liefert als Rückgabewert `FALSE`, falls der Teilstring im String nicht gefunden wird. Ist die Suche erfolgreich, liefert die Funktion als Rückgabewert standardmäßig eine Zeichenkette vom Vorkommen des Teilstrings **bis zum Ende** des Strings, in dem gesucht wird.
- ✓ Wird der dritte, optionale Parameter `TRUE` angegeben, wird die Zeichenkette **vor** dem ersten Auftreten des gesuchten Teilstrings zurückgegeben.
- ✓ Die Funktion `strrstr()` berücksichtigt die Groß- und Kleinschreibung der Zeichen.
- ✓ Die Funktion `stristr()` sucht entsprechend der Funktion `strrstr()`, ignoriert jedoch die Groß- und Kleinschreibung der Zeichenketten.

Beispiel: `strrstr.php`

```
<h1>strrstr() / stristr(): Suche nach der Nadel im Heuhaufen</h1>
<?php
① $string = "MarioC@herdtex.de";
② $teil_gross = "C@";
$teil_klein = "c@";
echo "<p>Vorgabe ist der String: <strong>$string</strong></p>";
echo "<p>strrstr():<br>Suche nach $teil_gross ergibt: " .
      strrstr($string, $teil_gross);
echo "<br>Suche nach $teil_klein ergibt: " .
      strrstr($string, $teil_klein) . "</p>";
echo "<p>stristr():<br>Suche nach $teil_klein ergibt: " .
      stristr($string, $teil_klein) . "</p>";
?>
```

- ① Die Variable `$string` erhält die Zeichenkette, die durchsucht werden soll (= "Heuhaufen").
- ② Die zu suchenden Zeichen werden in der Variablen `$teil_gross` und `$teil_klein` angegeben (= "Nadel").
- ③ Mit der Funktion `strrstr()` werden die Zeichen der Variablen `$teil_gross` innerhalb der Variablen `$string` gesucht. Zurückgegeben werden die Zeichen ab dem gefundenen Teilstring, also `C@herdtex.de`. Der Funktion `strrstr()` können Sie auch die zwei Zeichenketten direkt übergeben: `strrstr("MarioC@herdtex.de", "C@")`;
- ④ Die gleichen Zeichen werden in Kleinbuchstaben gesucht. Es erfolgt keine Ausgabe, da die gesuchte Zeichenfolge nicht gefunden wird, weil `strrstr()` (ohne das *i*) die Groß- und Kleinschreibung berücksichtigt.
- ⑤ Mithilfe der Funktion `stristr()` können Sie unabhängig von der Groß- und Kleinschreibung nach einer Zeichenkette suchen. Analog zu ④ wird mit der Funktion `stristr()` die Zeichenkette durchsucht, in diesem Fall auch gefunden und das Ergebnis angezeigt.



Anzeige der Beispieldatei "strrstr.php"

Nach einem Zeichen suchen

Syntax und Bedeutung der `strrchr()`-Anweisung

Die `strrchr()`-Anweisung findet das letzte Vorkommen eines Zeichens in einer Zeichenkette und liefert die restliche Zeichenkette ab dem gefundenen Zeichen bis zum Ende des Strings zurück.

- ✓ Im ersten Parameter `String` wird die Zeichenkette übergeben, die durchsucht werden soll. Der zweite Parameter `Zeichen` enthält das zu suchende Zeichen.
- ✓ Die Funktion `strrchr()` beginnt, den String von hinten zu durchsuchen, und findet somit das letzte Vorkommen des gesuchten Zeichens.
- ✓ Die Funktion `strrchr()` liefert als Rückgabewert einen String, falls das Zeichen gefunden wird, sonst FALSE.
- ✓ Beachten Sie: Auch wenn nach nur **einem** Zeichen gesucht wird, ist der Rückgabewert bei Sucherfolg meistens eine Zeichenkette.
- ✓ Groß- und Kleinschreibung ist zu beachten.

```
strrchr(String, Zeichen);
```

 Das `r` zwischen `str` und `chr` in `strrchr()` steht für *reverse*. Die Suche erfolgt rückwärts, beginnt somit am Ende des Strings. Die Funktion `strchr()` – ohne `r` zwischen `str` und `chr` – durchsucht den String von vorne. Das erste Vorkommen eines Zeichens wird ermittelt. Mit einiger Erfahrung können Sie aus der Funktionsbezeichnung die Funktionalität der jeweiligen Funktion ableiten.

Beispiel: `strrchr.php`

```
<h1>strrchr(): Suche nach <em>einem</em> Zeichen</h1>
<?php
    $string = "MarioC@herdtex.de";
    echo "<p>Vorgabe ist der String: <strong>$string</strong></p>";
    $teil_1 = "r";
    $teil_2 = "R";
    echo "<p>strrchr():<br>Suche nach $teil_1 ergibt: " . strrchr($string, $teil_1);
    echo "<br>Suche nach $teil_2 ergibt: " . strrchr($string, $teil_2) . "</p>";
?>
```



Anzeige der Beispieldatei "strrchr.php"

10.3 Position und Teil einer Zeichenkette ermitteln

Position eines Zeichens bestimmen

Um die erste bzw. letzte Position eines Zeichens in einer Zeichenkette zu bestimmen, verwenden Sie die Funktionen `strpos()` und `strrpos()`.

Syntax und Bedeutung der Funktion `strpos()`

- ✓ Innerhalb der Klammern werden der String sowie das zu suchende Zeichen angegeben.
- ✓ Optional können Sie den Parameter `Start` angeben. Hiermit beginnt die Suche nach dem Zeichen nicht am Anfang des Strings, sondern erst an der Stelle `Start`.
- ✓ Die Funktion `strpos()` gibt die erste Stelle an, an der das gesuchte Zeichen gefunden wurde. Liefert die Funktion den Wert `0` zurück, handelt es sich um das erste Zeichen im String. Wird der Wert `FALSE` zurückgeliefert, befindet sich das Zeichen nicht im angegebenen String.
- ✓ Im Unterschied zur Funktion `strpos()` liefert die Funktion `strrpos()` die letzte Position des Zeichens zurück, da mit der Suche von rechts begonnen wird.

```
strpos(String, Zeichen, [Start]);
strrpos(String, Zeichen, [Start]);
```

Teilstring einer Zeichenkette bestimmen

Mit der Funktion `substr()` können Sie einen Teil einer Zeichenkette ermitteln.

Syntax und Bedeutung der `substr()`-Anweisung

- ✓ Geben Sie im Parameter `String` die auszulesende Zeichenkette und im Parameter `Start` die Startposition an, an der das Auslesen der Zeichen (von links) beginnen soll. Wenn Sie einen negativen Wert für den Parameter `Start` angeben, dann beginnt das Auslesen der Zeichen von rechts, also vom Ende der Zeichenkette.
- ✓ Optional können Sie den Parameter `Länge` angeben. Geben Sie die Länge nicht an, werden die Zeichen vom Startpunkt bis zum Ende des Strings zurückgeliefert.

```
substr(String, Start [, Länge]);
```

Beispiel: `strpos.php`

Aus einer bestehenden E-Mail-Adresse soll die entsprechende Web-Adresse abgeleitet werden. Dazu lassen Sie in einer E-Mail-Adresse nach dem Zeichen `@` suchen und übergeben die nachfolgenden Zeichen an eine neue Variable. Dieser Zeichenkette werden die Zeichen `http://www.` vorangestellt, um die Web-Adresse zu generieren.

```

<h1>strpos() und substr()</h1>
<?php
① $string = "webmaster@php.net";
$zeichen = "@";
② echo 'E-Mail-Adresse: ' . $string;
③ $pos = strpos($string, $zeichen);
④ if ($pos === FALSE) {
    echo "<p>Zeichen <strong>$zeichen</strong> konnte nicht gefunden
werden!</p>";
} else {
    echo "<p>Wert von \$pos: $pos (Position des $zeichen-Zeichen)</p>";
    $webadresse = substr($string, $pos + 1);
    $webadresse = "http://www." . $webadresse;
    echo "<p>Web-Adresse: <a href=\"$webadresse\">$webadresse</a></p>";
}
?>
```

- ① Der Variablen \$string wird die E-Mail-Adresse webmaster@php.net übergeben. Das Zeichen @, nach dem gesucht werden soll, wird der Variablen \$zeichen zugewiesen.
- ② Die E-Mail-Adresse wird zur Kontrolle am Bildschirm ausgegeben.
- ③ Über die Funktion strpos() wird die Position gesucht, an der sich das Zeichen @ befindet. Dieser Wert wird in der Variablen \$pos gespeichert.
- ④ In der if-Anweisung wird abgefragt, ob das Zeichen @ gefunden wurde. Durch den Vergleichsoperator == wird geprüft, ob der Wert der Variablen \$pos gleich FALSE ist und beide identische Datentypen aufweisen. Dies ist zur Abgrenzung vom Wert 0 (Fund an der ersten Stelle im String) nötig, da 0 und FALSE zwar als gleiche Werte verstanden werden, jedoch aufgrund ihrer unterschiedlichen Datentypen nicht identisch sind. Ein Vergleich von 0 und FALSE mit == wäre richtig, obwohl die Funktion keine Position ermitteln konnte.
- ⑤ Der else-Zweig wird ausgeführt, falls das Zeichen @ gefunden wurde.
- ⑥ Mit der Funktion substr() wird die Zeichenfolge nach dem Zeichen @ in der Variablen \$webadresse abgelegt. Somit haben Sie den Namen der Domain aus der E-Mail-Adresse extrahiert.
- ⑦ Der Domain wird die Zeichenkette http://www. vorangestellt.
- ⑧ Der HTML-Code zum Anzeigen eines Hyperlinks wird erstellt und über den Befehl echo am Bildschirm ausgegeben.



10.4 Zählen innerhalb von Zeichenketten

Länge von Zeichenketten bestimmen

Über strlen() bestimmen Sie die Länge der angegebenen Zeichenkette, z. B. zur Prüfung, ob eine Mindestlänge für ein Eingabefeld in ein Formular eingehalten wurde.

Syntax und Bedeutung der strlen() -Anweisung

- ✓ Innerhalb der Klammern wird der String angegeben, dessen Zeichenlänge ermittelt werden soll.
- ✓ Als Ergebnis erhalten Sie die Anzahl der Zeichen bzw. 0, wenn die Zeichenkette leer ist.

Anzahl des Vorkommens bestimmen

Mit der Funktion `substr_count()` ermitteln Sie, wie oft ein Teilstring in einer Zeichenkette (String) vorkommt.

Syntax und Bedeutung der `substr_count()`-Anweisung

- ✓ Im Parameter `String` wird die Zeichenkette übergeben, innerhalb derer nach der Häufigkeit des `Teilstrings` durchsucht werden soll.
- ✓ Im Parameter `Teilstring` wird die Zeichenkette angegeben, von der Sie das Vorkommen im `String` zählen möchten.
- ✓ Als Rückgabewert erhalten Sie die Anzahl, wie oft der `Teilstring` im `String` gefunden wurde.
- ✓ Sich überlappende `Teilstrings` werden nicht mitgezählt, so wird z. B. der `Teilstring` `123123` nur einmal in der Zeichenkette `123123123` gezählt.

`substr_count(String, Teilstring);`

Beispiel: `substr_count.php`

In diesem Beispiel soll kontrolliert werden, ob eine Zeichenkette mindestens einmal das Zeichen `@` und mindestens einmal den Punkt `.` enthält.

```
<h1>substr_count(): Gültigkeit einer Mailadresse</h1>
<?php
①   $string = "webmaster@herdtx.de";
②   $okay = 1;
    echo "E-Mail-Adresse: " . $string;
③   if (substr_count($string, "@") == 0) {
        echo "<p>Zeichen <strong>@</strong> konnte nicht gefunden werden!</p>";
        $okay = 0;
    }
④   if (substr_count($string, ".") == 0) {
        echo "<p>Zeichen <strong>.</strong> konnte nicht gefunden werden!</p>";
        $okay = 0;
    }
⑤   if ($okay == 1) {
        echo "<p>Die E-Mail-Adresse scheint gültig zu sein.</p>";
    }
?>
```

- ① Die Variable `$string` beinhaltet die zu untersuchende E-Mail-Adresse. Sie können sie zum Testen der Abfragen beliebig ändern.
- ② Die Variable `$okay` wird angelegt, um das Testergebnis zu speichern. Zu Beginn wird sie standardmäßig auf den Wert `1` gesetzt, wobei `1` für gültig steht und `0` für ungültig.
- ③ Im `if`-Zweig wird über die Funktion `substr_count()` überprüft, ob sich im Wert der Variablen `$string` das Zeichen `@` befindet. Ist der Rückgabewert `0`, wurde das Zeichen also nicht gefunden, so wird eine entsprechende Meldung ausgegeben. Der Wert der Variable `$okay` wird auf `0` gesetzt.
- ④ In der zweiten `if`-Anweisung wird nach dem Zeichen `.` gesucht. Wird dieses nicht gefunden, wird analog zu dem vorhergehenden Zweig eine Meldung ausgegeben und der Wert der Variable `$okay` auf `0` gesetzt.
- ⑤ Die letzte Bedingung prüft, ob die Variable `$okay` den Wert `1` hat und somit die E-Mail-Adresse die Zeichen `@` und `.` enthält. Die dafür vorgesehene Meldung wird im Browser ausgegeben.



Anzeige Beispieldatei "substr_count.php"



Diese Überprüfung einer E-Mail-Adresse dient lediglich zur Verdeutlichung der Funktion `substr_count()`. Die Kriterien für eine gültige E-Mail-Adresse sind weitaus umfangreicher. Für den Einsatz in der Praxis ist dies kein ausreichender Ansatz.

10.5 Zeichenketten vergleichen

Zeichenketten miteinander vergleichen

Mithilfe der Funktionen `strcmp()` bzw. `strcasecmp()` können Sie zwei Zeichenketten miteinander vergleichen.

Syntax und Bedeutung der `strcmp()`-Anweisung

- ✓ Die Funktion vergleicht die einzelnen Zeichen von `String1` und `String2` anhand ihres ASCII-Codes. Beispielsweise ist das Zeichen **A** (ASCII-Code: 65) kleiner als das Zeichen **B** (ASCII-Code: 66).
- ✓ Die zwei Zeichenketten, die binär miteinander verglichen werden sollen, werden innerhalb der Klammern angegeben.
- ✓ Bei dem Vergleich wird nach Groß- und Kleinschreibung unterschieden.
- ✓ Als Rückgabe erhalten Sie einen Wert < 0 , wenn `String1 < String2` ist, oder einen Wert > 0 , wenn `String1 > String2` ist, oder den Wert 0 , wenn beide Zeichenketten gleich sind.
- ✓ Die Funktion `strcasecmp()` lässt im Unterschied zur Funktion `strcmp()` die Groß- und Kleinschreibung außer Acht.

```
strcmp(String1, String2);
strcasecmp(String1, String2);
```

10.6 Zeichenketten modifizieren

Mit folgenden Funktionen können Sie Zeichenketten unterschiedlich umwandeln, wie z. B. eine Zeichenkette wiederholen, Leerzeichen aus einer Zeichenkette entfernen oder Zeichenketten in Groß- bzw. Kleinschreibung umändern.

Zeichenketten wiederholen

Syntax und Bedeutung der `str_repeat()`-Anweisung

```
str_repeat(String, Anzahl);
```

- ✓ Der angegebene String wird mehrmals aneinandergesetzt.
- ✓ Wie oft die Zeichenkette wiederholt werden soll, geben Sie über die `Anzahl` an.
Beispiel: Eine mehrfache Ausgabe am Bildschirm erreichen Sie beispielsweise über die Angabe von
`echo str_repeat('-', 80);`

Leerraum oder andere Zeichen entfernen

Syntax und Bedeutung der Anweisungen

```
rtrim(String[, Zeichenliste]);
ltrim(String[, Zeichenliste]);
trim (String[, Zeichenliste]);
```

- ✓ Alle Funktionen entfernen beliebige Zeichen, die als optionaler Parameter Zeichenliste angegeben werden müssen. Wird der Parameter nicht angegeben, werden automatisch die sogenannten Leerräume entfernt. Das sind u. a. das Leerzeichen, die Zeichen "\n" für Zeilenumbruch oder "\t" für Tabulator.
- ✓ Die Funktion `rtrim()` löscht angegebene Zeichen oder Leerräume am Ende (`r` vor dem `trim` steht für `right`) eines Strings.
- ✓ `ltrim()` hat dieselbe Funktion, entfernt diese Zeichen aber am Anfang einer Zeichenkette (das `/` steht hier für `left`).
- ✓ Die Funktion `trim()` entfernt angegebene Zeichen bzw. die Leerräume am Anfang **und** Ende einer Zeichenkette.

Buchstaben innerhalb einer Zeichenkette umwandeln

Syntax und Bedeutung der Anweisungen

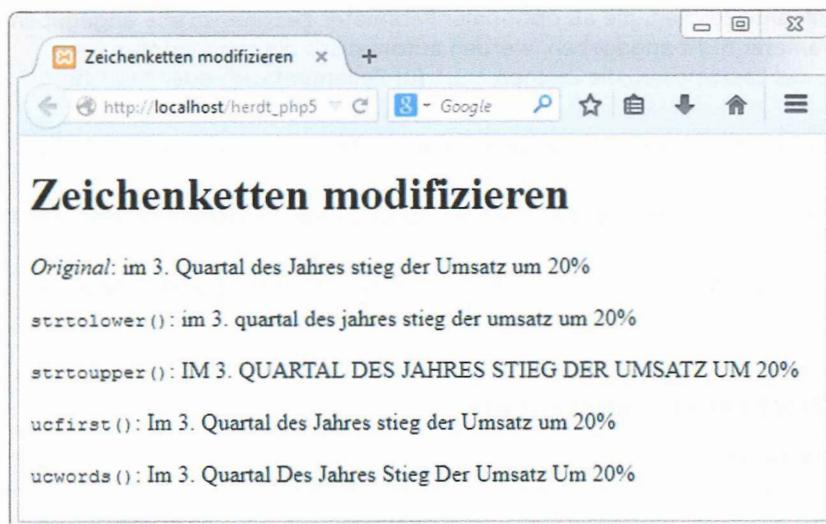
```
strtolower(String);
strtoupper(String);
ucfirst(String);
ucwords(String);
```

- ✓ Die Funktion `strtolower()` wandelt alle Zeichen der Zeichenkette in Kleinbuchstaben um.
- ✓ Die Funktion `strtoupper()` wandelt alle Zeichen der Zeichenkette in Großbuchstaben um.
- ✓ Mit der Funktion `ucfirst()` setzen Sie den ersten Buchstaben der Zeichenkette in einen Großbuchstaben um, sofern es sich hierbei um ein Zeichen des Alphabets handelt (`uc` am Anfang des Funktionsnamens steht hier für `UpperCase`).
- ✓ Die Funktion `ucwords()` setzt den ersten Buchstaben eines jeden Wortes in einen Großbuchstaben um. Auch hierbei muss es sich jeweils um ein alphabetisches Zeichen handeln.

Beispiel: str_umwandeln.php

```
<h1>Zeichenketten modifizieren</h1>
<?php
① $text = "im 3. Quartal des Jahres stieg der Umsatz um 20%";
    echo "<p><em>Original</em>: $text</p>";
② $kl = strtolower($text);
    echo "<p><code>strtolower()</code>: $kl</p>";
③ $gr = strtoupper($text);
    echo "<p><code>strtoupper()</code>: $gr</p>";
④ $uf = ucfirst($text);
    echo "<p><code>ucfirst()</code>: $uf</p>";
⑤ $uw = ucwords($text);
    echo "<p><code>ucwords()</code>: $uw</p>";
?
?>
```

- ① Der Variablen `$text` wird eine Zeichenkette zugewiesen, die über die verschiedenen Funktionen modifiziert werden soll.
- ② Der Beispieltext wird mit der Funktion `strtolower()` in Kleinbuchstaben umgewandelt.
- ③ Der Beispieltext wird mit der Funktion `strtoupper()` in Großbuchstaben umgewandelt.
- ④ Das erste Zeichen des übergebenen Wertes, also hier des ganzen Satzes, und damit nur das des ersten Wortes des Beispieltextes, wird mit der Funktion `ucfirst()` in einen Großbuchstaben umgewandelt.
- ⑤ Das erste Zeichen jedes Wortes des Beispieltextes wird mit der Funktion `ucwords()` in einen Großbuchstaben umgewandelt.



Ansicht der Beispieldatei "str_umwandeln.php"

Zeichen oder Zeichenfolgen innerhalb einer Zeichenkette austauschen

Manchmal ist es notwendig, spezielle Zeichen oder Zeichenfolgen auszutauschen, z. B. bei unerwünschten Sonderzeichen oder um Zeichenfolgen eine spezielle Formatierung zuzuweisen.

Syntax und Bedeutung der Funktion `strtr()`

```
strtr(String, Zeichen_von, Zeichen_nach);
```

- ✓ Die `strtr()`-Funktion bearbeitet den String, indem Zeichen aus `Zeichen_von` in die entsprechenden Zeichen aus `Zeichen_nach` umgesetzt werden. Als Ergebnis erhalten Sie die umgewandelte Zeichenkette.
- ✓ Die Zeichen, die ausgetauscht werden sollen, geben Sie bei der `strtr()`-Funktion in Anführungszeichen hintereinander an. Die entsprechende Anzahl von Zeichen sollte sich in der Angabe von `Zeichen_nach` befinden.
- ✓ Sind `Zeichen_von` und `Zeichen_nach` von unterschiedlicher Länge, so werden die überzähligen Zeichen ignoriert.
- ✓ Sonderfall: `strtr()` kann auch mit nur zwei Argumenten aufgerufen werden. In diesem Fall verhält sich die Funktion anders: Das zweite Argument muss ein assoziatives Array sein, das Zeichenkettenpaare enthält. Die Schlüssel der Array-Elemente stehen für die `Zeichen_von`, während die Werte der Array-Elemente als `Zeichen_nach` verwendet werden.

Syntax und Bedeutung der Funktion `str_replace()`

```
str_replace(String_von, String_nach, String);
```

- ✓ Im Gegensatz zur Funktion `strtr()`, bei der nur einzelne Zeichen ausgetauscht werden, können Sie mit `str_replace()` ganze Wörter oder Textpassagen vertauschen.
- ✓ Sie können jedoch immer nur eine Textpassage angeben, die ausgetauscht werden soll.
- ✓ Hierbei ist zu beachten, dass erst der zu ersetzenende, dann der hinzuzufügende String anzugeben ist. Der zu untersuchende String wird in dieser Funktion als dritter Parameter angegeben.
- ✓ Die Zeichen, die die Fundstellen ersetzen, können Buchstaben oder Zahlen sowie vollständige HTML-Tags beinhalten!



Bei der Angabe des zu ersetzenen Strings wird die Groß- und Kleinschreibung beachtet.

Beispiel: str_tauschen.php

```

<h1>Zeichen(ketten) austauschen</h1>
<?php
    echo "<p><em>strtr() 'Zeichen für Zeichen':</em>" ;
    $string = "-----|-----|-----|-----|";
    echo "<br>Vorher: $string";
    echo "<br>Nachher: " . strtr($string, '-|', 'x-') . "</p>";
    $string = "Wäre Jörg ein großer Sänger, würde er schöne Lieder singen.";
    echo "<p><em>strtr() flexibel mit Angabe eines Feldes:</em>";
    echo "<br>Vorher: $string <br>";
    $feld = array("ä" => "ae", "ö" => "oe", "ü" => "ue", "ß" => "ss");
    echo "Nachher: " . strtr($string, $feld) . "</p>";
    echo "<p><em>str_replace():</em>";
    $string = "Meine Tante lebt in Frankreich...";
    echo "<br>Vorher: $string";
    $string = str_replace("Tante", "<strong>Nichte</strong>", $string);
    echo "<br>Nachher (1): " . $string;
    $string = str_replace("Frankreich", "<em>Italien</em>", $string);
    echo "<br>Nachher (2): " . $string . "</p>";
?

```

- ① Die Variable \$string wird mit Zeichen gefüllt.
- ② Die Funktion strtr() wird aufgerufen, das Zeichen **-** wird durch **X** und das Zeichen **|** wird durch **-** ersetzt. Das Ergebnis wird ausgegeben.
- ③ Ein Array wird definiert, das Zeichenkettenpaare enthält.
- ④ Die Schlüsselwerte werden in einem Beispielsatz gesucht und durch die dazugehörigen Werte des entsprechenden Array-Elements ausgetauscht.
- ⑤ Mit der Funktion str_replace() soll das Wort *Tante* durch das fett gedruckte Wort *Nichte* ersetzt werden.
- ⑥ Die Funktion str_replace() wird ein weiteres Mal aufgerufen. Auch in diesem Fall wird eine Zeichenkettenersetzung dazu verwendet, um den Ländernamen auszutauschen und zusätzlich das HTML-Tag em einzufügen.



Anzeige der Beispieldatei "str_tauschen.php"

10.7 Mit Arrays und Zeichenketten arbeiten

Eine Zeichenkette in ein Array umwandeln

Wenn Sie eine Zeichenkette an einem bestimmten Trennzeichen aufteilen möchten und die Teilstrings, die Sie dadurch erhalten, in einem Array speichern möchten, verwenden Sie die Funktion `explode()`.

Syntax und Bedeutung der Funktion `explode()`

```
explode(Trennzeichen, String [, Limit]);
```

- ✓ Die Funktion `explode()` ermöglicht es, eine Zeichenkette anhand eines Trennzeichens, das Sie selbst bestimmten können, aufzuteilen und als einzelne Elemente in einem indizierten Array zu speichern. So können Sie beliebige Wertelisten mit demselben Trennzeichen, z. B. alle Wörter eines Satzes per Trennung über das Leerzeichen, einfach in einem Array speichern.
- ✓ Mit der optionalen Angabe von `Limit` können Sie die maximale Anzahl der zurückgelieferten String-Teile angeben. Besitzt eine Zeichenkette mehr Trennzeichen, als im `Limit` angegeben wird, enthält das letzte Element des Arrays den Rest der Zeichenkette. Wird der Parameter `Limit` negativ angegeben, werden alle String-Teile zurückgegeben, mit Ausnahme der der Zahl `Limit` entsprechenden Anzahl von rechts gezählt.

Aus einem Array eine Zeichenkette erzeugen

Mit der Funktion `implode()` können Sie die Elemente eines Arrays zu einer Zeichenkette verbinden.

```
implode(Verbindungszeichen, Array);
```

- ✓ Mit dieser Funktion können Sie die einzelnen Elemente eines Arrays in einer Zeichenkette zusammenführen.
- ✓ Zwischen den einzelnen Elementen steht jeweils das Verbindungszeichen, das Sie selbst festlegen.

Beispiel: `ex-implode.php`

In diesem Beispiel werden die Funktionen `explode()` und `implode()` verwendet.

```

<h1> explode() / implode(): Zeichenkette und Array</h1>
<?php
① $vorgabestring = "Elstar; Gala; Jonagold; Boskoop; Delicious";
  echo "<p><strong>Vorgabe-Zeichenkette:</strong> $vorgabestring</p>";
  echo "<p><strong>explode()</strong></p>";
② $ausgabe = explode(";", $vorgabestring);
③ $laenge = count($ausgabe);
  echo "<p>";
④ for ($i = 0; $i < $laenge; $i++) {
    echo "Teilstring $i: $ausgabe[$i]<br>";
  }
  echo "</p>";
  echo "<pre>";
⑤ print_r($ausgabe);
  echo "</pre>";
  echo "<p><strong>implode()</strong></p>";
⑥ $ergebnis = implode(" * ", $ausgabe);
  echo "<p>Die Zeichenkette lautet: <br>$ergebnis</p>";
?
  
```

- ① Der Variablen \$vorgabestring wird eine Zeichenkette mit Begriffen zugewiesen, die durch Semikola voneinander getrennt sind.
- ② Mit der Funktion explode() wird diese Zeichenkette am Semikolon aufgeteilt und in der Variablen \$ausgabe als indiziertes Array gespeichert.
- ③ Mithilfe der Array-Funktion count() wird die Länge des Arrays, also die Anzahl der Array-Elemente, festgestellt.
- ④ Die Elemente des Arrays werden einzeln mit einer for-Schleife am Bildschirm ausgegeben.
- ⑤ Der Inhalt des Arrays \$ausgabe wird zusätzlich mit dem speziellen Befehl print_r() am Bildschirm ausgegeben. Wird print_r() für ein Array aufgerufen, so werden die Indizes und Werte des Arrays angezeigt.
- ⑥ Der Variablen \$ergebnis werden mit der Funktion implode() die einzelnen Array-Elemente des Arrays \$ausgabe als Zeichenkette - getrennt durch die Zeichenkette " * " - zugewiesen und ausgegeben.

Vorgabe-Zeichenkette: Elstar; Gala; Jonagold; Boskoop; Delicious

```
explode()  
  
Teilstring 0: Elstar  
Teilstring 1: Gala  
Teilstring 2: Jonagold  
Teilstring 3: Boskoop  
Teilstring 4: Delicious  
  
Array  
{  
    [0] => Elstar  
    [1] => Gala  
    [2] => Jonagold  
    [3] => Boskoop  
    [4] => Delicious  
}  
  
implode()  
  
Die Zeichenkette lautet:  
Elstar * Gala * Jonagold * Boskoop * Delicious
```

Anzeige der Beispieldatei "ex-implode.php"

10.8 Übung

Mit Zeichenkettenfunktionen arbeiten

Übungsdatei: --

Ergebnisdatei: uebung.php

- ① Nach einer Berechnung erhalten Sie eine Fließkommazahl mit diversen Nachkommastellen, z. B. 78,123456789. Formatieren Sie die Ausgabe, sodass drei Stellen vor dem Komma und fünf Nachkommastellen ausgegeben werden.

- ② Gegeben sind folgende Variablen:

```
$string1 = "Beachten Sie das Angebot für die "
$string2 = "folgende Kalenderwoche: "
$string3 = " "
$string4 = "Bananen, 5 Kilo für nur 5.- Euro!"
```

Geben Sie die Variablen mithilfe der Funktion `printf()` in einer formatierten Zeichenkette aus.

Setzen Sie hierfür die Länge der Variablen `$string3`, die mit dem festgelegten Zeichen * gefüllt wird, auf 5 Zeichen. Trennen Sie die vier Variablen mit der Zeichenkette --- voneinander ab.

- ③ Fügen Sie die einzelnen Variablen in einer neuen Variablen `$string` aneinander. Die Zeichenkette, die in der Variablen `$string` gespeichert wurde, soll mithilfe der Funktion `explode()` am Zeichen " " (Leerstelle) getrennt werden und die einzelnen Elemente des entstehenden Arrays sollen ausgegeben werden. Anschließend verwenden Sie die Funktion `implode()`, um das Array in eine Zeichenkette zu speichern. Verwenden Sie als Trennzeichen das Zeichen # und geben Sie das Ergebnis aus.
- ④ Ersetzen Sie die Zeichenkette *das Angebot* durch die fett gedruckte Zeichenkette **unser Sonderangebot** und speichern Sie das Ergebnis in der Variablen `$string5`. Tauschen Sie die Zeichenkette *Bananen* durch die Zeichenkette *Alle Obstsorten* aus, speichern Sie das Ergebnis in der Variablen `$string6` und geben Sie das Ergebnis im Browser aus.

Suche nach der "Nadel im Heuhaufen"

Übungsdatei: --

Ergebnisdatei: **textsache.php**

- ⑤ Entwerfen Sie eine Datei (*textsache.php*) mit einem einfachen HTML-Formular, in das Sie einen Beispieltext in ein mehrzeiliges Eingabefeld und einen Suchbegriff in ein Eingabefeld eintragen können. Die Auswertung der Suche soll in derselben Datei geschehen und Informationen über die Anzahl der Treffer beinhalten (Funktion `substr_count()`), wenn der Suchbegriff eingegeben wurde. Um eine Fehlermeldung zu vermeiden, prüfen Sie per `isset()`, ob die `$_POST`-Variable für den Suchbegriff vorhanden ist. Zudem soll der durchsuchte Text mit markierten Fundstellen unterhalb des Formulars ausgegeben werden (Funktion `str_replace()`). Orientieren Sie sich mit Ihrer Lösung an der nachfolgenden Abbildung einer Beispillösung.

The screenshot shows a web browser window with the title 'Übung Kapitel 10 - Textsuc...'. The address bar displays 'http://localhost/herdt_php56_grundlagen/uebungen_ergebnisse/kap10/textsache.php'. The main content area has a heading 'Begriff in einer Textpassage suchen'. Below it, there is a large block of text (Lorem ipsum) and a smaller block below it. A text input field labeled 'Originaltext:' contains the first block of text. A search input field labeled 'Suche nach:' contains the word 'amet'. A button labeled 'Zeichenkette suchen' is present. The results section shows the text with occurrences of 'amet' highlighted in yellow. A message 'Suche nach "amet": 6 Mal gefunden.' is displayed above the highlighted text. The bottom part of the results shows the second block of text with its occurrences of 'amet' also highlighted.

Textsuche in einem Beispieltext

11 Datum und Zeit

In diesem Kapitel erfahren Sie

- ✓ wie Sie das aktuelle Datum und die Uhrzeit des Systems auslesen
- ✓ wie Sie die Ausgabe des Datums steuern können
- ✓ wie Sie Datumsdifferenzen berechnen
- ✓ wie Sie ein Datum auf Gültigkeit prüfen

Voraussetzungen

- ✓ Arrays

11.1 Datum und Zeit ermitteln

In vielen PHP-Skripten sind Datums- und Zeitberechnungen notwendig. PHP verwendet dabei das Datum und die Uhrzeit des Servers, auf dem das Programm läuft.

Datum und Zeit auslesen

Informationen zu Datum und Zeit erhalten Sie über die Funktion `getdate()`. Diese Funktion liefert das Ergebnis in einem assoziativen Array zurück.

Syntax der `getdate()`-Anweisung

- ✓ `getdate()` gibt ein assoziatives Array zurück. Über von PHP vordefinierte Schlüssel können Sie auf bestimmte Zeit- bzw. Datumsinformationen zugreifen. Die nachfolgende Tabelle gibt Ihnen einen Überblick, welche Werte im Rückgabe-Array zu finden sind.
- ✓ `getdate()` kann ohne Parameter aufgerufen werden. In dem Fall erhalten Sie die aktuellen Datums- und Zeitwerte des Webservers, also die Informationen zum Zeitpunkt des Aufrufs.
- ✓ `getdate()` kann optional mit einem Parameter `Zeitstempel` aufgerufen werden. Der Parameter `Zeitstempel` erwartet die Anzahl der Sekunden, die seit dem 01.01.1970 vergangen sind. Sämtliche Zeitangaben in PHP beruhen auf diesem Datum, das auch als Beginn der UNIX-Epoche bezeichnet wird. Aus diesem Grund wird der Zeitstempel als **UNIX-Timestamp** oder kurz **Timestamp** bezeichnet. `getdate()` gibt dann ein Array zurück, in dem die Werte bezogen auf den Timestamp enthalten sind.

`getdate([Zeitstempel]);`

Schlüssel	Erklärung	Rückgabewerte
seconds	Sekunde der aktuellen Uhrzeit	0 bis 59
minutes	Minute der aktuellen Uhrzeit	0 bis 59
hours	Stunde der aktuellen Uhrzeit	0 bis 23
mday	Tag des aktuellen Monats	1 bis 31
wday	Numerischer Tag der Woche	0 = Sonntag, 1 = Montag, ... 6 = Samstag
mon	Monat als Zahl	1 = Januar, 2 = Februar, ... 12 = Dezember
year	Jahreszahl	1970 bis 2038

Schlüssel	Erklärung	Rückgabewerte
yday	Numerischer Tag des Jahres	0 bis 365
weekday	Ausgeschriebener Wochentag (in Englisch)	Sunday, Monday, ... , Saturday
month	Ausgeschriebener Monat (in Englisch)	January, February, ... , December
0	Sekunden seit 01.01.1970	0 bis ...

Beispiel: *aktuell.php*

Die Informationen im Rückgabearray von der Funktion `getdate()` zum jetzigen Zeitpunkt (ohne Aufruf eines Zeitstempels) sollen angezeigt werden. Dazu wird jedes Element des assoziativen Arrays angesprochen.

①	<?php \$jetzt = getdate() ;
②	echo "<pre>" ;
③	print_r(\$jetzt) ;
④	echo "</pre>" ;
⑤	echo "<p>Stunde: " . \$jetzt["hours"];
⑥	echo "
Minute: " . \$jetzt["minutes"];
⑦	echo "
Sekunde: " . \$jetzt["seconds"];
⑧	echo "
Tag der Woche: " . \$jetzt["wday"] . " = " . \$jetzt["weekday"];
⑨	echo "
Tag des Monats: " . \$jetzt["mday"];
	echo "
Tag des Jahres: " . \$jetzt["yday"];
	echo "
Monat: " . \$jetzt["mon"] . " = " . \$jetzt["month"];
	echo "
Jahr: " . \$jetzt["year"];
	echo "
Zeitstempel: " . \$jetzt["0"] . "</p>" ;
	?>

- ① Über die Funktion `getdate()` wird die aktuelle Zeit- und Datuminformation des Webservers ausgelesen und in der Variablen `$jetzt` gespeichert. Dabei handelt es sich um ein assoziatives Array.
- ② Mithilfe der Funktion `print_r()` wird die Arrayvariable `$jetzt` im Browser ausgegeben.
- ③ Die Stunden-, Minuten- und Sekundenwerte der aktuellen Zeit werden über die entsprechenden, vordefinierten Schlüssel im Array `$jetzt` angesprochen und ausgegeben.
- ④ Der Wert des Arrayeintrags `$jetzt["wday"]` liefert einen numerischen Wert, welcher den Wochentag repräsentiert. In diesem Fall der Wert 2 für Tuesday (Dienstag). Beachten Sie, dass die Zählung der Wochentage bei 0 = Sonntag beginnt. Hinter der Zahl für den Wochentag wird über den Arrayschlüssel `weekday` der Name des Wochentags ausgegeben. Mit der PHP-Standardinstallation erhalten Sie die englischen Bezeichnungen.
- ⑤ Über den Schlüssel `mday` erhalten Sie den Tag des Monats als Zahlenwert.
- ⑥ Der Wert `$jetzt["yday"]` liefert einen numerischen Wert. Dieser entspricht dem n-ten Tag im Jahr.
- ⑦ Über die Schlüssel `mon` und `month` erhalten Sie den Zahlenwert sowie den englischen Namen des Monats.
- ⑧ Hiermit erhalten Sie das Jahr des aktuellen Datums.
- ⑨ Als letzte Ausgabe wird der aktuelle Zeitstempel in Sekunden seit dem 01.01.1970 ausgegeben. Falls Sie `getdate()` mit einem Zeitstempel aufgerufen haben, finden Sie den Wert hier wieder.

```

Array
(
    [seconds] => 29
    [minutes] => 12
    [hours] => 17
    [mday] => 23
    [wday] => 2
    [mon] => 12
    [year] => 2014
    [yday] => 356
    [weekday] => Tuesday
    [month] => December
    [0] => 1419351149
)

Stunde: 17
Minute: 12
Sekunde: 29
Tag der Woche: 2 = Tuesday
Tag des Monats: 23
Tag des Jahres: 356
Monat: 12 = December
Jahr: 2014
Zeitstempel: 1419351149

```

Daten des 23.12.2014 um 17:12:29 Uhr

11.2 Datum und Zeit formatieren

Mit der `date()`-Funktion können Sie ein Datum für die Ausgabe formatieren.

Syntax und Bedeutung der `date()`-Anweisung

- ✓ Die `date()`-Anweisung erwartet als Parameter Formatanweisungen, welche Datumssegmente angezeigt werden sollen. In der nachfolgenden Tabelle finden Sie eine Übersicht der wichtigsten Angaben und deren Auswirkungen.
- ✓ Der Parameter `Format` wird als Zeichenkette in Anführungszeichen angegeben.
- ✓ `date()` kann mit nur dem ersten Parameter der Formattanweisungen aufgerufen werden. In dem Fall erhalten Sie den formatierten Datumsstring des aktuellen Zeitpunkts zurück.
- ✓ Der optionale zweite Parameter `Zeitstempel` ermöglicht Ihnen, ein spezielles Datum anzugeben. Dieser Zeitstempel entspricht den Sekunden seit dem 01.01.1970.

`date(Format [, Zeitstempel]);`

Formattanweisungen (Auswahl)

Von einer langen Liste möglicher Formattanweisungen, die Sie im Parameter `Format` angeben können, finden Sie die wichtigsten in nachfolgender Tabelle. Über die einzelnen Format-Zeichen steuern Sie, welche Elemente des Datums genutzt und in welchem Format diese formatiert werden sollen.

Eine vollständige Übersicht aller Format-Zeichen finden Sie unter <http://php.net/manual/de/function.date.php>

Format-Zeichen	Resultat
d	Tag des Monats, zweistellig und mit führender Null: 01 bis 31
G	Stunde im 24-Stunden-Format ohne führende Null: 0 bis 23
H	Stunde im 24-Stunden-Format mit führender Null: 00 bis 23
i	Minuten mit führender Null: 00 bis 59
j	Tag des Monats ohne führende Null: 1 bis 31
m	Zahl des Monats mit führender Null: 01 bis 12
n	Zahl des Monats ohne führende Null: 1 bis 12
r	Formatierte Ausgabe nach RFC 822/2822, z. B. Thu, 28 May 2009 23:43:51 +0200
s	Sekunden mit führender Null: 00 bis 59
Y (großes Y)	Jahr als vierstellige Zahl, z. B. 2001
y (kleines y)	Jahr als zweistellige Zahl, z. B. 01



Die Funktion `date()` unterstützt nur Zeiten zwischen dem 13.12.1901 und 19.01.2038 (vor PHP 5.1.0 lag dieser Zeitraum z. B. bei Windows zwischen 01.01.1971 und 19.01.2038). Wird ein Datum vor bzw. nach diesen Eckdaten aufgerufen, gibt der PHP-Interpreter je nach Webserver und Betriebssystem entweder den 01.01.1970 oder ein anderes falsches Datum aus, jedoch ohne eine spezielle Fehlermeldung anzuzeigen.

Beispiel: date.php

In diesem Beispiel sollen die verschiedenen Angaben der Formatanweisungen sichtbar gemacht werden.

```
<?php
① echo date("d.m.y");
② echo "<br>" . date("d.m.Y", time() + 86400);
③ echo "<br>Tag: " . date("d.m.Y") . ", Uhrzeit: " . date("H:i:s");
④ echo "<br>" . date("j.n.y");
⑤ echo "<br>" . date("r");
?>
```

- ① Mit der Formatanweisung `d.m.y` werden Tag, Monat und Jahr jeweils zweistellig formatiert. Über die `echo`-Anweisung geben Sie die formatierte Zeichenkette im Browser aus.
- ② In diesem Funktionsaufruf mit der Formatanweisung `d.m.Y` wird die Jahresangabe vierstellig dargestellt. Als zweiter Parameter wird hier ein Zeitstempel übergeben, welcher im Funktionsaufruf berechnet wird. Über `time()` wird der aktuelle Zeitstempel ermittelt, zu diesem Wert werden 86400 Sekunden hinzuaddiert. 86400 ist das Ergebnis aus $24 * 60 * 60$ (Stunden * Minuten * Sekunden), also der Sekundenanzahl eines Tages. Der übergebene Zeitstempel entspricht also genau einem Tag in der Zukunft. Um den Aufruf nachvollziehbarer zu gestalten, können Sie auch `date("d.m.Y", time() + 24 * 60 * 60 * 60)` im PHP-Code schreiben. Da Punkt- vor Strichrechnung gilt, werden zuerst die Sekunden pro Tag berechnet und dann zum aktuellen Zeitstempel hinzuaddiert.
- ③ Hier werden zusätzlich zu Datum und Zeit weitere Zeichenketten ausgegeben.
- ④ Die Funktion `date()` wird aufgerufen. Die Ausgabe von Tag und Monat erfolgt ohne führende Nullen, die des Jahres erfolgt zweistellig.
- ⑤ Die Anweisung `r` gibt standardmäßig das Datum und die Uhrzeit nach den im Standard RFC 822/2822 festgelegten Regeln aus. Diese Formatierung wird beispielsweise als Angabe von Uhrzeit und Datum im Mailheader, dem nicht sichtbaren Kopfbereich einer E-Mail, verwendet.



Anzeige der Beispieldatei "date.php"

11.3 Datumsangabe an Sprache anpassen

Englische Monatsbezeichnungen manuell übersetzen

Bei der Ermittlung des aktuellen Datums erhalten Sie die englischen Tages- und Monatsbezeichnungen. Auf einer deutschsprachigen Webseite soll das Datum jedoch in deutscher Sprache ausgegeben werden. Das nachfolgende Beispiel wandelt mithilfe eines selbst erstellten Arrays die englischen Bezeichnungen in die entsprechenden deutschen Namen um.

Beispiel: monat_dt.php

Über ein Array sollen beim Auslesen des aktuellen Datums die deutschsprachigen Bezeichnungen der Monate ausgegeben werden.

```

① <?php
    $monat = array(1 => "Januar", "Februar", "März", "April",
                   "Mai", "Juni", "Juli", "August", "September",
                   "Oktober", "November", "Dezember");
②
    $datum = getdate();
    echo "<p><strong>Englischsprachige Bezeichnung des Monats</strong>";
    echo "<br>Heute ist der " . $datum["mday"] . ". ";
    echo $datum["month"] . " " . $datum["year"] . ".</p>";
    echo "<p><strong>Deutschsprachige Bezeichnung des Monats</strong>";
    echo "<br>Heute ist der " . $datum["mday"] . ". ";
    echo $monat[$datum["mon"]] . " " . $datum["year"] . ".</p>";
?
```

- ① Die Namen der Monate werden im Array \$monat definiert, der Index beginnt mit der Zahl 1. Diese Zuordnung wird vorgenommen, damit die Monatsnamen die dazugehörigen Monatszahlen als Index erhalten. Da bei indizierten Arrays immer der nächst höhere Index für das nächste Element verwendet wird, reicht es hier, lediglich den ersten Array-Eintrag mit dem Index 1 zu versehen, die Indexe der Monatszahlen werden automatisch von 1 - 12 vergeben.
- ② In der Array-Variablen \$datum werden die Datumswerte gespeichert.
- ③ Über den Arrayschlüssel month wird der englische Name des Monats ausgegeben.
- ④ Damit ein bestimmtes Element des Arrays \$monat über den Index angesprochen werden kann, benötigen Sie den Monat als Zahl. Die Zahl liefert Ihnen \$datum["mon"]. Es folgt die erneute Ausgabe des aktuellen Datums. Diesmal wird der Wert aus dem von Ihnen angelegten Array \$monat mit dem Wert von \$datum["mon"] ausgelesen und angezeigt.



*Englische und deutsche Monatsbezeichnungen
(Beispieldatei "monat_dt.php")*

11.4 Länder- und Spracheinstellungen ändern

Einstellungen für deutsche Sprache festlegen

PHP unterstützt bei Länder- und sprachspezifischen Informationen eine Reihe verschiedener Sprachen. Dies bedeutet, Sie können beispielsweise die deutsche Bezeichnung von Monatsnamen direkt ausgeben lassen. Dazu müssen Sie über die Funktion setlocale() die entsprechende Sprache einstellen.

Syntax und Bedeutung der setlocale()-Anweisung

```
setlocale(Kategorie, Sprache);
```

- ✓ Der erste Parameter Kategorie legt fest, auf welche Angaben sich die Umstellung der Sprache auswirken soll. Folgende Angaben sind möglich:

LC_ALL	alle Einstellungen
LC_COLLATE	für den Vergleich von Zeichenketten
LC_CTYPE	betrifft Klassifizierungen und Umwandlung von Zeichen
LC_MONETARY	für Währungsfunktionen
LC_NUMERIC	für das Dezimal-Trennzeichen bei Zahlen
LC_TIME	betrifft Zeit- und Datumsformatierungen

- Der zweite Parameter legt die anzuwendende Sprache und Region fest und wird in Hochkommata angegeben. Folgende Werte sind zulässig (Auszug aus einer Liste von mehr als 50 möglichen Werten): Nachfolgende Kurzbezeichner wurden bisher in PHP integriert:

de_DE	Deutschland
de_AT	Österreich
de_CH	Schweiz
fr_FR	Frankreich
en_GB	Großbritannien
en_US	USA

Die Funktion `setlocale()` ist systemabhängig. Sollten die Werte für die anzuwendende Sprache nicht greifen, verwenden Sie alternativ die dreibuchstabigen Sprachcodes nach ISO 639, z. B. deu, fra oder eng. Alternativ kann auch die Schreibweise 'deu_deu' zum Erfolg führen.

Nachfolgend wird die deutsche Sprache für alle Werte eingestellt.



```
setlocale(LC_ALL, "de_DE");
```

Datum und Zeit sprachspezifisch ausgeben

Damit Sie die aktuellen Zeit- und Datumswerte anhand der eingestellten lokalen Informationen nutzen können, müssen Sie die Funktion `strftime()` verwenden.



Ähnlich der Funktion `date()` können Sie über `strftime()` ein Datum für eine Ausgabe formatieren. Hierbei ist jedoch zu beachten, dass sich die Formatanweisungen der beiden Funktionen grundlegend unterscheiden.

Syntax der `strftime`-Anweisung

```
strftime(Format [, Zeitstempel]);
```

- Die Anweisung erwartet als ersten Parameter die Formatanweisungen.
- Ohne die Angabe des zweiten Parameters `Zeitstempel` liefert die Funktion die aktuelle Datums- und Zeitinformation zurück.

Ausgewählte Formatanweisungen

Mit diesen Formatanweisungen bestimmen Sie, welche Elemente des Datums genutzt werden sollen. Alle Formatanweisungen sind in Anführungszeichen " anzugeben.

Format-Zeichen	Resultat
%a	Abgekürzter Tag der Woche
%A	Bezeichnung für den Wochentag
%b	Abgekürzter Monatsname
%B	Name des Monats
%d	Tag des Monats mit führender Null: 01 bis 31
%H	Stunde im 24-Stunden-Format mit führender Null: 00 bis 23
%j	Tag des Jahres von 1 bis 366
%m	Zahl des Monats mit führender Null: 01 bis 12
%M	Minuten mit führender Null: 00 bis 59

Format-Zeichen	Resultat
%S	Sekunden mit führender Null: 00 bis 59
%x	Datumswiedergabe ohne Zeit
%X	Zeitwiedergabe ohne Datum
%Y (großes Y)	Jahr als vierstellige Zahl, z. B. 2010
%y (kleines y)	Jahr als zweistellige Zahl, z. B. 09
%Z	Sommerzeit
%D	Wie %m/%d/%y
%T	Wie %H:%M:%S

Die vollständige Übersicht aller Format-Zeichen finden Sie unter <http://php.net/manual/en/function.strftime.php>.

Beispiel: `strftime.php`

Analog zum Beispiel der `date()`-Funktion werden die Datums- und Zeitangaben mit der `strftime()`-Funktion gezeigt. Sie kann im Gegensatz zur Funktion `date()` die lokalen Einstellungen über `setlocale()` berücksichtigen.

```
<?php
    setlocale(LC_ALL, "deu");
①    echo strftime("%A %d %B %Y %H:%M:%S", time()) . "<br>";
②    echo strftime("%a., %d. %b. %y um %X", time()) . "<br>";
③    echo strftime("%x = %j. Tag des Jahres %Y", time()) . "<br>";
④    echo strftime("%a, %d %b %Y %X", time());
?>
```



Anzeige der Beispieldatei "strftime.php"

11.5 Zeitfunktionen

Aktuelle Zeit mit der Funktion `time()` bestimmen

Die Zeit, die seit dem 01.01.1970 um 00:00:00 Uhr (Greenwich-Zeit; GMT Greenwich Mean Time) vergangen ist, wird auch **UNIX-Timestamp** (oder einfach **Zeitstempel**) genannt und wird in Sekunden berechnet. Dies ist für die weiteren Zeit- und Datumsfunktionen von Bedeutung, da alle weiteren Datums- und Zeitberechnungen auf dieser Angabe beruhen.

Syntax der Funktion `time()`

- ✓ Diese Funktion ist ohne die Angabe eines Parameters zu verwenden.
- ✓ Da diese Funktion den aktuellen Zeitstempel zurückgibt, können Sie auch durch einfache Addition bzw. Subtraktion andere Zeitstempel als den aktuellen erhalten. So erhalten Sie z. B. durch die Berechnung: `time() - 7 * 86400` (ein Tag = 86400 Sekunden) den Zeitstempel für den Zeitpunkt vor genau einer Woche.

`time();`

UNIX-Timestamp eines Datums mit der Funktion `mkttime()` bestimmen

Um den Zeitstempel, also die Sekunden vom 01.01.1970 bis zum angegebenen Datum, zu erhalten, verwenden Sie die `mkttime()`-Funktion.

Syntax und Bedeutung der Funktion `mkttime()`

`mkttime([Stunde [, Minute] [, Sekunde] [, Monat] [, Tag] [, Jahr]]);`

- ✓ Die Funktion erwartet optional die sechs angegebenen Parameter.
- ✓ Einzelne Parameter können Sie von rechts nach links weglassen, also zuerst Jahr, dann Tag, Monat usw. PHP verwendet bei fehlenden Werten den aktuellen Datums- oder Zeitwert des Webservers. Dies bedeutet, dass beim Aufruf von `mkttime()` ohne Parameter dasselbe Ergebnis zurückgeliefert wird wie bei `time()`.
- ✓ Möchten Sie einen Wert nicht angeben, ersetzen Sie ihn durch die Zahl 0. Dies kann sinnvoll sein, wenn Sie nur mit dem Datum arbeiten wollen. In dem Fall setzen Sie die Parameter Stunde, Minute und Sekunde auf den Wert 0.

Beispiel: `date_diff.php`

Anhand eines vorgegebenen Datums wird die Differenz zum aktuellen Datum berechnet.

```

① <?php
    $tag = 15;
    $monat = 1;
    $jahr = 1969;

    ② $start = mkttime(0, 0, 0, $monat, $tag, $jahr);
    ③ $diff = time() - $start;
    ④ echo "<strong>" . (floor($diff / 86400)) . " Tage</strong> liegen zwischen ";
    ⑤ echo "heute (" . date("d.m.Y") . ") und dem " . date("d.m.Y", $start) . ".";
?>
```

- ① Die Variablen werden mit den Angaben eines Datums gefüllt, hier: 15.1.1969.
- ② Aus diesen Angaben werden über die Funktion `mkttime()` die bis dato vergangenen Sekunden berechnet und in der Variablen `$start` gespeichert. Da Stunden-, Minuten- und Sekundenwerte für dieses Beispiel keine Rolle spielen, werden sie jeweils mit dem Wert 0 angegeben.
- ③ Aus dem aktuellen Zeitstempel `time()` und dem Zeitstempel der Variablen `$start` wird die Differenz zum jetzigen Zeitpunkt berechnet. Der Wert liegt in Sekunden vor.
- ④ Um aus den Sekunden einen Wert in Tagen zu berechnen, wird dieser durch 86400 dividiert. Der Wert 86400 entspricht der Anzahl der Sekunden pro Tag ($24 * 60 * 60$). Die zusätzlich verwendete Funktion `floor()` runden eine Fließkommazahl auf die nächst kleinere Ganzzahl ab, z. B. ergibt `floor(139.35)` 139.
- ⑤ Das aktuelle und das von zu Beginn des Beispiels definierte Datum wird über die Funktion `date()` formatiert ausgegeben. Da diese Funktion als optionalen Parameter den Zeitstempel in Sekunden erwartet, wird der bereits gespeicherte Zeitstempel über die Variable `$start` übergeben.



Ausgabe der Differenz zweier Daten (Beispieldatei "date_diff.php")

Genaue Zeitstempel und Zeitspannen berechnen mit der Funktion `microtime()`

Je nach Anwendungsfall kann es nicht ausreichen, Zeitstempel lediglich in Sekunden zur Verfügung zu haben. Für geringere Zeiteinheiten steht in PHP die Funktion `microtime()` zur Verfügung. `microtime()` liefert den aktuellen Zeitstempel in Mikrosekunden zurück. Wegen ihrer größeren Genauigkeit wird die Funktion häufig zur Berechnung von Zeitspannen verwendet, z. B. zur Berechnung der Ausführungszeit von komplexeren Programmen.

Syntax der Funktion `microtime()`

- ✓ Diese Funktion kann ohne die Angabe eines Parameters verwendet werden und liefert dann den aktuellen Zeitstempel in der Form `microtime([Schalter])` *Mikrosekunden Sekunden*.
- ✓ Wird der Parameter `TRUE` angegeben, erfolgt die Ausgabe des Zeitstempels in Mikrosekunden als Gleitkommazahl.

Beispiel: `microtime.php`

Die Funktion wird mit und ohne Parameter aufgerufen. Zusätzlich wird die Ausführungszeit einer Befehlsfolge berechnet.

```

<h1>microtime(): Zeitstempel mit und ohne Parameter</h1>
<?php
①   echo "Funktionsaufruf ohne Parameter: " . microtime() . "<br>";
    echo "Funktionsaufruf mit Parameter: " . microtime(TRUE);
?
<h1>Zeitmessung mit microtime()</h1>
<?php
    echo "<p><em>Berechnung der Quadratzahlen von 1 - 100000</em><br>";
    $start = microtime(TRUE);
③    for ($i = 1; $i <= 100000; $i++) {
④        echo "$i: " . sqrt($i) . "<br>";
    }
⑤    $ende = microtime(TRUE);
⑥    echo "Ausführungszeit: " . ($ende - $start) . " Sekunden.</p>";
?

```

- ① Die Funktion `microtime()` wird mit und ohne Parameter aufgerufen und am Bildschirm ausgegeben.
- ② Der Variablen `$start` wird der aktuelle Zeitstempel vor Ausführen der Schleife ③ in Mikrosekunden zugewiesen.
- ③ In einer Schleife erfolgt die Berechnung und Ausgabe der Quadratwurzeln der Zahlen von 1 bis 100000.
- ④ Der Variablen `$ende` wird der aktuelle Zeitstempel nach Ausführung der Schleife ③ in Mikrosekunden zugewiesen.
- ⑤ Die Variable `$start` wird von der Variablen `$ende` subtrahiert, um die Ausführungszeit der Schleife zu berechnen.

Zeitstempel einer englischen Datumsangabe

Die Funktion `strtotime()` wandelt ein beliebiges Datum, das in englischer Schreibweise angegeben werden muss, in einen Zeitstempel um.

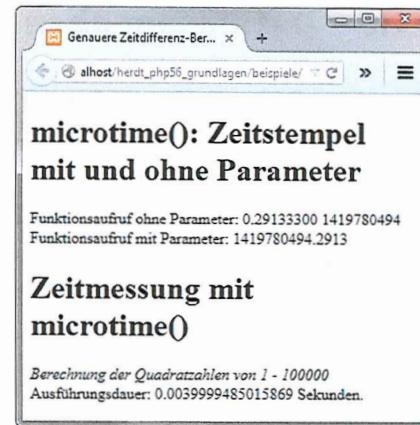
Syntax und Bedeutung der Funktion `strtotime()`

- ✓ Im Parameter Datumsangabe wird eine englische Datumsangabe als Zeichenkette übergeben.

`strtotime(Datumsangabe [, Zeitstempel]);`

Der Zeitstring, der `strtotime()` übergeben wird, kann sowohl eine Zeichenkette im englischen Dateiformat sein, es können aber auch *sprechende Rechenoperationen* angegeben werden. Die Varianten des Zeitstring sind vielfältig. Die nachfolgende Tabelle zeigt einige Beispiele, weitere Beispiele finden Sie unter <http://php.net/manual/de/function.strptime.php>.

<code>strtotime("now");</code>	Gibt die aktuelle Zeit des Servers aus.
<code>strtotime("24 May 2015");</code>	Gibt den Zeitstempel für den 24. Mai 2015 um 00:00:00 Uhr aus.
<code>strtotime("+1 day");</code>	Fügt dem aktuellen Datum einen Tag hinzu.
<code>strtotime("-1 week");</code>	Zieht von der aktuellen Zeit genau eine Woche ab.
<code>strtotime("+1 week 2 days 4 hours 2 minutes");</code>	Rechnet zur aktuellen Zeit eine Woche, zwei Tage, vier Stunden und zwei Minuten hinzu.



Ausgabe der Beispieldatei "microtime.php" (ohne die 100000 echo-Anweisungen ④)

strtotime(): Zeitstempel aus einer englischen Datumsangabe

Aktuelles Datum und aktuelle Uhrzeit: 23.12.2014 - 17:39:26

Funktion	Zeitstempel	Datumsausgabe des Zeitstempels
<code>strtotime("now")</code>	1419352766	Tue, 23 Dec 2014 17:39:26 +0100
<code>strtotime("24 May 2009")</code>	1243116000	Sun, 24 May 2009 00:00:00 +0200
<code>strtotime("+1 day")</code>	1419439166	Wed, 24 Dec 2014 17:39:26 +0100
<code>strtotime("-1 week")</code>	1418747966	Tue, 16 Dec 2014 17:39:26 +0100
<code>strtotime("+1 week 2 days 4 hours 2 minutes")</code>	1420144886	Thu, 01 Jan 2015 21:41:26 +0100

Anzeige der verschiedenen Parameter (Beispieldatei "strtotime.php")

11.6 Datumsangaben überprüfen

Datums- bzw. Zeitangabe auf Gültigkeit überprüfen

Um eine Datumsangabe auf Gültigkeit zu überprüfen, benutzen Sie die Funktion `checkdate()`.

Syntax und Bedeutung der `checkdate()`-Anweisung

- ✓ Die Anweisung erwartet als Parameter den Monat, den Tag sowie das Jahr. Achten Sie bei der Übergabe der Daten auf die richtige Reihenfolge.
- ✓ Monat kann einen Wert zwischen 1 und 12 besitzen. Der Tag ist jeweils abhängig vom Monat. Das Jahr kann einen Wert zwischen 1 und 32767 besitzen.
- ✓ Als Rückgabewert liefert die Funktion den Wert `TRUE`, falls das Datum existiert, sonst `FALSE`.

`checkdate(Monat, Tag, Jahr)`

Beispiel: `checkdate.php`

Im Beispiel wird ein Formular erstellt, das die Eingabe eines Datums in ein HTML-Formular vorsieht. In der selben Datei wird mit PHP überprüft, ob das eingegebene Datum gültig ist.

```

① <h1>Datum eingeben und auf Gültigkeit prüfen</h1>
    <form action=<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
        Geben Sie ein beliebiges Datum im Format TT.MM.JJJJ ein:
        <input type="text" name="datum" size="10" maxlength="10">
        <input type="submit" name="absenden" value="Prüfen">
    </form>
    <?php
②     error_reporting(E_PARSE | E_ERROR);
③     if (isset($_POST["absenden"])) {
④         $data = explode(".", $_POST["datum"]);
⑤         if (!checkdate($data[1], $data[0], $data[2])) or ( count($data) != 3)) {
            echo("<p>" . $_POST["datum"] . " ist kein korrektes Datum!</p>");
        } else {
            echo("<p>Das Datum " . $_POST["datum"] . " ist korrekt!</p>");
        }
    }
?>
```

- ① Die Angabe von `$_SERVER["PHP_SELF"]` im `action`-Parameter des HTML-Formular-Tags `form` legt fest, dass die Formulardaten an die Datei `checkdate.php` selbst zur Auswertung gesendet werden.
- ② Das `error_reporting()` wird auf schwerwiegende Fehler eingestellt, um Warnungen und Hinweise zu unterdrücken, die seit PHP 5.4 standardmäßig angezeigt werden. Wird diese Einstellung nicht vorgenommen, erzeugt `checkdate()` Meldungen vom Typ `Notice` und `Warning`, falls `$_POST["datum"]` kein korrektes Datumsformat hat.
- ③ Es erfolgt die Prüfung auf Existenz der Variablen `$_POST["absenden"]`. Diese Variable ist nur dann vorhanden, wenn die Schaltfläche `Prüfen` gedrückt und damit die Variable `absenden` aus dem Formular übermittelt wurde. Ergibt die Prüfung `TRUE`, wurde das Formular versendet und der Anweisungsblock wird ausgeführt. Andernfalls ist das Formular noch nicht versendet worden, der Anweisungsblock wird übersprungen.
- ④ Das vom Nutzer eingegebene Datum wird über die Zeichenkettenfunktion `explode()` an den Dezimalstellen getrennt, die einzelnen Segmente werden von PHP als indiziertes Array `$data` gespeichert.
- ⑤ Zum Testen des Datums werden über die einzelnen Einträge des Arrays `$data` übergeben. Wird die Eingabe korrekt nach dem Muster `TT.MM.JJJJ` vorgenommen, ist nach dem Einsatz von `explode()` der Monat das zweite Arrayelement (mit dem Index 1), der Tag ist der erste Eintrag im Array (Index 0), das Jahr der dritte Eintrag (Index 2). Je nach Rückgabe der `checkdate()`-Funktion und der Elementanzahl des Arrays `$data` wird eine entsprechende Mitteilung im Browser ausgegeben.



Ausgabe der Beispieldatei "checkdate.php" nach Eingabe des 29.02.2015 und Klick auf den Prüfen-Button

11.7 Übung

Mit Datums- und Zeitangaben arbeiten

Übungsdatei: --

Ergebnisdatei: date_time.php

- ① Geben Sie mithilfe der Funktion `date()` folgende Datums- und Zeitangaben aus. Die Angaben werden automatisch durch das aktuelle Datum ersetzt.

23.12.14
23-12-2014
23.12.2014 - 16:55:12
12/23/14 - 4:55 PM
2014-12-23
16:55 Uhr

- ② Die Funktion `date()` liefert die Tagesnamen standardmäßig in englischer Sprache zurück. Lassen Sie sich den aktuellen Wochentag mittels `setlocale()` und `strftime()` in deutscher Sprache anzeigen.
 ③ Lesen Sie für den aktuellen Zeitstempel den Wochentag aus. Verwenden Sie die Fallauswahl `switch`, um für alle möglichen Wochentage eine beliebige Bildschirmausgabe festzulegen, z. B. *Heute ist Dienstag*.

Zeitmessung durchführen

Übungsdatei: --

Ergebnisdatei: vergleich.php

- ④ Ein Bekannter behauptet, dass PHP Quadratzahlen am schnellsten berechnet, wenn man eine Funktion zur Berechnung von Quadratzahlen erstellt und diese aufruft. Da Sie sich nicht sicher sind, ob eine direkte Berechnung oder der Aufruf einer Funktion schneller ist, wollen Sie beide Varianten vergleichen. Erstellen Sie eine Datei (`vergleich.php`) und berechnen mit einer `for`-Schleife die Quadratzahlen für die Zahlen 1 bis 10000 – einmal über Aufruf einer selbst definierten Funktion und einmal durch direkte Berechnung. Verwenden Sie die Funktion `microtime()`, um die Ausführungszeit beider Berechnungsarten zu vergleichen. Geben Sie das Ergebnis des Vergleichs auf dem Bildschirm aus.



Anzeige der Beispillösung (Datei "vergleich.php")

12 Sessions

In diesem Kapitel erfahren Sie

- ✓ was Sessions sind und zu welchem Zweck sie eingesetzt werden
- ✓ wie Sessions aktiviert werden
- ✓ wie Sessions funktionieren
- ✓ wie Sie Sessions nutzen
- ✓ wie Daten in Sessions gespeichert und abgerufen werden

Voraussetzungen

- ✓ Arbeiten mit Formularen
- ✓ Arrays
- ✓ Kontrollstrukturen

12.1 Mit Sessions arbeiten

Grundlagen zu Sessions

Eine Datenübertragung im Internet erfolgt über das Protokoll HTTP. HTTP ist allerdings ein **zustandsloses Protokoll**, das heißt, mehrere Anfragen, also Seitenaufrufe – auch desselben Seitenbesuchers – werden grundsätzlich als voneinander unabhängige Aktionen betrachtet. Daten, die ein Benutzer beispielsweise in ein Formular eingegeben hat, werden in Variablen nur bis zum auswertenden PHP-Programm weitergegeben. Bereits auf der übernächsten aufgerufenen Webseite sind die Daten nicht mehr vorhanden. Der Webserver kann also, wenn keine weiteren Maßnahmen ergriffen wurden, **nicht** feststellen, welche Seitenaufrufe vom selben Benutzer kommen.

Für diverse Abwicklungen auf Webseiten benötigen Sie jedoch genau eine Funktionalität, über die Sie einen **Besucher wiedererkennen** und mit ihm verbundene Daten zwischenspeichern können. Sie wollen z. B. über mehrere Webseiten Eingaben ermöglichen und dann als Bestellung absenden (Shopsystem). Oder Sie möchten, dass sich auf Ihren Webseiten Benutzer einloggen können und möchten auf jeder weiteren Seite prüfen, ob der Benutzer durch seine Anmeldung autorisiert ist, die jeweilige Seite aufzurufen.

Gerade bei sogenannten personalisierten Seiten, die auf den Benutzer „reagieren“, ist es notwendig, bestimmte Daten, z. B. den Login-Status, von Seite zu Seite mitzuführen bzw. weiterzugeben.

PHP unterstützt diese Anforderungen mithilfe von Sessions (Sitzungen). Eine Session bietet die Möglichkeit, Daten während eines Webseitenbesuchs über mehrere Seiten zwischenspeichern und für jede einzelne Seite verfügbar zu halten. Der Nutzen einer Session liegt darin, dass während einer Session Daten gespeichert und diese von jeder weiteren PHP-Seite weiterverwendet werden können. Eine Session endet,

- ✓ wenn der Benutzer den Browser schließt,
- ✓ wenn eine Webseite mit entsprechenden Befehlen aufgerufen wird,
- ✓ nach einer definierten Zeitspanne
- ✓ oder – je nach Programmierung – auch beim Verlassen der Webseiten, auf denen die Session definiert wurde.



Die Standarddauer einer Session beträgt 1440 Sekunden (**24 Minuten**). Mit jedem neuen Seitenaufruf während der Sitzung wird diese erneut gesetzt. Nach 1440 Sekunden Inaktivität wird die Sitzung des Seitenbesuches beendet. Zum Beenden einer Session reicht oft das Schließen eines einzelnen Tabs des Browsers nicht aus. Die Session wird erst beendet, wenn der **Browser komplett geschlossen** wird.

Session-IDs: Identifikation einer Session

Damit Informationen zu einem Besucher innerhalb einer Website seitenübergreifend verwaltet werden können, erhält ein Besucher beim ersten Zugriff auf mit Sessions definierte Webseiten eine zufällige, eindeutige 32-stellige ID, die **Session-ID**. Diese wird bei jedem Aufruf einer Seite an den Webserver gesendet, woran dieser den Besucher wiedererkennt. Anhand der Session-ID können gespeicherte Daten zugeordnet werden.

Solange der Besucher auf den Webseiten verweilt, auf denen die Session gestartet wurde, werden auf dem Webserver benutzerbezogene Daten in einer **Session-Datei** gespeichert. Dieses können z. B. Formular-eingaben des Nutzers sein, aber auch Daten, die anhand von Login-Daten aus Datenbanken oder anderen Quellen ermittelt wurden. Bei der Session-Datei handelt es sich um eine Textdatei, die auf dem Webserver in einem Ordner für temporäre Dateien abgelegt wird. Bei der Standardeinstellung der im Buch verwendeten XAMPP-Installation handelt es sich unter Windows um das Verzeichnis C:\xampp\tmp. Unter Mac OS befindet sich das entsprechende Verzeichnis unter /Applications/XAMPP/xamppfiles/tmp.

Es handelt sich hierbei um ein serverseitiges Session-Cookie. Cookie ist die Bezeichnung für die Textdatei, die Informationen über einen Vorgang sammelt. In jedem Skript, das zur Session gehört, wird die Session-ID zur abermaligen Authentifizierung benötigt. Die Übertragung der Session-ID von Skript zu Skript wird als **Durchschleifen** bezeichnet. Das Durchschleifen der Session-ID ist einfach zu realisieren, da diese Funktionalität intern von PHP verwaltet wird. Dazu werden die Daten der Session temporär auf dem Webserver gespeichert. Das Durchschleifen einer Session-ID findet nur innerhalb der PHP-Skripte statt, die eine Session mithilfe der Funktion `session_start()` fortsetzen. Durch Aufruf einer Seite mit der Dateinamenerweiterung *.htm(l) oder einer *.php-Seite, auf der `session_start()` nicht aufgerufen wurde, verlässt der Nutzer also die Session.

Als Gegenstück zum Session-Cookie speichert der Browser einen Browser-Cookie mit der Session-ID ab. Die Session-ID wird mit jedem Seitenaufruf an den Server gesendet. Damit wird die konkrete Verknüpfung zwischen Client und Server hergestellt. Akzeptiert der Browser des Benutzers **keine** Cookies, stellt das in der Standardkonfiguration des Webservers ein Hindernis dar. Jedes aufgerufene Skript würde eine neue Session-ID liefern. In diesem Fall können Sie in Ihre PHP-Skripte die folgende Zeile integrieren. Damit wird der Standardwert für `use_trans_sid` von 0 aus der `php.ini` für das aktuelle PHP-Skript gesetzt:

```
ini_set("session.use_trans_sid", 1);
```

Durch diese Einstellung wird die PHP-Konfiguration für das aufrufende Skript geändert. Die Session-ID wird automatisch über die URL übergeben und ist somit in der Adresszeile des Browsers sichtbar (z. B. [...]index.php?PHPSESSID=ro9k4jhqitc1jipkgd25jafn03). Allerdings stellt dies ein Sicherheitsrisiko dar. Links mit einer Session-ID können unter den Favoriten gespeichert oder einschließlich Session-ID per E-Mail versendet werden. Diese Links können mit anderen Browsern geöffnet werden. Falls mit der Session-ID Login-Informationen verknüpft sind, wäre ein zweiter Nutzer über einen solchen Link auf einer Seite angemeldet, obwohl er sich nicht authentifiziert hat.

Sie können die Session-ID sowie weitere Daten aus einer Session auch in einem Cookie speichern, das beim Benutzer abgelegt wird. Auf die Möglichkeit, mit Cookies beim Benutzer zu arbeiten, wird in diesem Buch nicht eingegangen.

12.2 Session starten bzw. fortsetzen

Wenn Sie mit einer Session arbeiten wollen, gehen Sie wie folgt vor:

- ✓ Allen Dateien, die innerhalb der Session erreichbar sein sollen, geben Sie die Dateinamenerweiterung *.php.
- ✓ Jede dieser Dateien beginnt mit Öffnen eines PHP-Blocks (<?php) in der ersten Zeile der Datei, gefolgt von der Funktion `session_start()`, um eine Session zu starten bzw. eine laufende Session fortzusetzen. Das gilt auch für Dateien, die ausschließlich HTML-Tags enthalten.

Um eine Session zu starten, verwenden Sie die Funktion `session_start()`. Beim Aufruf des Programms wird eine Session gestartet und eine eindeutige Session-ID erzeugt. Auf die Session-ID können Sie bei Bedarf mit der Funktion `session_id()` zugreifen.

Syntax und Bedeutung der Funktion `session_start()`

```
session_start();
```

- ✓ Bei Aufruf der Funktion `session_start()` wird eine neue Session begonnen oder eine aktuelle Session wieder aufgenommen. Ist die vom User übermittelte Session-ID aktiv, wird diese ID von PHP automatisch erkannt und damit die bestehende Session fortgeführt. Wird keine Session-ID übergeben, generiert PHP eine neue Session-ID und startet damit eine neue Session.
- ✓ Jede Webseite, die zu einer Session gehören soll, muss den Befehl `session_start()` beinhalten, ansonsten kann keine Session begonnen bzw. fortgesetzt werden.
- ✓ Die Funktion erwartet keine Parameter.
- ✓ Rückgabewert der Funktion ist `TRUE` (bei Erfolg) bzw. `FALSE` (bei Fehlschlag).



Sie müssen `session_start()` aufrufen, bevor Sie eine Ausgabe im Browser vornehmen. Eine echo-Anweisung in PHP oder ein HTML-Tag vor dem Aufruf dieser Funktion führen zu einer Fehlermeldung ("header already sent"). Denselben Effekt haben Leerzeichen außerhalb von PHP-Blöcken. Auch ein versehentlich eingefügtes Leerzeichen hinter dem schließenden PHP-Tag einer inkludierten PHP-Datei, die vor dem Aufruf von `session_start()` eingebunden wird, kann zu dieser Fehlermeldung führen. Um Schwierigkeiten zu vermeiden, sollten Sie als Erstes ganz oben in der Datei einen PHP-Block einfügen und danach `session_start()` aufrufen.

Standardmäßig lautet der Name der Session `PHPSESSID`, der in der Datei `php.ini` definiert ist. Sie können eine Session zur Laufzeit auch mit einem anderen Namen versehen. Über diesen Namen können Sie diese Session im späteren Verlauf wieder ansprechen.

Syntax und Bedeutung der Funktion `session_name()`

```
session_name(Bezeichnung);
```

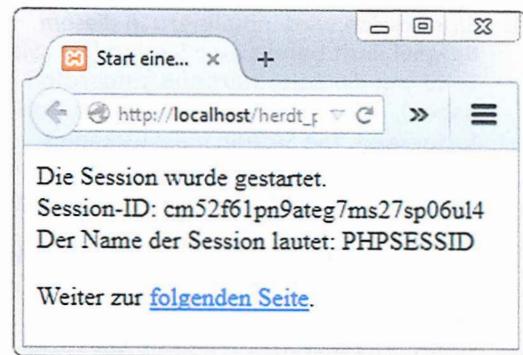


Möchten Sie bei Aufruf einer Webseite einen eigenen Session-Namen definieren, rufen Sie die Funktion `session_name()` **vor** dem Starten der Session mit `session_start()` auf. Damit wird der Wert der Session mit dem angegebenen Namen erstellt bzw. übernommen. Achtung: Der Parameter Bezeichnung darf nicht nur aus Zahlen bestehen, sondern muss mindestens einen Buchstaben enthalten.

Beispiel `start.php`

①	<?php
②	<code>session_start();</code>
③	<code>\$id = session_id();</code> <code>echo "<!DOCTYPE html><html><head><title>Start einer</code> <code> Session</title></head><body>Die Session wurde gestartet.
";</code> <code>echo "Session-ID: " . \$id;</code> <code>echo "
Der Name der Session lautet: " . session_name();</code> <code>echo "<p>Weiter zur folgenden</code> <code> Seite.</p></body></html>";</code> <code>?></code>

- ① Mit dem Befehl `session_start()` wird eine neue Session gestartet oder eine gestartete Session fortgesetzt.
- ② Der Variablen `$id` wird die aktuelle Session-ID zugewiesen.
- ③ Über die Funktion `session_name()` wird der Name der aktuellen Session ausgegeben.



Anzeige der Beispieldatei "start.php"

12.3 Daten einer Session speichern

PHP hilft Ihnen bei Start und Betrieb einer Session. Die automatisch angelegte Session-Datei kann gewünschte Informationen aufnehmen und für die Dauer der Session verfügbar halten. Das Speichern von Daten in einer Session hingegen geschieht nicht automatisch, sondern hängt von der Programmierung ab. Um Daten in der Session-Datei zu speichern, verwenden Sie die superglobale Array-Variable `$_SESSION`.

```
$_SESSION["Element"] = Wert;
z.B. $_SESSION["benutzer"] = "Max";
```

Mit dieser Zuweisung erreichen Sie, dass das Element unter dem Array-Schlüssel "benutzer" in die Session-Datei eingetragen wird. Damit steht es in der Session als Element der Array-Variablen `$_SESSION` zur Verfügung. Am Ende eines PHP-Skripts werden üblicherweise alle Variablen aus dem Arbeitsspeicher des Servers gelöscht. So auch die Werte aus `$_SESSION`. Setzt das als nächstes aufgerufene Skript die Session fort, werden die Daten aus der Session-Datei in die Variable `$_SESSION` eingelesen. Die Inhalte stehen somit wieder zur Verfügung, nachdem Sie `session_start()` aufgerufen haben. Dieser Vorgang wiederholt sich von Skript zu Skript bis zum Ende der Session.

Daten in einer Session speichern

In ein Formular werden Daten eingegeben. Im auswertenden Programm werden diese Daten an Variablen übergeben, auf die Sie innerhalb einer Session von jeder anderen Seite zugreifen können.

Beispiel: `formular.php`

```
① <?php
    session_start();
?><!DOCTYPE html>
<html>
    <head><title>Formular zur Eingabe der Daten</title></head>
    <body>
        <h1>Session: Angaben zur Person</h1>
        <p>Bitte füllen Sie die nachfolgenden Eingabefelder aus: </p>
        <form action="auswertung.php" method="POST">
            <p>Vorname: <input type="text" name="vorname"></p>
            <p>Nachname: <input type="text" name="nachname"></p>
            <p>Wohnort: <input type="text" name="ort"></p>
            <p><input type="submit" value="Abschicken"></p>
        </form>
    </body>
</html>
```

- ① Eine Session wird initialisiert. In diesem Beispiel läuft bereits eine Session (falls Sie die Seite von der Seite `start.php` aufgerufen haben), also wird die bestehende Session fortgesetzt. Die Session muss lückenlos fortgesetzt werden, deshalb erhält auch die Datei, die nur ein HTML-Formular beinhaltet, die Dateinamenerweiterung `*.php` und initialisiert eine Session. Da der PHP-Block zur Initialisierung der Session notwendig ist, muss es eine PHP-Datei sein. In HTML-Dateien können Sie diesen PHP-Block nicht einsetzen.
- ② Ab hier wird ein einfaches Formular aufgebaut. Als Ziel-Skript des Formulars wird über das `action`-Attribut die PHP-Datei `auswertung.php` definiert, in der die Auswertung des Formulars vorgenommen werden soll.

Formular zur Eingabe der D...

Vorname: _____

Nachname: _____

Wohnort: _____

Abschicken

Ausgabe der Beispieldatei "formular.php"

Beispiel: `auswertung.php`

```
<?php
    session_start();
    echo "<!DOCTYPE html><html><head><title>Daten ins Session speichern</title></head><body><h1>Daten in der Session speichern</h1>";
    echo "<p>Sie haben folgende Daten im Formular eingetragen:</p>";
    echo "<br>Vorname: " . $_POST["vorname"];
    echo "<br>Nachname: " . $_POST["nachname"];
    echo "<br>Ort: " . $_POST["ort"] . "</p>";
    $_SESSION["vorname"] = $_POST["vorname"];
    $_SESSION["nachname"] = $_POST["nachname"];
    $_SESSION["ort"] = $_POST["ort"];
    $_SESSION["zeit"] = time();
    echo "<p>Folgende Daten sind nun in der Session gespeichert: </p>";
    echo "<pre>";
    print_r($_SESSION);
    echo "</pre>";
    echo "<p>Weiter zur <a href='auslesen.php'>folgenden Seite</a>.</p></body></html>";
?>
```

- ① Die laufende Session wird fortgesetzt.
- ② Die Daten aus dem übermittelten Formular werden über die Array-Variable `$_POST` ausgegeben.
- ③ In `$_SESSION` werden die Variablen `vorname`, `name` und `ort` (Daten aus dem Formular) sowie zusätzlich die Variable `zeit`, die den aktuellen Zeitstempel enthält, gespeichert. Damit werden die Angaben in der aktuellen Session-Datei im Ordner `C:\xampp\tmp` (Windows) bzw. `/Applications/XAMPP/xamppfiles/tmp` (Mac) gespeichert und für den Zugriff über die Variable `$_SESSION` während der gesamten Session bereitgestellt.
- ④ Zur Überprüfung der in der Session gespeicherten Variablen wird mithilfe der Funktion `print_r()` die Variable `$_SESSION` ausgelesen und angezeigt.

Session-Datei anzeigen lassen

Session-Variablen können vom Client nicht manipuliert werden. Nachdem sie erzeugt und gespeichert wurden, existieren sie nur im Datenspeicher des Servers und können somit nur vom Skript gelesen werden. PHP liest diese Textdatei am Anfang jeder Session ein. Bei jeder Wertzuweisung an die Session-Variable `$_SESSION` speichert PHP die geänderten oder neu hinzugekommenen Sessiondaten wieder ab.

The screenshot shows a browser window with the title "Daten ins Session speichern". The URL is `http://localhost/herdt_p`. The page content is as follows:

Daten in der Session speichern

Sie haben folgende Daten im Formular eingetragen:
 Vorname: Max
 Nachname: Mustermann
 Ort: Musterhausen

Folgende Daten sind nun in der Session gespeichert:

```
Array
(
    [vorname] => Max
    [nachname] => Mustermann
    [ort] => Musterhausen
    [zeit] => 1419354329
)
```

Weiter zur [folgenden Seite](#).

Ausgabe der Beispieldatei "auswertung.php"

- Wechseln Sie unter Windows in den Ordner `C:\xampp\temp` bzw. `/Applications/XAMPP/xamppfiles/tmp` unter Mac OS.
- Öffnen Sie die Datei mit der aktuellen Session-ID mit Notepad++ oder einem einfachen Texteditor. Der Dateiname der Session-Datei besteht aus dem Präfix `sess_`, gefolgt von der Session-ID (z. B. `sess_fam2lg77kt9k26r1ufii3j95u5`).

Die Datei hat folgenden Inhalt (in Form eines sogenannten serialisierten Array):

```
vorname|s:3:"Max";nachname|s:10:"Mustermann";ort|s:12:"Musterhausen";zeit|i:1345294585;
```

Die einzelnen Variablen werden nach folgendem Schema abgelegt:

```
Name|Datentyp[:Länge]:Variablen-Inhalt;
```

The screenshot shows a browser window with the title "Start einer Sess...". The URL is `http://localhost/herdt_p`. The page content is as follows:

Die Session wurde gestartet
 Session-ID: cm52f61pn9ateg7ms27sp06u14
 Der Name der Session lautet: PHPSESSID

Anzeige der aktuellen Session-ID

The screenshot shows a Notepad++ editor window titled "sess_cm52f61pn9ateg7ms27sp06u14 - Editor". The content of the file is:

```
Datei Bearbeiten Format Ansicht ?
vorname|s:3:"Max";nachname|s:10:"Mustermann";ort|s:12:"Musterhausen";zeit|i:1419354656;
```

Session-Datei mit gespeicherten Daten

12.4 Daten einer Session abrufen

Die gespeicherten Werte einer Session können Sie über den Namen der jeweiligen Variablen direkt ermitteln bzw. über die Funktion `list()` komplett auslesen. Da es sich bei der Variablen `$_SESSION` um eine Array-Variable handelt, können Sie alle Funktionen zur Weiterverarbeitung der Daten anwenden, die Sie aus dem Kapitel Arrays kennen.

Beispiel: `auslesen.php`

Die Werte, die Sie im vorigen Beispiel in der Session-Datei gespeichert haben, möchten Sie auf einer neuen Seite auslesen. In dem Beispiel werden zwei Möglichkeiten aufgezeigt.

```

① <?php
    session_start();
?><!DOCTYPE html>
<html>
    <head><title>Auslesen der Sessiondaten</title></head>
    <body>
        <h1>Auslesen der Sessiondaten</h1>
        <?php
            echo "<p><em>Folgende Variablen wurden gespeichert ('list'): </em><br>";
            while (list($key, $value) = each($_SESSION)) {
                echo $key . " = " . $value . "<br>";
            }
            echo "<p><em>Direktes Ansprechen der Variablen:</em><br>";
            echo "Vorname aus der Session-Datei: " . $_SESSION["vorname"] . "<br>";
            echo "Nachname aus der Session-Datei: " . $_SESSION["nachname"] . "<br>";
            echo "Ort aus der Session-Datei: " . $_SESSION["ort"] . "<br>";
            echo "Zeitstempel aus der Session-Datei: " . $_SESSION["zeit"] . "</p>";
            echo "<p>Weiter zur <a href='session_destroy.php'>folgenden Seite</a>.</p>";
        ?>
    </body>
</html>
```

- ① Die laufende Session wird fortgesetzt.
- ② Über die `while`-Schleife und die Funktion `list()` wird jedes Element des Arrays `$_SESSION` aus-gelesen. Mit den Standardvariablen `$key` und `$value` können Sie den Schlüssel und den Wert jeder Session-Variablen am Bildschirm anzeigen lassen.
- ③ Zusätzlich wird jede Variable einzeln anhand des Namens angesprochen und ausgegeben.



Anzeige der Beispieldatei "auslesen.php"

12.5 Sessiondaten und Session löschen

Neben dem Speichern und Auslesen der Daten können Sie diese auch komplett löschen und die Session beenden. Dies ist z. B. notwendig, wenn in einem Online-Shop die Bestellung ausgeführt wurde und somit der Warenkorb des Kunden gelöscht wird.

Löschen der Sessiondaten

Wenn Sie alle Session-Variablen löschen wollen, also komplett die Inhalte der Array-Variablen `$_SESSION` sowie den Inhalt der Session-Datei, dann verwenden Sie die folgende Anweisung:

```
$_SESSION = array();
```

- ✓ Mit dieser Anweisung initialisieren Sie die Session-Variable neu und leeren damit das vorhandene Array bzw. Sie ersetzen das bestehende Session-Array durch ein leeres Array. Die Session läuft nach dieser Anweisung weiter.

Wollen Sie einzelne Variablen der Session löschen und die restlichen Daten unberührt lassen, verwenden Sie die Funktion `unset()`:

```
unset(Variable);
```

- ✓ Sie geben als Parameter `Variable` die genaue Bezeichnung der Variablen an, die Sie löschen möchten. Beispielsweise möchten Sie die Angabe zum Vornamen aus den vorherigen Beispielen in diesem Kapitel aus der Session entfernen: `unset($_SESSION["vorname"]);`

Löschen der Session mit der Funktion `session_destroy()`

Wenn Sie die komplette Session beenden wollen, reicht es nicht aus, die Sessiondaten zu löschen. Die Session läuft weiter. Durch Ausführen der Funktion `session_destroy()` wird die Session-Datei gelöscht. Beim nächsten Versuch, Sessiondaten zu speichern, oder beim Versuch, eine Session zu initialisieren, meldet PHP zurück, dass es die dazugehörige Session-Datei nicht (mehr) findet. In diesem Fall wird eine neue Session gestartet. Eine neue Session-ID wird erzeugt und die dazugehörige Session-Datei angelegt.

```
session_destroy();
```

- ✓ Mit `session_destroy()` beenden Sie die aktuelle Session. Dabei wird die Session-Datei im Ordner `C:\xampp\tmp` bzw. `/Applications/XAMPP/xamppfiles/tmp` gelöscht.

Beispiel: `session_destroy.php`

Die Daten der bisherigen Session sollen gelöscht und anschließend die gesamte Session beendet werden.

```

①  <?php
    session_start();
?><!DOCTYPE html>
<html>
    <head><title>Sessiondaten und Session löschen</title></head>
    <body>
        <h1>Sessiondaten und Session löschen</h1>
        <?php
        echo "<pre>";
②      print_r($_SESSION);
        echo "</pre>";
③      unset($_SESSION["vorname"]);
        echo "<pre>";
④      print_r($_SESSION);
        echo "</pre>";
        $_SESSION = array();
⑤      print_r($_SESSION);
        echo "<p>Die Session mit der ID " . session_id() . " wurde ";
        if (session_destroy()) {
            echo "erfolgreich gelöscht.";
        } else {
            echo "nicht gelöscht.";
        }
        echo "</p>";
    ?>
</body>
</html>

```

- ① Die aktuelle Session wird fortgesetzt.
- ② Mit der Funktion `print_r()` geben Sie zur Kontrolle die Session-Variablen mehrmals am Bildschirm aus. Die Ausgabe des `pre`-Tags vor und hinter dem `print_r()` dient der zeilenweisen Ausgabe im Browser.
- ③ Mithilfe der Funktion `unset()` löschen Sie die angegebene Session-Variable (`$_SESSION["vorname"]`). Alle anderen Session-Variablen bleiben unberührt. Die Session läuft weiter.
- ④ Indem Sie der Variablen `$_SESSION` ein leeres Array zuweisen, löschen Sie alle Session-Variablen.
- ⑤ Danach löschen Sie die Session-Datei und damit die Session. Die Funktion `session_destroy()` liefert einen Wert zurück, und zwar `TRUE`, wenn der Löschvorgang erfolgreich war, bzw. `FALSE` bei Fehlgeschlagen der Aktion. Über eine `if`-Abfrage generieren Sie eine entsprechende Ausgabe.



Ausgabe der Beispieldatei "session_destroy.php"

12.6 Fallbeispiel "Shop"

Sie haben in den vorigen Abschnitten die grundlegenden Funktionen rund um Sessions kennengelernt. Die bisherigen Beispiele waren allerdings linear aufgebaut. In typischen Anwendungsbeispielen von Sessions wie z. B. Shops werden Sie jedoch mit weiteren Problemen konfrontiert:

- ✓ Sie arbeiten in der Regel innerhalb einer Session mit mehreren Formularen und müssen aufpassen, dass sich in der Session gespeicherte Daten der einzelnen Formulare nicht gegenseitig überschreiben.
- ✓ Sie rufen ein Formular eventuell mehrfach auf. Im Idealfall sind bestellte Artikelmengen eingetragen und Sie können jederzeit weitere Artikel bestellen oder Bestellmengen verändern.
- ✓ Sie können die Seiten beliebig innerhalb der Session wechseln. Daten dürfen dabei nicht verloren gehen.

Beschreibung des Fallbeispiels

Um die genannten Sachverhalte abzubilden, zeigt das Fallbeispiel einen kleinen Shop mit zwei Formularen (= Artikelgruppen), über welche Sie Artikel bestellen können. Sie können jederzeit zum Warenkorb sowie auf die einzelnen Formularseiten wechseln. Um den Bestellvorgang zu beenden, geben Sie auf einer abschließenden Seite Ihre persönlichen Daten ein. Die Daten zur Bestellung werden ausgegeben und in einer *.csv-Datei gespeichert. Danach wird die Session beendet.

Das Fallbeispiel ist möglichst einfach gehalten. Aus Gründen der Übersichtlichkeit und Nachvollziehbarkeit wurden weder ein ansprechendes Layout noch zusätzliche Funktionen programmiert. Das Shopbeispiel besteht aus folgenden Dateien:

Dateiname	Kurzbeschreibung
<i>fallbeispiel_start.php</i>	Startseite des Shops, mit Links auf die Formulare zur Bestellung von Schokolade und Pralinen
<i>fallbeispiel_artikel.inc.php</i>	Include-Datei, in der die verwendeten Artikel in Array-Variablen zur Verfügung gestellt werden
<i>fallbeispiel_form-schoko.php</i> <i>fallbeispiel_form-praline.php</i>	Formulardateien zur Bestellung von Artikeln. Pro Artikelgruppe wird eine separate Formulardatei verwendet.
<i>fallbeispiel_warenkorb.php</i>	Zentrale Auswertungsdatei für die übermittelten Daten aus den Formularen und dem Warenkorb
<i>fallbeispiel_kasse.php</i>	Datei zum Abschließen des Bestellvorgangs, Eingabe persönlicher Daten, Speichern der Bestellung in einer externen Datei und Beenden der Session

Beispiel: *fallbeispiel_start.php*

Die Startseite des Fallbeispiels verlinkt auf die Formulare im Shop. Die Datei trägt die Dateinamen-erweiterung **.php*, die Session wird über die Funktion *session_start()* initialisiert. Beachten Sie, dass eine Session gestartet wird, obwohl die Datei abgesehen von der Initialisierung der Session ausschließlich aus HTML-Elementen besteht.

Ausgabe der Startseite des Fallbeispiels
"fallbeispiel_start.php"

Beispiel: fallbeispiel_artikel.inc.php

In der Include-Datei werden in Array-Variablen die Artikelnummern und Artikelbezeichnungen definiert. In der Praxis werden diese Angaben in der Regel aus Datenbanken oder vergleichbaren Quellen ausgelesen und über entsprechende Variablen zur Verfügung gestellt.

```

① <?php
    $schoko_feld = array("s-1" => "Weiße Schokolade",
                          "s-2" => "Vollmilch-Schokolade",
                          "s-3" => "Bio-Vollmilch-Schokolade",
                          "s-4" => "Zartbitter-Schokolade");

    $praline_feld = array("p-1" => "Marzipan-Pralinen",
                          "p-2" => "Mokka-Pralinen",
                          "p-3" => "Nougat-Pralinen",
                          "p-4" => "Walnuss-Pralinen");
?
```

- ① Die Variable \$schoko_feld wird als Array-Variable definiert. Gespeichert wird ein assoziatives Array mit Angaben zu Artikelnummern (Schlüssel) und Artikelbezeichnungen (Werte). Analog dazu wird für jede weitere Artikelgruppe eine weitere Array-Variable nach gleichem Muster angelegt.

Beispiel: fallbeispiel_form-schoko.php (fallbeispiel_form-praline.php)

Die beiden Formulare sind nach dem gleichen Muster aufgebaut. Die Artikel werden aus der Datei *fallbeispiel_artikel.inc.php* eingelesen. Der Benutzer kann für die Artikel die gewünschten Bestellmengen eingeben. Bei nochmaligem Aufruf der Formulardatei wird eine eventuell früher eingetragene Bestellmenge aus der Session-Variablen ausgelesen und angezeigt. In diesem Beispiel können Sie durch Eingabe des Wertes 0 bei der Bestellmenge den Artikel aus der Session löschen.

```

① <?php
    session_start();
②     include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html>
<html>
    <head><title>Schokolade-Bestellformular</title></head>
    <body>
        <h1>Fallbeispiel "Shop": Formular 1 - Schokolade</h1>
        <p>Bestellung: Schokolade - tragen Sie die gewünschte Menge ein.</p>
        <form action="fallbeispiel_warenkorb.php" method="POST">
            <table border="1" bgcolor="#D5F0F5">
                <tr><th>Art.-Nr.</th><th>Artikel</th><th>Menge</th><th>Einheit</th></tr>
                <?php
                    foreach ($schoko_feld as $key => $value) {
                        echo "<tr><td align='center'>$key</td><td>$value</td>";
                        echo "<td><input type='text' name='$key' value='";
                        if (isset($_SESSION[$key])) {
                            echo $_SESSION[$key];
                        } else {
                            echo "0";
                        }
                        echo "></td><td>Stück</td></tr>";
                    }
                <?php
            </table>
            <input type="submit" value="Bestellen" />
        </form>
    </body>
</html>
```

```

⑤      value="" . (isset($_SESSION[$key]) ? $_SESSION[$key] : '0') . ""
      size='5' style='text-align:right'>";
echo "</td><td>Tafel (100g)</td></tr>";
}
?>
<tr>
<td colspan="4">
<input type="submit" name="schoko" value="In den Warenkorb">
<input type="submit" name="abbruch" value="Abbrechen">
</td>
</tr>
</table></form></body></html>

```

- ① Durch Angabe von `session_start()` wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Mithilfe der Funktion `include()` wird die Datei `fallbeispiel_artikel.inc.php` eingebunden, die Informationen zum Warenbestand enthält.
- ③ Die Datei `fallbeispiel_warenkorb.php` wird als Ziel-Skript des Formulars und damit zur Auswertung der Eingaben definiert.
- ④ In einer `foreach`-Schleife wird das Array der Artikel der gewünschten Warengruppe ausgelesen. Schlüssel und Wert werden flexibel in einer Tabelle dargestellt. Enthalten ist ein Eingabefeld, in das der Benutzer die Bestellmenge eingibt.
- ⑤ Durch das Attribut `value` des `input`-Elements wird die Vorbelegung gesetzt, die davon abhängt, ob bereits im Laufe der Session eine Bestellmenge für diesen Artikel ausgewählt wurde. Falls ja, wird der vorhandene Wert aus der Session ausgelesen, falls nicht wird der Wert `0` eingetragen. Zu diesem Zweck wird im Beispiel die Kurzschreibweise der `if-else`-Anweisung - der ternäre Operator - verwendet.
- ⑥ Die `Submit`-Schaltfläche namens `schoko` sorgt dafür, dass das Formular abgesendet wird. In dem Fall werden die eingegebenen Mengen aus dem Formular in der Session gespeichert.
- ⑦ Eine zweite `Submit`-Schaltfläche (`abbruch`) übergibt ebenfalls das Formular. In der auswertenden Datei werden dann allerdings keine Eingaben verarbeitet.

Art.-Nr.	Artikel	Menge	Einheit
s-1	Weiße Schokolade	5	Tafel (100g)
s-2	Vollmilch-Schokolade	0	Tafel (100g)
s-3	Bio-Vollmilch-Schokolade	0	Tafel (100g)
s-4	Zartbitter-Schokolade	3	Tafel (100g)

Anzeige der Beispieldatei "fallbeispiel_form-schoko.php" mit einigen Beispieldaten

Beispiel: `fallbeispiel_warenkorb.php`

Die Auswertung der Formulareingaben der beiden Bestellformulare erfolgt in einer einzigen Datei, der Datei `fallbeispiel_warenkorb.php`. Durch eine `if`-Anweisung wird geprüft, aus welchem Formular Daten übergeben wurden. Die Daten werden in die Session-Datei übernommen, ohne dass bereits vorhandene Daten anderer Formulare beeinflusst werden. Das Skript dient einem weiteren Zweck, nämlich der Anzeige des Warenkorbs. In der Datei sind Verlinkungen auf alle anderen Dateien innerhalb des Shops enthalten.

```

<?php
①   session_start();
②   include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html>
<html>
  <head><title>Ihr Warenkorb</title></head>
  <body>
    <h1>Ihr Warenkorb</h1>
    <?php
      ③ if (isset($_POST["schoko"]) or isset($_POST["praline"])) {
        ④   while (list ($key, $value) = each($_POST)) {
          ⑤     if ($value >= 1) {
            $_SESSION[$key] = floor($value);
          } else {
            if (isset($_SESSION[$key])) {
              ⑥               unset($_SESSION[$key]);
            }
          }
        }
      }
      ⑦ echo "<table border='1'>
              <tr><th>Art.-Nr.</th><th>Artikel</th><th>Menge</th></tr>";
      while (list($key, $value) = each($_SESSION)) {
        if (substr($key, 0, 1) == "s") {
          echo "<tr><td>$key</td><td>$schoko_feld[$key]</td>
                  <td>$value</td></tr>";
        }
        if (substr($key, 0, 1) == "p") {
          echo "<tr><td>$key</td><td>$praline_feld[$key]</td>
                  <td>$value</td></tr>";
        }
      }
      echo "</table>";
    ?>
    <p>Was möchten Sie tun?</p>
    <ul>
      <li><a href="fallbeispiel_form-schoko.php">Schokolade bestellen</a></li>
      <li><a href="fallbeispiel_form-praline.php">Pralinen bestellen</a></li>
      <li><a href="fallbeispiel_kasse.php">Bestellung abschließen</a></li>
    </ul>
  </body></html>

```

- ① Durch Angabe von `session_start()` wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Mithilfe der Funktion `include()` wird die Datei `fallbeispiel_artikel.inc.php` eingebunden, welche Informationen zum Warenbestand enthält.
- ③ Es wird geprüft, ob die Variable `$_POST["schoko"]` oder die Variable `$_POST["praline"]` vorhanden ist. Das ist nur dann der Fall, wenn Daten aus einer der Formulardateien zur Auswertung versendet wurden. Je nachdem, ob Sie vom Formular für Pralinen- oder von dem für Schokoladenbestellung zu diesem Skript gelangt sind, werden nur die dort vorgenommenen Eintragungen in der Session gespeichert. Durch dieses Vorgehen können sich Daten aus verschiedenen Formularen nicht gegenseitig überschreiben.

- ④ Über eine while-Schleife werden alle Formulardaten (`$_POST`) mithilfe der `list()`-Funktion Variablen zugewiesen. Die Artikelnummer bildet den Schlüssel (`$key`), die vom Benutzer eingegebene Bestellmenge den Wert (`$value`). Es wird abgefragt, ob mindestens der Wert 1 bei der Bestellmenge eingegeben wurde. Damit scheiden Zeichenketten aus, da sie den Zahlenwert 0 besitzen. Abgefangen werden auch Gleitkommazahlen durch die Verwendung von `floor()`, wodurch jeder Zahlenwert auf die nächst kleinere Ganzzahl abgerundet wird. Der Wert wird in der Session gespeichert.
- ⑤ Entspricht der Wert nicht den genannten Kriterien, wird die Variable für diesen Artikel - sofern vorhanden - aus der Session-Datei gelöscht. Ein Eintrag von 0 im Bestellformular löscht den betreffenden Eintrag wieder aus der Session.
- ⑥ Analog zur while-Schleife zum Einlesen der Formulardaten ④ erfolgt hier das Einlesen der Sessiondaten, um den aktuellen Warenkorb darstellen zu können.
- ⑦ Zur Darstellung der einzelnen Warengruppen ist neben der in der Session-Datei gespeicherten Angabe zur Artikelnummer das Auslesen der Artikelbezeichnung geplant. Aus diesem Grund wird über die Funktion `substr($key, 0, 1)` der erste Buchstabe der Artikelnummer ausgelesen. In diesem Beispiel sind nur die Werte s (Schokolade) oder p (Praline) möglich. Abhängig davon werden die Artikelbezeichnungen aus den Artikel-Array-Variablen entnommen.

Art.-Nr.	Artikel	Menge
s-1	Weiße Schokolade	5
s-4	Zartbitter-Schokolade	3
p-1	Marzipan-Pralinen	2
p-2	Mokka-Pralinen	1
p-4	Walnuss-Pralinen	12

Was möchten Sie tun?

- [Schokolade bestellen](#)
- [Pralinen bestellen](#)
- [Bestellung abschließen](#)

Anzeige des Warenkorbs im Fallbeispiel
(Datei "fallbeispiel_warenkorb.php")

Beispiel: fallbeispiel_kasse.php

Die Bestellung soll abgeschlossen werden. Der Benutzer gibt zu diesem Zweck seine persönlichen Daten ein (in diesem Beispiel nur Vorname, Name und Ort). Danach werden die kompletten Bestelldaten am Bildschirm angezeigt und in einer *.csv-Datei zur Weiterverarbeitung mit Excel gespeichert. Danach wird der Inhalt der Sessionvariablen `$_SESSION` und abschließend die komplette Session gelöscht.

```

<?php
① session_start();
② include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html><html>
<head><title>Kasse</title></head>
<body>
<h1>Fallbeispiel "Shop": Bestellung abschließen</h1>
<?php
③ if (isset($_POST["absenden"])) {
    $vorname = $_POST["vorname"];
    $nachname = $_POST["nachname"];
    $ort = $_POST["ort"];
    echo "<p>Sie haben folgende Bestellung übermittelt:</p>";
    echo "<p><strong>$vorname $nachname aus $ort</strong></p>";
    echo "<table border='1'><tr><th>Art.-Nr.</th><th>Artikel</th>
        <th>Menge</th></tr>";
    $bestellung = "\n\nArt.-Nr.;Artikel;Menge\n";
    ⑥ while (list($key, $value) = each($_SESSION)) {
        if (substr($key, 0, 1) == "s") {
            echo "<tr><td>$key</td><td>$schoko_feld[$key]</td>
                <td>$value</td></tr>";
            $bestellung .= "$key;$schoko_feld[$key];$value\n";
        }
    }
}

```

```

    if (substr($key, 0, 1) == "p") {
        echo "<tr><td>$key</td><td>$praline_feld[$key]</td>
              <td>$value</td></tr>";
        $bestellung .= "$key;$praline_feld[$key];$value\n";
    }
}
$bestellung .= "\nbestellt von\n$n$vorname;$nachname;$ort\n\n";
echo "</table><p>Vielen Dank! Die Session wird beendet.</p>";
if (file_put_contents("bestellung.csv", $bestellung, FILE_APPEND)) {
    echo "<p><em>Die Bestelldaten wurden in der Datei bestellung.csv
          gespeichert</em></p>";
}
$_SESSION = array();
session_destroy();
} else {
?>
<p>Bitte füllen Sie die nachfolgenden Eingabefelder aus: </p>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="POST">
    <p>Vorname: <input type="text" name="vorname"></p>
    <p>Nachname: <input type="text" name="nachname"></p>
    <p>Wohnort: <input type="text" name="ort"></p>
    <p><input type="submit" name="absenden" value="Absenden und Bestellung
          abschließen"></p>
</form>
<?php
}
?>
</body></html>

```

- ① Durch Angabe von `session_start()` wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Mithilfe der Funktion `include()` wird die Datei `fallbeispiel_artikel.inc.php` eingebunden, die Informationen zum Warenbestand enthält.
- ③ Zum Abschluss der Bestellung steht in dieser Datei das Formular zur Eingabe persönlicher Daten zur Verfügung ⑩. Mithilfe von `isset($_POST["absenden"])` wird geprüft, ob das Formular bereits ausgefüllt wurde. Das Formular wird nur angezeigt, wenn es noch nicht abgesendet wurde ⑨.
- ④ Die Formulardaten werden zur einfachen Verwendung in den Variablen `$vorname`, `$nachname` und `$ort` gespeichert.
- ⑤ Die Variable `$bestellung` wird definiert und nach und nach mit Inhalt gefüllt. Die Variable beinhaltet später alle Bestelldaten.
- ⑥ Auch in diesem Skript erfolgten das Auslesen der Sessiondaten und die Darstellung des Warenkorb-Inhalts auf dem Bildschirm.
- ⑦ Die Variable `$bestellung` wird mithilfe der Funktion `file_put_contents()` in die Datei `bestellung.csv` im selben Ordner gespeichert. Die Daten werden an eventuell bestehende Daten angehängt. Im Erfolgsfall wird eine entsprechende Meldung auf dem Bildschirm ausgegeben.
- ⑧ Schließlich wird die Variable `$_SESSION` geleert und die komplette Session gelöscht.

Fallbeispiel "Shop": Bestellung abschließen

Sie haben folgende Bestellung übermittelt:

Max Mustermann aus Musterhausen

Art.-Nr.	Artikel	Menge
s-1	Weiße Schokolade	5
s-4	Zartbitter-Schokolade	3
p-1	Marzipan-Pralinen	2
p-2	Mokka-Pralinen	1
p-4	Walnuss-Pralinen	12

Vielen Dank! Die Session wird beendet.

Die Bestelldaten wurden in der Datei bestellung.csv gespeichert

Anzeige der Beispieldatei "fallbeispiel_kasse.php" bei Eintrag persönlicher Daten (links) und bei Ende des Bestellvorgangs (rechts)

Zusätzlich wurden die Daten in der Datei *bestellung.csv* zur Weiterverarbeitung, z. B. durch Excel, gespeichert:

```
Art.-Nr.;Artikel;Menge
s-1;Weiße Schokolade;5
s-4;Zartbitter-Schokolade;3
p-1;Marzipan-Pralinen;2
p-2;Mokka-Pralinen;1
p-4;Walnuss-Pralinen;12

bestellt von
Max;Mustermann;Musterhausen
```

Inhalt der Datei "bestellung.csv" nach Ausführung einer Beispielbestellung

12.7 Übung

Einen einfachen Shop erstellen

Übungsdatei: --

Ergebnisdateien: *u_formular.php*, *u_bestellung.php*, *u_abschluss.php*

- ① Erstellen Sie ein Bestellformular zur Bestellung von Honig (*u_formular.php*). Beginnen Sie mit dieser Datei eine Session. Die in das Bestellformular eingegebenen Daten sollen an die Datei *u_bestellung.php* übergeben werden.

The screenshot shows a web browser window with the title "Bestellfo...". The URL bar displays "http://localhost/herdt...". The main content area has a heading "Übung: Honigbestellung". Below it is a text input field with the placeholder "Bitte geben Sie die Bestellmenge an (Einheit: 500-g-Glas)". Underneath is a table with two columns: "Honig" and "Menge". The table contains five rows, each with a dropdown menu and an input field. The last row contains a button labeled "Abschicken".

Honig	Menge
Akazienhonig	
Heidehonig	
Kleehonig	
Tannenhonig	
<input type="button" value="Abschicken"/>	

Übungsdatei "*u_formular.php*"

- ② Speichern Sie in der Datei *u_bestellung.php* die Daten aus dem Formular in der Session und lassen Sie die Sessiondaten inklusive Session-ID anzeigen.

The screenshot shows a web browser window with the title "Bestellü...". The URL bar displays "http://localhost/herdt...". The main content area has a heading "Honigbestellung". Below it is a text "Sie haben folgende Mengen bestellt:" followed by a list of honig types and their quantities: Akazienhonig: 6 Gläser, Heidehonig: 7 Gläser, Tannenhonig: 12 Gläser. Then, it says "Die Session-ID lautet:" followed by a long session ID: 898412soh0qji27qggumlr4i82. At the bottom right, there is a link "Weiter zur Eingabe persönlicher Daten" and the text "dem Abschluss der Bestellung."

Anzeige der Übungsdatei
"u_bestellung.php"

- ③ Verlinken Sie auf eine weitere Datei innerhalb der Session (*u_abschluss.php*), in der in einem Formular Angaben zu Name, Wohnort und Mailadresse gemacht werden können.

Honigbestellung - Abschluss

Bitte geben Sie noch Ihre Kontaktdaten ein:

Vorname : Max

Nachname : Mustermann

Wohnort : Musterhausen

Mailadresse : nax@mustermann.com

Anzeige der Übungsdatei
"u_abschluss.php" zur Eingabe
persönlicher Daten

- ④ Lesen Sie anschließend nach Absenden des Formulars in derselben Datei die kompletten Sessiondaten über eine Schleife aus und geben Sie sie am Bildschirm aus. Abschließend sollen die Session-Variable geleert und die Session beendet werden.

Dies sind die in der Session gesammelten Daten:

Akazienhonig: 6
Heidehonig: 7
Kleehonig:
Tannenhonig: 12
Vorname: Max
Nachname: Mustermann
Wohnort: Musterhausen
Mailadresse: max@mustermann.com

Damit ist die Session beendet. [Klicken Sie hier](#), um eine neue Session zu beginnen.

Anzeige der Zusammenfassung der Sessiondaten

- ⑤ Zusatzaufgabe: Schauen Sie sich das Fallbeispiel auf den vorigen Seiten genau an und überlegen Sie, welche Anpassungen Sie vornehmen können, z. B. Layout, andere oder weitere Artikel, Erweitern des Beispiels um Preisangaben und -berechnungen. Setzen Sie eine Ihrer Ideen auf der Basis der vorliegenden Dateien um.

13 Grundlagen Datenbank MySQL

In diesem Kapitel erfahren Sie

- ✓ wie Sie mit dem MySQL-Server arbeiten
- ✓ welche Möglichkeiten die Benutzeroberfläche phpMyAdmin bietet
- ✓ wie eine Verbindung zur Datenbank hergestellt wird
- ✓ wie Sie eine SQL-Abfrage an den Server übergeben
- ✓ wie Sie Rückgabewerte der SQL-Abfragen erhalten und weiterverarbeiten

Voraussetzungen

- ✓ Aufbau und Struktur von Datenbanken
- ✓ SQL-Syntax
- ✓ Kontrollstrukturen
- ✓ Arrays
- ✓ Arbeiten mit Formularen

13.1 MySQL-Datenbanken mit phpMyAdmin verwalten

PHP unterstützt die Arbeit mit vielen verschiedenen Datenbankmanagementsystemen. Der Zugriff und die Steuerung von Datenbanken durch PHP folgen ähnlichen Gesetzmäßigkeiten, größtenteils unabhängig vom verwendeten Datenbankmanagementsystem. Im Internet ist die Kombination von PHP mit MySQL am weitesten verbreitet. Daher wird in diesem Kapitel das Datenbankmanagementsystem MySQL in der Zusammenarbeit mit PHP betrachtet.

In dem für dieses Buch eingesetzten Bundle XAMPP (Version 5.6.3) sind das Datenbankmanagementsystem MySQL, Version 5.6.21, und phpMyAdmin, Version 4.2.11, enthalten.



Der XAMPP-Webserver wird regelmäßig auf neue PHP-, MySQL- und phpMyAdmin-Versionen aktualisiert (nicht jede neue Version einer Software wird integriert, einzelne Versionen von PHP bzw. MySQL werden mitunter übersprungen). So werden Sie in Zukunft XAMPP-Installationspakete finden, welche neuere Versionen der einzelnen Komponenten berücksichtigen. Diese können im Detail (Layout, Funktionen), besonders in Bezug auf phpMyAdmin, von der in diesem Buch verwendeten Version abweichen.

MySQL



XAMPP Control Panel: Schaltzentrale zum Starten und Beenden der Komponenten.

Die Modulnamen *Apache* und *MySQL* sind hellgrün hinterlegt, wenn die Server laufen (Windows) bzw. durch grüne Symbole im manager-osx markiert (Mac OS, siehe Anhang).

MySQL ist ein häufig genutztes Open-Source-Datenbankmanagementsystem, dessen Sprachumfang weitgehend dem SQL3-Standard entspricht. Mithilfe von SQL-Anweisungen werden Datenbanken, Tabellen und die darin enthaltenen Datensätze sowie Benutzer und ihre Berechtigungen verwaltet.

Zur Verwaltung von MySQL-Datenbanken wird häufig die Bedienoberfläche phpMyAdmin verwendet. Beide Komponenten (MySQL und phpMyAdmin) werden bei der Installation von XAMPP mit eingerichtet. Achten Sie im XAMPP Control Panel darauf, dass der Apache-Webserver und der MySQL-Server gestartet sind. Nur dann können Sie mit PHP und MySQL arbeiten. Falls das nicht der Fall sein sollte, starten Sie beide Komponenten über einen Klick auf die jeweils nebenstehende Schaltfläche *Start*.



PHP ist nicht auf MySQL beschränkt, sondern kann mit diversen anderen Datenbanken arbeiten. Mitunter findet man auch den Einsatz von ORACLE Datenbanken bei Webapplikationen mit PHP. Diese werden vor allem dann eingesetzt, wenn die Datenmengen, die verarbeitet werden müssen, sehr groß sind und die Datenbanken hoch performant sein müssen. Die Interaktionen zwischen PHP und den unterschiedlichen Datenbanken sind prinzipiell vergleichbar. Da MySQL am häufigsten in Kombination mit PHP zu finden ist, wird in diesem Buch anhand von MySQL und mithilfe von phpMyAdmin das Arbeiten mit Datenbanken vorgestellt.



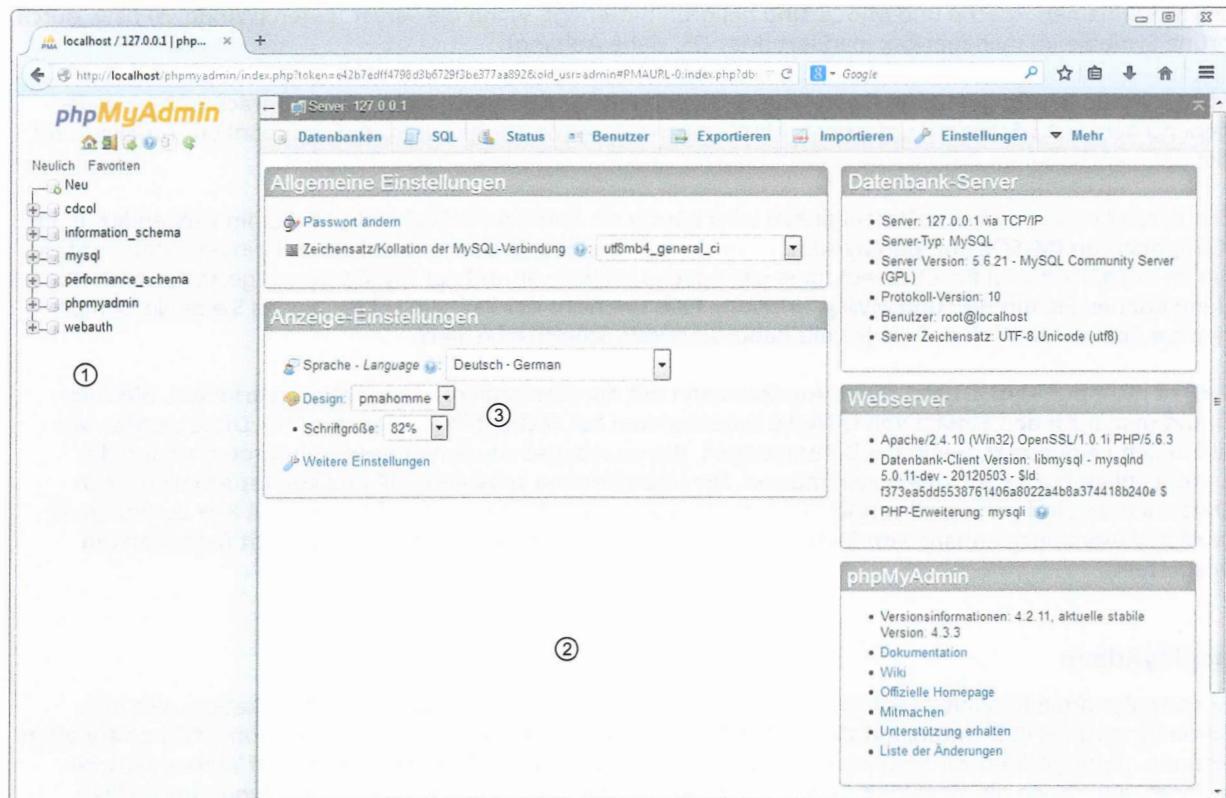
phpMyAdmin

Bei phpMyAdmin handelt es sich um eine in PHP geschriebene Open-Source-Web-Applikation, also eine Sammlung von PHP-Skripten, mit der Sie MySQL-Datenbanken über eine grafische Weboberfläche verwalten können. phpMyAdmin wird hauptsächlich zur Verwaltung von MySQL-Datenbanken auf Webservern verwendet, auf denen die einzelnen Kunden keine Rechte haben, die von MySQL zur Verfügung gestellten Kommandozeilen-Verwaltungstools direkt auszuführen. Die meisten Internetprovider bieten phpMyAdmin als Standardtool für die Administration von MySQL-Datenbanken an.

phpMyAdmin ermöglicht Ihnen folgende Aktionen:

- ✓ Erstellen und Löschen von Datenbanken
- ✓ Erstellen, Kopieren, Löschen und Ändern von Tabellen
- ✓ Hinzufügen, Löschen und Ändern von Datensätzen
- ✓ Ausführen von SQL-Anweisungen
- ✓ Anzeigen der Datensätze in den Tabellen
- ✓ Laden von Textdaten in Tabellen
- ✓ Exportieren von Daten in unterschiedliche Formate, beispielsweise in CSV- oder SQL-Dateien
- ✓ Importieren von Daten
- ✓ Administrieren verschiedener Server und einzelner Datenbanken
- ✓ Verwaltung von MySQL-Benutzern

Die Startseite von phpMyAdmin rufen Sie über die Navigation der XAMPP-Oberfläche auf, die Sie über die Eingabe von <http://localhost> im Browser erreichen. Alternativ können Sie die direkte URL <http://localhost/phpmyadmin/> aufrufen.



Startseite von phpMyAdmin

Über diese Oberfläche können die verschiedenen MySQL-Befehle per Hyperlink, Schaltflächen oder über Eingabefelder ausgeführt werden. Auf der linken Seite finden Sie die Auflistung aller vorhandenen Datenbanken des MySQL-Servers ①. Mit der Auswahl einer Datenbank öffnen Sie die entsprechenden Tabellen. Im rechten Hauptbildschirm ② werden die einzelnen Funktionen als Eingabefelder und Schaltflächen angeboten. Auf der Startseite können Sie die Sprache auswählen, die von phpMyAdmin verwendet werden soll. Hier können Sie auch zum Layout vorheriger Versionen wechseln ③.

13.2 MySQL-Datenbanken mit phpMyAdmin erstellen

Neue Datenbanken in MySQL anlegen

- ▶ Um eine neue Datenbank zu erzeugen, wählen Sie in der oberen Navigation den Bereich *Datenbanken* aus und geben in das Eingabefeld *Neue Datenbank anlegen* den Namen, hier z. B. *obstladen*, für die neue Datenbank ein und klicken Sie auf die Schaltfläche *Anlegen*.
- ▶ Der Zeichensatz (Kollation) legt den Default-Schriftzeichensatz für die Datenbank fest. Dieser beeinflusst, wie später Datensätze sortiert werden. Auch die einzelnen Tabellen können jeweils mit einer bestimmten Kollation angelegt werden. Da seit der Version PHP 5.4 UTF-8 als Standardzeichensatz festgelegt ist, empfiehlt es sich auch hier, den Zeichensatz *utf8_general_ci* auszuwählen.

Neue Datenbank anlegen

Tabellen erstellen

Haben Sie eine neue Datenbank erstellt, erkennt phpMyAdmin, dass die neue Datenbank noch keine Tabellen enthält. Es wird Ihnen ein Formular angezeigt, über welches Sie neue Tabellen erstellen können.

- Tragen Sie in das Eingabefeld **Name** ① den Tabellennamen **bestellung** und in das Eingabefeld **Anzahl der Felder** ② die Anzahl der Felder - nämlich "6" - ein, die in der Tabelle erstellt werden sollen.
- Betätigen Sie die Schaltfläche **OK** ③.

Neue Tabelle mithilfe von phpMyAdmin erstellen

Felder erstellen

Es öffnet sich ein neues Fenster im Webbrowser, in dem die von Ihnen angegebene Anzahl an Feldern angezeigt wird.

- Tragen Sie in die vorhandenen Eingabefelder gemäß folgender Tabelle die entsprechenden Werte ein.
Danach können Sie über verschiedene Auswahl- und Eingabefelder die Eigenschaften der einzelnen Tabellenfelder konfigurieren.

Name	Typ	Länge/Werte	Standard	Kollation	Attribute	Null	Index	A_I
id	INT		Kein(e)					
vorname	VARCHAR	50	Kein(e)					
nachname	VARCHAR	50	Kein(e)					
ort	VARCHAR	50	Kein(e)					
sorte	VARCHAR	20	Kein(e)					
menge	INT		Kein(e)					

Der Aufbau der Beispieldatentabelle ist nur exemplarisch. Angaben wie Straße, PLZ, Mailadresse etc. fehlen. Außerdem würde sich u. a. für eine reale Datenbank zur Vermeidung redundanter Daten die Teilung in Kundentabelle und Bestelltabelle anbieten. Auch das Vorhandensein der möglichen Nullwerte soll die Arbeit mit diesem Beispiel vereinfachen.



Datentypen zuweisen

In den Feldern können Sie folgende Eintragungen bzw. Einstellungen vornehmen. Für das Beispiel sind folgende Felder zu beachten:

Feld	Beschreibung	Parameter in SQL-Syntax
Name	Tragen Sie in dieses Eingabefeld den Feldnamen ein.	<Feldname>
Typ	Aus diesem Auswahlfeld wählen Sie den Datentyp des Feldes aus. Im Beispiel werden aus einer Vielzahl möglicher Datentypen nur die Typen INT (Ganzzahl) und VARCHAR (<i>various characters</i> , beliebiger Text) verwendet.	<Datentyp>
Länge/Set	In dieses Eingabefeld tragen Sie die maximale Länge des Datenfeldinhaltes ein.	(AnzahlZeichen)
Standard	Tragen Sie in dieses Eingabefeld den Standardwert ein, der dem Feld zugewiesen werden soll, wenn das Feld bei der Dateneingabe keinen Wert erhält.	DEFAULT '<Wert>'

Feld	Beschreibung	Parameter in SQL-Syntax
Null	Wenn das Feld beim Eintragen eines Datensatzes leer bleiben darf, wird das Häkchen gesetzt, anderenfalls wird eine Eingabe für das Feld erwartet.	NULL oder NOT NULL
Index	Das Kombinationsfeld bietet u. a. die Einträge <i>Primary</i> und <i>Unique</i> zur Festlegung eines Feldes als Primärschlüssel bzw. als Feld, das eindeutige Werte enthalten muss.	PRIMARY, UNIQUE
A_I	Über dieses Kontrollkästchen können Sie den Wert <code>auto_increment</code> einem ganzzahligen Datenbankfeld zuweisen. Beim Einfügen eines neuen Datensatzes in die Tabelle wird der Wert des mit <code>auto_increment</code> gekennzeichneten Feldes automatisch durch MySQL um eins erhöht (inkrementiert). Innerhalb der Tabelle darf nur ein Feld mit diesem Attribut versehen werden, dieses Feld ist automatisch der Primärschlüssel der Tabelle.	AUTO_INCREMENT

Primärschlüssel anlegen

Um ein Feld als Primärschlüssel zu definieren, wählen Sie im Feld *Index* die entsprechende Option aus. Beachten Sie bei der Zuweisung des Primärschlüssels, dass innerhalb einer Tabelle ...

- ✓ nur ein Primärschlüssel zugewiesen werden darf;
- ✓ das Feld, dem der Primärschlüssel zugewiesen wird, nicht als *Null*-Feld definiert werden darf;
- ✓ das Feld, dem der Primärschlüssel zugewiesen wird, eindeutige Werte enthalten muss (kein Wert darf mehrfach vorhanden sein. Beim Versuch, denselben Wert mehrfach einzufügen, reagiert MySQL mit einer Fehlermeldung).

 Pro Tabelle gibt es maximal einen Primärschlüssel. Die Einstellung *Unique* können Sie mehrfach definieren. Felder, die als *Unique* definiert werden, können ebenfalls nur einmal mit demselben Wert gefüllt werden. Auch hier reagiert MySQL mit einer Fehlermeldung, falls ein Wert für ein *Unique*-Feld bereits vorhanden ist und Sie diesen noch einmal eintragen möchten.

Sollten Sie in einer Tabelle kein Feld mit eindeutigen Werten haben, wird empfohlen, wie im Beispiel ein zusätzliches Feld mit einer automatisch generierten laufenden Nummerierung (Auto-Inkrement-Wert) hinzuzufügen, das als Primärschlüssel dient.

Tabelle speichern

- Betätigen Sie die Schaltfläche *Speichern*, um die Tabelle zu speichern.

Neue Tabelle in vorhandener Datenbank erstellen

 Wollen Sie in einer vorhandenen Datenbank eine neue Tabelle erstellen, wählen Sie in der linken phpMyAdmin-Navigation die Tabelle aus. Unter der Übersicht der vorhandenen Tabellen finden Sie oben beschriebenes Formular zum Anlegen von Tabellen

- Stellen Sie sicher, dass Sie sich nach dem Erstellen einer neuen Datenbank im Register *Struktur* befinden.
- oder Klicken Sie im Startbildschirm von phpMyAdmin auf den Eintrag *Datenbanken* und in der Übersicht der vorhandenen Datenbanken auf den Namen der Datenbank, in der Sie eine Tabelle erstellen wollen.

Beispiel: Datenbank *obstladen*

The screenshot shows the phpMyAdmin interface for the 'obstladen' database. The 'bestellung' table is selected. The table structure includes columns: id (int(11)), vorname (varchar(50)), nachname (varchar(50)), ort (varchar(50)), sorte (varchar(20)), and menge (int(11)). The 'Aktion' column contains links for 'Bearbeiten', 'Löschen', 'Primärschlüssel', 'Unique', 'Index', and 'Mehr' for each row. Below the table, there is a toolbar with buttons for 'Druckansicht', 'Beziehungsübersicht', 'Tabellenstruktur analysieren', 'Tabelle verfolgen', and 'Spalten verschieben'. A modal dialog box is open, showing the 'Information' tab with details about the table's storage usage and statistics.

#	Name	Typ	Kollation	Attribute	Null	Standard	Extra	Aktion
1	id	int(11)		Nein	kein(e)		AUTO_INCREMENT	Bearbeiten Löschen Primärschlüssel Unique Index Mehr
2	vorname	varchar(50)	latin1_swedish_ci	Ja	NULL			Bearbeiten Löschen Primärschlüssel Unique Index Mehr
3	nachname	varchar(50)	latin1_swedish_ci	Ja	NULL			Bearbeiten Löschen Primärschlüssel Unique Index Mehr
4	ort	varchar(50)	latin1_swedish_ci	Ja	NULL			Bearbeiten Löschen Primärschlüssel Unique Index Mehr
5	sorte	varchar(20)	latin1_swedish_ci	Nein	kein(e)			Bearbeiten Löschen Primärschlüssel Unique Index Mehr
6	menge	int(11)		Ja	NULL			Bearbeiten Löschen Primärschlüssel Unique Index Mehr

Alle auswählen markierte: Anzeigen Bearbeiten Löschen Primärschlüssel Unique Index

Druckansicht Beziehungsübersicht Tabellenstruktur analysieren Tabelle verfolgen Spalten verschieben

1 Spalte(n) einfügen An das Ende der Tabelle An den Anfang der Tabelle Nach **id** OK

+ Indizes

Information

Speicherplatzverbrauch

Daten	16 KiB	Format	Compact
Index	0 B	Kollation	latin1_swedish_ci
Insgesamt	16 KiB	Nächster Autoindex	1
Erzeugt am 24. Dez 2014 um 15:08			

Tabelle "bestellung" in der Datenbank "obstladen" in phpMyAdmin

Tabellenstruktur verändern

Die Tabellenstruktur können Sie im Nachhinein ändern, beispielsweise mit den Hyperlinks **Bearbeiten** oder **Löschen**. Diese Hyperlinks befinden sich in der Spalte **Aktion** neben den Feldnamen in der Tabellenstrukturansicht und sind in phpMyAdmin mit sprechenden Links versehen, die auf die möglichen Optionen hinweisen:

The screenshot shows the 'Aktion' section of the table structure. It displays multiple rows of actions for each column: 'Bearbeiten', 'Löschen', 'Primärschlüssel', 'Unique', 'Index', 'Räumlich', 'Volltext', and 'Unterschiedliche Werte'. The 'Bearbeiten' and 'Löschen' links are underlined, indicating they are active or clickable.

Neben dem Verändern der Tabellenstruktur haben Sie auch die Möglichkeit, neue Felder hinzuzufügen. Unterhalb der Strukturübersicht der Tabelle finden Sie folgendes Formular:

The screenshot shows the 'Spalte(n) einfügen' (Add Column) form. It includes fields for 'Anzahl' (1), 'Position' ('An das Ende der Tabelle'), and 'Nach' ('id'). There is also an 'OK' button.

1 Spalte(n) einfügen An das Ende der Tabelle An den Anfang der Tabelle Nach **id** OK

Um neue Datenfelder zur bestehenden Tabelle hinzuzufügen, geben Sie die Anzahl der neuen Felder ein und wählen Sie aus, an welcher Position innerhalb der Tabelle die neuen Felder erstellt werden sollen.

Im HERDT-Buch *PHP - Fortgeschrittene Techniken der Web-Programmierung* wird das Thema PHP und MySQL vertiefend behandelt. Sie finden dort sowohl ausführliche Informationen zu phpMyAdmin als auch einen Kurzüberblick über die wichtigsten SQL-Befehle sowie Tipps zu fortgeschrittenen Techniken rund um PHP und MySQL.

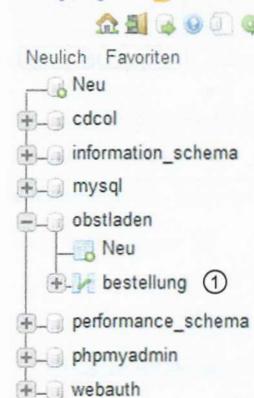
13.3 Mit einer MySQL-Tabelle arbeiten

Datensätze in eine Tabelle eintragen

Sie können Datensätze in eine MySQL-Tabelle sowohl mithilfe von phpMyAdmin als auch, wie später beschrieben, über PHP-Befehle eintragen.

- ▶ Wechseln Sie mit dem Hyperlink *Tabellename* ① in die Tabellenstrukturansicht.
- ▶ Klicken Sie am oberen Bildschirmrand auf den Hyperlink *Einfügen*, um Daten für einen Datensatz einzugeben.
- ▶ Klicken Sie auf die Schaltfläche *OK* ②, um zur Tabelle zurückzukehren, oder wählen Sie aus dem Kombinationsfeld ③ den Eintrag *anschließend einen weiteren Datensatz einfügen*, um einen weiteren Datensatz einzufügen.
- ▶ Im Beispiel handelt es sich bei dem Feld *id* um ein Feld mit einer automatischen Nummerierung ④. Aus diesem Grund lassen Sie das Feld leer. MySQL vergibt automatisch den nächsthöheren freien Wert bei der Speicherung des Datensatzes.
- ▶ Bestätigen Sie mit *OK*.
- ▶ Über den Hyperlink *Anzeigen* können Sie sich die Datensätze der Tabelle ansehen.

phpMyAdmin



Einen neuen Datensatz in eine Tabelle einfügen

- ▶ Tragen Sie die Datensätze aus der folgenden Abbildung in die Tabelle ein.

			id	vorname	nachname	ort	sorte	menge
<input type="checkbox"/>			1	Arndt	Hoffmann	Stuttgart	Elstar	15
<input type="checkbox"/>			2	Corinna	Delphi	Hamburg	Janagold	3
<input type="checkbox"/>			3	Petra	Meyer	Wien	Gala	5
<input type="checkbox"/>			4	Peter	Schmidt	Berlin	Elstar	25

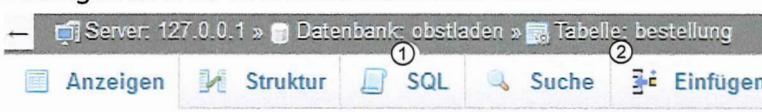
Beispieldatenebene "bestellung" in Datenbank "obstladen"

13.4 SQL-Dumps exportieren und importieren

Für die Arbeit mit einem PHP-Projekt werden Sie in der Regel mit bereits vorhandenen Datenbanken arbeiten. phpMyAdmin stellt Funktionen für den Import und Export von Datenbank- und Tabellenstruktur sowie der Daten bereit.

Daten exportieren

Datenbank- und Tabellenstruktur sowie die Daten selber können per Klick aus phpMyAdmin exportiert werden. Das Ergebnis eines Exports ist ein sogenannter **SQL-Dump** (SQL-Exportdatei). Dieser kann als Datei gespeichert werden. Es handelt sich um eine Text-Datei mit SQL-Statements, die zum Import der Daten dient. Für SQL-Dumps steht die Dateinamenerweiterung `*.sql` zur Verfügung.

- ▶ Je nachdem, ob Sie eine ganze Datenbank oder nur eine Tabelle exportieren möchten, klicken Sie in der phpMyAdmin-Navigation links eine Datenbank oder eine Tabelle an.
- ▶ Wählen Sie in der oberen Menüleiste im Hauptfenster den Menüpunkt **Exportieren**. Falls nur die Datenbank ① angezeigt wird, können Sie alle Tabellen der Datenbank exportieren. Wird dort zusätzlich ein Tabellename ② angezeigt, führt der Klick auf *Exportieren* zum Export der einzelnen Tabelle.


phpMyAdmin bietet zwei Varianten des Datenexports, den schnellen sowie den angepassten. Standardmäßig ist *Schnell* ausgewählt. Hier müssen Sie keine weiteren Einstellungen vornehmen, der SQL-Dump wird mit Standardeinstellungen generiert, der für den Import der Tabellen in andere Datenbanken notwendig ist.

Möchten Sie zusätzlich die Datenbank selber exportieren, müssen Sie unter dem Punkt *Art des Export* die Auswahl *Angepasst* treffen und zusätzlich den Befehl `CREATE DATABASE` aktivieren.
- ▶ Bestätigen Sie mit OK. Wählen Sie im geöffneten Dialogfenster, ob Sie den SQL-Dump mit einem Editor öffnen oder auf Ihrer Festplatte speichern möchten.

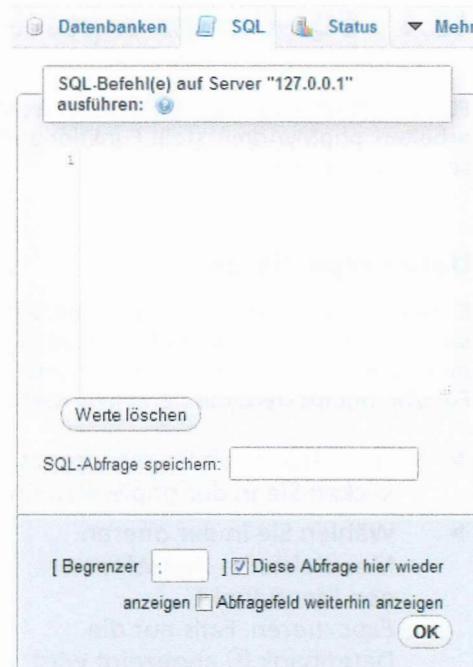
Ein SQL-Dump enthält alle SQL-Befehle, die zum Anlegen von Datenbanken und Tabellen bzw. zum Einfügen von Daten notwendig sind. Die SQL-Befehle liegen genau in der Reihenfolge vor, wie sie für den Import von Datenbank- und Tabellenstruktur und Daten benötigt werden. SQL-Dumps müssen mit der Dateinamenerweiterung `*.sql` gespeichert werden.
- ▶ Alternativ zum Export als Downloaddatei können Sie bei Auswahl des angepassten Datenexports die Option *Ausgabe als Text anzeigen* auswählen. In dem Fall werden die SQL-Befehle in einem Textfeld angezeigt und können von dort direkt zum Import herauskopiert werden.

Daten importieren

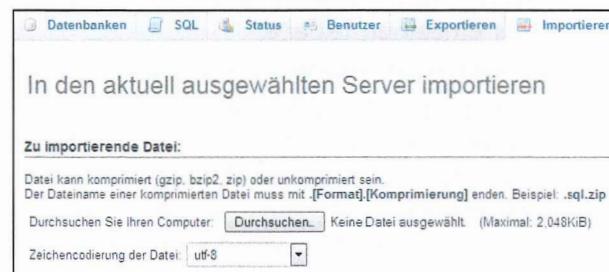
Der Import eines SQL-Dumps ist einfach. Um die Daten einer `*.sql`-Datei zu importieren, haben Sie zwei Möglichkeiten:

- ▶ Öffnen Sie die SQL-Datei mit einem Texteditor, z. B. Notepad++.
- ▶ Wählen Sie in der oberen Menüleiste den Eintrag *SQL* aus. Falls Sie nur eine Tabelle in eine Datenbank importieren möchten, wählen Sie zuerst eine Datenbank aus. Sie erkennen an der obersten Leiste im Hauptfenster von phpMyAdmin, ob Sie sich auf der Datenbankserverübersicht oder auf der Übersicht einer ausgewählten Datenbank befinden.

- ▶ Nach Klick auf den Menüpunkt *SQL* öffnet sich ein Textfeld zur Eingabe. Kopieren Sie die Befehle aus der SQL-Datei in das Textfeld hinein, bestätigen Sie das Speichern mit dem Button *OK*. Die Datenbank bzw. Tabelle wird importiert, indem alle SQL-Befehle ausgeführt werden.
- ▶ Im Fehlerfall zeigt phpMyAdmin eine Fehlermeldung an. Übliche Fehler sind z. B.: Eine Datenbank, die Sie importieren möchten, existiert bereits oder eine fehlerhafte SQL-Datei bzw. Datensätze mit eindeutigen IDs, die in Tabellen bereits vergeben sind.
- ▶ Alternativ können Sie auch den Menüpunkt *Importieren* wählen. In dem Fall steht Ihnen ein Upload-Formular zur Verfügung.
- ▶ Über die Schaltfläche *Durchsuchen...* wählen Sie die SQL-Datei auf Ihrer Festplatte aus.



- ▶ Bestätigen Sie den Button *OK*. Der Import startet. Je nach Größe der Datei dauert dies einen Moment. Auch hier kann es zur Fehleranzeige kommen, welche die gleichen Ursachen haben, wie oben bereits beschrieben.



Beispieldaten importieren: *obstladen-bestellung.sql*

Die in den vorherigen Kapiteln erstellte Datenbank, die Tabelle sowie die Daten stehen als SQL-Dump unter den Beispieldaten zur Verfügung. Beachten Sie bei Import-Dateien folgendes:

- ▶ Das CREATE-Statement für die Datenbank wird mit `CREATE DATABASE IF NOT EXISTS` eingeleitet. Die Datenbank wird nur dann erzeugt, wenn keine Datenbank mit dem Namen *obstladen* vorhanden ist.
- ▶ Das Statement zum Erstellen der Tabelle wird ebenfalls mit `CREATE TABLE IF NOT EXISTS` eingeleitet. Die Tabelle wird nur erzeugt, wenn sie noch nicht vorhanden ist.
- ▶ Sowohl Datenbank als auch Tabelle werden mit dem UTF8-Zeichensatz angelegt.
- ▶ Die Import-Datei enthält den Befehl `TRUNCATE TABLE 'bestellung';`. Dieser Befehl löscht eventuell vorhandene Daten aus der Tabelle *bestellung*.
- ▶ Die Daten der Tabelle werden importiert, der Primärschlüssel wird gesetzt, der Wert für das Autoinkrement wird angepasst.

13.5 PHP und MySQL

Nachdem Sie eine MySQL-Datenbank sowie eine Tabelle darin erstellt haben, können Sie mithilfe von PHP Datensätze aus einer Tabelle bearbeiten, beispielsweise anzeigen, ändern, löschen, aber auch erzeugen. Die notwendigen Schritte für den Zugriff auf eine Datenbank werden in diesem Kapitel schrittweise gezeigt. Zur Bearbeitung der Datenbank benötigen Sie entsprechende Berechtigungen (vgl. Anhang).



PHP nutzte früher für den Zugriff auf Datenbanken die Erweiterung *MySQL*. Dies war bislang die Standardmethode, um mit Datenbanken zu arbeiten. Seit der Version PHP 5.5 gilt diese Erweiterung als veraltet. Das bedeutet, die klassische *MySQL*-Erweiterung kann noch genutzt, die Erweiterung wird aber in zukünftigen PHP-Versionen entfernt werden. Stattdessen empfiehlt www.php.net, die *mysqli*-Erweiterung (*MySQL Improved Extension*) oder die *PDO_MYSQL*-Erweiterung (*PHP Data Objects (PDO) interface*) zu nutzen. Beide Erweiterungen nutzen den objektorientierten Ansatz der PHP-Programmierung (mehr über Objektorientierung finden Sie im HERDT-Buch *PHP - Fortgeschrittene Techniken der Web-Programmierung*). Obwohl auch *mysqli* eine objektorientiert Klasse ist, bietet diese Erweiterung sogenannte Alias-Funktionen. Das sind Funktionsaufrufe, die den Funktionen der alten *MySQL*-Erweiterung in der Schreibweise ähneln, tatsächlich verbergen sich dahinter aber Methodenaufrufe der *mysqli*-Klasse. Aufgrund der ähnlichen und einfacheren Syntax der *mysqli*-Funktionen wird in diesem Buch diese PHP-Erweiterung für das Arbeiten mit Datenbanken vorgestellt.

Benutzer für Datenbankzugriffe aus PHP anlegen

Mit der Installation von XAMPP wird standardmäßig ein Nutzerkonto für den Zugriff auf den Datenbankserver eingerichtet. Der Name des Nutzers ist `root`, dieser wird ohne Passwort angelegt. Dieses Nutzerkonto verfügt über die Administrationsrechte für den kompletten Datenbankserver. Der `root`-Nutzer sollte in PHP-Skripten auf keinen Fall genutzt werden. Sowohl die vollständigen Administrationsrechte als auch das fehlende Passwort stellen ein hohes Sicherheitsrisiko dar.

- Legen Sie zuerst einen Nutzer für die Verwendung in PHP-Skripten an. Im phpMyAdmin finden Sie in der oberen Menüleiste den Link *Benutzer* ①. Über diesen Menüpunkt gelangen Sie zu der Übersicht bereits hinterlegter Benutzer. Unterhalb dieser Tabelle finden Sie einen Link *Benutzer hinzufügen* ②.

Mit Klick auf den Link *Benutzer hinzufügen* öffnet sich das entsprechende Formular. Der Benutzernamen ist frei wählbar. Das Feld *Host* ist beim Öffnen mit einem %-Zeichen vorausgefüllt. Dieses können Sie belassen, damit kann der Benutzer den Datenbankserver über einen beliebigen Host ansprechen, z. B. auch über die IP-Adresse des Rechners. Falls Sie im Auswahlfeld *Host* den Eintrag *Lokal* auswählen, wird das Feld automatisch mit dem Wert *localhost* gefüllt. Damit schränken Sie den Zugriff auf den Datenbankservernamen *localhost* ein, was eine höhere Sicherheit darstellt. Vergeben Sie zusätzlich ein sicheres Passwort oder verwenden Sie die Schaltfläche *Generieren*, um ein Passwort zu erzeugen.

The screenshot shows the 'User Management' section of phpMyAdmin. At the top, there's a navigation bar with tabs: 'Datenbanken', 'SQL', 'Status', 'Benutzer' (selected), and 'Exp'. Below the navigation is a sub-navigation with 'Benutzerübersicht' (selected) and 'Benutzergruppen'. A circled '1' is located near the 'Benutzerübersicht' tab. The main area is titled 'Benutzerübersicht' and contains a table of users:

Benutzer	Host	Passwort	Globale Rechte	Benutzergruppe
<input type="checkbox"/> Jeder	%	--	USAGE	
<input type="checkbox"/> Jeder	localhost	Nein	USAGE	
<input type="checkbox"/> pma	localhost	Nein	USAGE	
<input type="checkbox"/> root	127.0.0.1	Nein	ALL PRIVILEGES	
<input type="checkbox"/> root	::1	Nein	ALL PRIVILEGES	
<input type="checkbox"/> root	localhost	Nein	ALL PRIVILEGES	

Below the table are buttons: 'Alle auswählen' (Select all), 'markierte:' (Selected), 'Exportieren' (Export), and 'Benutzer hinzufügen' (Add User) with a circled '2' next to it.

The 'Anmelde-Informationen' (Login Information) form is shown below, with fields for 'Benutzename', 'Host', 'Passwort', 'Wiederholen', and 'Passwort generieren'.

Zusätzlich müssen die Rechte für den Benutzer gesetzt werden. Für einfache Datenbankabfragen reichen die Rechte aus der Gruppe *Daten* ①. Sollen über PHP auch Änderungen an der Tabellenstruktur möglich sein, vergeben Sie zusätzlich entsprechende Rechte aus der Gruppe *Struktur* ②. Administrationsrechte ③ hingegen werden einem Benutzer für PHP-Skripte nicht vergeben, dies würde ein Sicherheitsrisiko darstellen. Bei Internetprovidern sind in der Regel Rechte wie `CREATE DATABASE`, `CREATE USER` und `SET PASSWORD` geblockt. Selbst wenn der Provider phpMyAdmin zur Verwaltung der Datenbank anbietet, können Sie darüber keine Datenbanken löschen oder Nutzer anlegen. Für diese Aktionen bieten Provider andere Administrationsoberflächen an.



- Nach Eingabe der Daten und Vergabe der Rechte betätigen Sie die Schaltfläche **OK**. Der Benutzer wird gespeichert und steht sofort für den Einsatz in PHP-Skripten zur Verfügung.

PHP mit dem MySQL-Server verbinden

Eine Verbindung zum MySQL-Server wird aus dem PHP-Programm heraus hergestellt. Folgende Funktionen werden zum Aufbau und Beenden einer Verbindung zu einem MySQL-Server verwendet:

<code>mysqli_connect()</code> ;	Verbindung zum MySQL-Server aufnehmen
<code>mysqli_close()</code> ;	Verbindung zum verbundenen MySQL-Server beenden

Syntax der Funktion `mysqli_connect()`

```
mysqli_connect ([Server [, Benutzername [, Passwort [, Datenbank [,Port  
[, Socket]]]]]);
```

- ✓ Erster Parameter ist der Name oder die IP des Datenbankservers. Falls `NULL` oder `localhost` übergeben wird, versucht die Funktion die Verbindung zum lokalen Datenbankserver aufzunehmen.
- ✓ Zweiter Parameter ist der Datenbank-Benutzername, dritter das dazugehörige Passwort. Wird kein Benutzername angegeben, wird der Name des Benutzers verwendet, dem der Server-Prozess gehört. Wird kein Passwort angegeben, wird ein leeres Passwort benutzt.
- ✓ Als vierter Parameter kann der Name der Datenbank übergeben werden. Falls Sie diesen Parameter weg lassen, stellt `mysqli_connect` die Verbindung zum Datenbankserver her, ohne aber eine Datenbank auszuwählen. Mit der Angabe einer Datenbank wird diese mit dem Funktionsaufruf sofort ausgewählt.
- ✓ Als fünften Parameter können Sie einen Port angeben. Für die Verbindung zum Datenbankserver ist standardmäßig der Port 3306 konfiguriert, was bei den meisten Webservern auch der Fall ist. Insofern ist die Angabe eines Ports zumeist überflüssig. Ist für den Datenbankserver ein anderer Port konfiguriert, müssen Sie den entsprechenden Port als `integer` angeben.
- ✓ Der letzte optionale Parameter ist der Pfad zu einem Socket, z. B. `/tmp/mysql.sock`.
- ✓ Alle Parameter von `mysqli_connect()` sind optional.
- ✓ Als Rückgabewert erhalten Sie ein Objekt, welches Sie in einer Variablen speichern. Diese Variable hat automatisch den Datentyp `object`. Diese Variable repräsentiert die Verbindung zur Datenbank, Sie benötigen diese für alle weiteren Aktionen mit der Datenbank.
- ✓ Scheitert der Verbindungsaufbau zur Datenbank, gibt `mysqli_connect()` `FALSE` zurück.

- ✓ Sie können in einem PHP-Skript mehr als eine Datenbankverbindung aufbauen (z. B. weil Sie auf mehrere Datenbanken zugreifen wollen). In dem Fall vergeben Sie für jedes Datenbankobjekt eine individuelle Variable (z. B. \$verbindung1, \$verbindung2).

Schlägt `mysqli_connect()` fehl, wird nur eine Warnung ausgegeben, das PHP-Skript selber jedoch weiter ausgeführt. Dies ist häufig nicht gewünscht, da jeder weitere Befehl zur Steuerung von Datenbanken damit fehlschlägt. Es empfiehlt sich der Einsatz der Funktion `mysqli_connect()` in Kombination mit dem Befehl `or die("Meldung")`.



```
mysqli_connect() or die("Meldung");
```

Dieser `or`-Zweig wird dann ausgeführt, wenn die Datenbankverbindung fehlgeschlagen ist. Die Ausführung des Skripts wird nach der Ausgabe des Parameters Meldung abgebrochen.

```
Beispiel: mysqli_connect("localhost", "php-user", "vHM4HVjzqEjyPK3r")
          or die("Verbindung konnte nicht hergestellt werden.");
```

Alternativ können Sie den Rückgabewert auf `FALSE` überprüfen, um die Seite nicht abrupt abzubrechen und eine nutzerfreundliche Meldung anzuzeigen, die Sie entsprechend in das Webseitenlayout implementieren. In dem Fall dürfen nachfolgende SQL-Zugriffe nicht ausgeführt werden, sondern sollten über `if`-Abfragen übersprungen werden.

Syntax der Funktion `mysqli_close()`

```
mysqli_close(Verbindungskennung);
```

- ✓ Diese Funktion beendet eine Verbindung mit dem Datenbank-Server. Die Angabe der Verbindungskennung ist zwingend notwendig. Ohne diesen Parameter gibt PHP eine Fehlermeldung vom Typ `warning` aus. Die Datenbankverbindung wird nicht geschlossen.
- ✓ PHP schließt, nachdem ein PHP-Skript vollständig ausgeführt wurde, automatisch die Server-Verbindung. Sie sollten dennoch eine Verbindung immer selbst schließen, um genutzte Ressourcen sicher – und möglichst frühzeitig – freizugeben. Dies gehört unter anderem zum guten Stil in der PHP-Programmierung.

Beispiel: db_connect.php

Es wird eine Verbindung zum lokalen MySQL-Server hergestellt und wieder geschlossen. Zur Demonstration werden entsprechende Meldungen am Bildschirm ausgegeben.

①	<pre><?php \$server = "localhost"; \$user = "php-user"; \$pass = "vHM4HVjzqEjyPK3r";</pre>
②	<pre>// Verbindungsaufnahme mit dem MySQL-Server \$verbindung = mysqli_connect(\$server, \$user, \$pass) or die("Verbindung konnte nicht hergestellt werden.");</pre>
③	<pre>echo "<p>Die Verbindung mit dem Server wurde hergestellt.</p>"; \$return = mysqli_close(\$verbindung); if (\$return) {</pre>
④	<pre> echo "<p>Die Verbindung mit dem Server wurde beendet. </p>"; } else { echo "<p>Die Verbindung mit dem Server konnte nicht geschlossen werden. </p>"; } ?></pre>

- ① Zu Beginn des Skripts werden die Anmelddaten für einen Zugriff auf den MySQL-Server hinterlegt. Der Server ist auf dem lokalen Rechner installiert (localhost). Den Variablen \$user für den Benutzer und \$pass für das Passwort werden die Werte des PHP-Benutzers zugewiesen.
- ② Über die Funktion `mysqli_connect()` und mit der Angabe der Anmelddaten wird eine Verbindung zum MySQL-Server hergestellt. Das zurückgegebene Objekt wird in der Variablen \$verbindung gespeichert.
- ③ Sind die Anmelddaten nicht korrekt bzw. ist der MySQL-Server nicht gestartet, kann keine Verbindung hergestellt werden. In diesem Fall erhält die Variable \$verbindung den Rückgabewert FALSE, der den mit `or` eingeleiteten Zweig auslöst. Dort wird über die Funktion `die()` eine Fehlermeldung ausgegeben und das aktuelle Skript beendet.
- ④ War die Verbindung erfolgreich, wird am Ende des Skripts die aktuelle Verbindung zum Server mit der Funktion `mysqli_close()` getrennt. Als Parameter wird die Variable \$verbindung übergeben, welche das Verbindungsobjekt enthält.



Die gewünschte Datenbank auswählen

Nach einer erfolgreichen Verbindung zum MySQL-Server bestimmen Sie mithilfe von `mysqli_select_db()`, welche Datenbank genutzt werden soll.

<code>mysqli_select_db();</code>	Auswahl der Datenbank
----------------------------------	-----------------------

Syntax der Funktion `mysqli_select_db()`

<code>mysqli_select_db(Verbindungskennung, Datenbankname);</code>

- ✓ Das von `mysqli_connect()` zurückgelieferte Objekt wird als erster Parameter angegeben.
- ✓ Als zweiten Parameter übergeben Sie den Namen der zu nutzenden Datenbank `Datenbankname`.
- ✓ Der Rückgabewert ist bei Erfolg `TRUE`, ansonsten `FALSE`.

Beispiel: `db_select.php`

Nachdem eine Verbindung zum lokalen MySQL-Server hergestellt wurde, muss nun die gewünschte Datenbank ausgewählt werden.

```
<?php
$server = "localhost";
$user = "php-user";
$pass = "vHM4HVjzqEjyPK3r";
$database = "obstladen";

// Verbindungsaufnahme mit dem MySQL-Server
$verbindung = mysqli_connect($server, $user, $pass)
    or die("Verbindung konnte nicht hergestellt werden.");

// Auswahl der gewünschten Datenbank
① mysqli_select_db($verbindung, $database)
    or die("Fehler beim Zugriff auf die gewünschte Datenbank");

② echo "<p><em>Die Datenbank <strong>$database</strong> wurde
ausgewählt.</em></p>";
?>
```

Beispieldatei "db_select.php"

- ① Die gezeigten Skriptzeilen werden in die Beispieldatei `db_select.php` integriert. Nach der Verbindung mit dem Datenbank-Server wird über die Funktion `mysqli_select_db()` die als zweiter Parameter angegebene Datenbank `obstladen` gespeichert in der Variablen `$database` ausgewählt. Wichtig ist, dass als erster Parameter die Objektvariable für die Verbindungskennung angegeben ist. Schlägt die Auswahl der Datenbank fehl, wird das Skript nach der Ausgabe einer Meldung beendet.
- ② Es erfolgt eine Ausgabe, dass die Datenbank ausgewählt wurde. Diese Zeile des Skripts wird nur erreicht, wenn die Aktivierung der Datenbank erfolgreich war.

Verbindung mit einem Datenbankserver herstellen und eine Datenbank auswählen

Beispiel: `db_connect_select.php`

Die beiden Schritte, Verbindungsaufbau zum Datenbankserver und Auswahl der Datenbank können in einem Schritt umgesetzt werden.

```
<?php
$server = "localhost";
$user = "php-user";
$pass = "vHM4HVjzqEjyPK3r";
$database = "obstladen";

// Verbindungsaufnahme mit dem MySQL-Server und Datenbankauswahl
$verbindung = mysqli_connect($server, $user, $pass, $database)
    or die("Verbindung konnte nicht hergestellt werden.");

echo "<p>Die Verbindung mit dem Server wurde hergestellt, die Datenbank
<strong>$database</strong> wurde ausgewählt.</p>";
?>
```

- ① Beim Verbindungsaufbau über `mysqli_connect()` wird der vierte Parameter für die Auswahl der Datenbank angegeben. Nach dem Herstellen der Verbindung wird die angegebene Datenbank ausgewählt. Eine separate Auswahl der Datenbank erübrigkt sich damit. Falls die Verbindung nicht hergestellt oder die Datenbank nicht ausgewählt werden kann, liefert `mysqli_connect()` FALSE zurück, das PHP-Skript wird in dem Fall mit dem `die()`-Befehl und der Ausgabe einer Fehlermeldung beendet.
- ② Nur wenn der Verbindungsaufbau und die Auswahl der Datenbank erfolgreich waren, wird diese Zeile im PHP-Code erreicht – es wird eine Erfolgsmeldung angezeigt.

13.6 MySQL-Abfragen

Abfrage senden

Zur Abfrage der Tabellendaten einer bestimmten Datenbank sind die nachfolgenden Befehle notwendig.

<code>mysqli_query();</code>	Senden einer MySQL-Anfrage zur aktiven Datenbankverbindung
------------------------------	--

Syntax der Funktion `mysqli_query()`

<code>mysqli_query(Verbindungskennung, SQL-Abfrage);</code>

- ✓ Eine Anfrage an MySQL realisieren Sie über die Funktion `mysqli_query()`.
- ✓ Der erste Parameter stellt die Verbindungskennung dar, als zweiten geben Sie ein SQL-Statement, also eine beliebige Datenbank-Abfrage an.
- ✓ Als Rückgabewert erhalten Sie entweder ein Objekt (bei SELECT-Statements), ein TRUE (z. B. bei INSERT-Statements) oder ein FALSE im Fehlerfall. Das zurückgelieferte Objekt bei einem SELECT-Statement liefert die Anzahl der Felder sowie die Anzahl der Zeilen, die ermittelt worden sind, die eigentlichen Daten sind in diesem Objekt **nicht** enthalten. Die Daten selber werden danach über weitere `mysql`-Funktionen ermittelt (weiter unten).

Beispiel: mysqli_query.php

In diesem Beispiel wird eine Verbindung zur Datenbank aufgenommen und eine SQL-Anweisung ausgeführt. Zur Veranschaulichung wird jede Aktion als Meldung im Browser ausgegeben.

```
<?php
$server = "localhost";
$user = "php-user";
$pass = "vHM4HVjzqEjyPK3r";
$database = "obstladen";

① $verbindung = mysqli_connect($server, $user, $pass, $database)
    or die("Verbindung konnte nicht hergestellt werden.");

echo "<p>Die Verbindung mit dem Server wurde hergestellt, die Datenbank
      <strong>$database</strong> wurde ausgewählt.</p>";

② $sql = "SELECT * FROM bestellung";
③ if (mysqli_query($verbindung, $sql)) {
    echo "<p>Die SQL-Anweisung war erfolgreich.</p>";
} else {
    echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
}
④ $return = mysqli_close($verbindung);
if ($return) {
    echo "<p>Die Verbindung mit dem Server wurde beendet. </p>";
} else {
    echo "<p>Die Verbindung mit dem Server konnte nicht geschlossen
          werden. </p>";
}
?>
```

- ① Es wird versucht, eine Verbindung zum SQL-Server herzustellen und mit dem Aufruf gleichzeitig die Datenbank `obstladen` (über die Variable `$database`) ausgewählt. Kommt keine Verbindung zustande, wird das Skript mit einer Fehlermeldung abgebrochen.
- ② Es folgt die Formulierung einer SQL-Abfrage. Der SQL-Befehl kann in Anführungszeichen direkt in der Funktion `mysqli_query()` notiert werden. Eleganter und übersichtlicher ist es hingegen, den SQL-Befehl in einer Variablen zu speichern. So haben Sie die Möglichkeit, den Befehl über PHP dynamisch zusammenzusetzen. Im Beispiel sollen alle Felder der Tabelle `bestellung` ausgewählt werden. Beachten Sie, dass die Abfrage in Zeile ② noch nicht ausgeführt wird, sondern lediglich in der Variablen `$sql` gespeichert wird.
- ③ Danach wird die Funktion `mysqli_query()` mit den Parametern `$verbindung` und `$sql` aufgerufen. Die Funktion gibt im Erfolgsfall ein Objekt zurück, ansonsten `FALSE`. Je nachdem, ob die Anweisung erfolgreich ausgeführt werden konnte, wird eine entsprechende Meldung angezeigt.
- ④ Zum Schluss wird die Verbindung zum MySQL-Server wieder geschlossen.



Fehlertext von SQL-Operationen anzeigen

<code>mysqli_error();</code>	Ausgabe eines SQL-Fehlers
------------------------------	---------------------------

Syntax der Funktion `mysqli_error()`

```
mysqli_error(Verbindungskennung);
```

Um Informationen zum aufgetretenen Fehler – sei es bei der Verbindungsaufnahme, bei Abfragen oder anderen Operationen – zu erhalten, verwenden Sie die Funktion `mysqli_error()`. Diese leitet den Fehler-Text der zuletzt ausgeführten MySQL-Funktion vom MySQL-Server an das PHP-Skript weiter. Diese Informationen können dabei helfen, Fehler schnell zu finden und zu beheben.

Beispiel: Ausschnitt aus `mysqli_error.php`

In diesem Ausschnitt eines PHP-Skripts soll eine Fehlerrückgabe ausgelöst werden. Hierfür wurde bewusst ein syntaktischer Fehler ① in einer SQL-Abfrage eingefügt. Es werden keine Feldnamen bzw. kein * für die Rückgabe aller Feldnamen, die für eine SELECT-Anweisung zwingend erforderlich sind, angegeben, die aus der Tabelle ausgelesen werden sollen.

```
$sql = "SELECT ① FROM bestellung";
// FEHLER: hier wurde nicht angegeben,
// welche Felder der Tabelle ausgewählt werden sollen

if (mysqli_query($verbindung, $sql)) {
    echo "Die SQL-Anweisung war erfolgreich.";
} else {
    echo "SQL-Fehler!<br>SQL meldet: " . mysqli_error($verbindung);
}
```



Ansicht der Beispieldatei "mysqli_error.php"

13.7 Rückgabe aus MySQL-Abfrage auswerten

Datensätze einer MySQL-Tabelle mithilfe von PHP anzeigen

Nachdem die Verbindung zur MySQL-Datenbank `obstladen` hergestellt wurde, können Sie z. B. die Anzahl der Datensätze oder alle Datensätze der Tabelle `bestellung` anzeigen. Wenn Sie wissen möchten, wie viele Datensätze die SQL-Abfrage zurückliefert, verwenden Sie die Funktion `mysqli_num_rows()`.

Syntax der Funktion `mysqli_num_rows()`

```
mysqli_num_rows(Abfrageergebnis);
```

- ✓ Diese Anweisung liefert Ihnen die Anzahl der Datensätze, die mit dem SQL-Befehl `SELECT` an das Skript zurückgeliefert werden.
- ✓ Der Parameter `Abfrageergebnis` ist das zurückgelieferte Objekt der aktuellen SQL-Abfrage, also dem Rückgabewert der Funktion `mysqli_query()`.

```
Beispiel: $ergebnis = mysqli_query("SELECT * FROM bestellung");
echo mysqli_num_rows($ergebnis);
```

Ausgabe der Datensätze mit der Funktion `mysqli_fetch_array()`

Über die Funktion `mysqli_fetch_array()` in Kombination mit einer `while`-Schleife können Sie die einzelnen Zeilen der Datenbankabfrage auslesen. Dazu notieren Sie im Kopf der `while`-Schleife die Funktion `mysqli_fetch_array()` mit dem Objekt, das Sie von der Funktion `mysqli Query()` als Rückgabewert erhalten haben, als Parameter. Bei jedem Schleifendurchlauf wird eine Ergebnissezeile als Array gespeichert. Die `while`-Schleife wird solange ausgeführt, bis alle Einträge der Datenbankabfrage durchlaufen sind.

Syntax der Funktion `mysqli_fetch_array()`

```
mysqli_fetch_array(Abfrageergebnis [, Ergebnistyp]);
```

- ✓ Der anzugebende Parameter `Abfrageergebnis` stellt die Ergebniskennung der aktuellen SQL-Abfrage dar.
- ✓ Mit der optionalen Angabe `Ergebnistyp` legen Sie fest, mit welchem Arraytyp die Daten des Ergebnisses zurückgegeben werden. Die möglichen Parameter sind `MYSQLI_ASSOC`, `MYSQLI_NUM` oder `MYSQLI_BOTH`. Hiermit bestimmen Sie, wie Sie die Elemente des Arrays ansprechen möchten.
 - ✓ Bei `MYSQLI_ASSOC` verwenden Sie den Feldnamen der Tabelle als Schlüssel im Array (z. B. `zeile["sorte"]`, assoziatives Array).
 - ✓ Bei `MYSQLI_NUM` arbeiten Sie über die Angabe des numerischen Indexes (z. B. `zeile[3]`, numerisch indiziertes Array).
 - ✓ Wenn Sie die Option `MYSQLI_BOTH` verwenden, sind beide Varianten zum Ansprechen eines Elements möglich. (Das Array enthält dann sowohl numerische als auch assoziative Schlüssel, d. h., jeder Wert ist im Array doppelt vorhanden und über zwei verschiedene Schlüssel ansprechbar.)
- ✓ Wenn Sie keinen `Ergebnistyp` angeben, wird standardmäßig die Option `MYSQLI_BOTH` verwendet.

Beispiel: `mysqli_fetch_array.php (Auszug)`

In diesem Beispiel werden Datensätze aus einer Tabelle ausgelesen und angezeigt.

```

$SQL = "SELECT * FROM bestellung";
$abfrage = mysqli_query($verbindung, $SQL);
if (!$abfrage) {
    echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
}
$anzahl = mysqli_num_rows($abfrage);
echo "<p>In der Tabelle befinden sich $anzahl Datensätze:</p>";
echo "<ul>";
② while ($zeile = mysqli_fetch_array($abfrage)) {
    echo "<li>" . $zeile["id"] . ", "
    . $zeile["vorname"] . ", "
    . $zeile["nachname"] . ", "
    . $zeile["ort"] . ", "
    . $zeile["sorte"] . ", "
    . $zeile["menge"] . " kg";
}
echo "</ul>";
```

Auszug aus der Datei „`mysqli_fetch_array.php`“. Aufbau der Datenbankverbindung und das Schließen der Verbindung werden hier nicht dargestellt.

- ① Die Anzahl der Datensätze ermitteln Sie mit der Funktion `mysqli_num_rows()`.

- ② Mithilfe der Funktion `mysqli_fetch_array()` ermitteln Sie ein Array, das dem aktuellen Datensatz (also einer Zeile des Ergebnisses aus der Tabelle) entspricht. Gleichzeitig wird durch diese Funktion der sogenannte Datensatzeiger auf den nächsten Datensatz des Ergebnisses gesetzt. Durch die while-Schleife werden somit nacheinander alle Datensätze der Tabelle an ein Array (`$zeile`) übergeben. Die Schlüssel dieses Arrays sind die Feldnamen der MySQL-Tabelle. Wenn keine weiteren Datensätze vorliegen, wird `FALSE` zurückgegeben und damit die Schleife beendet.

Daten aus einer Tabelle auslesen und darstellen

In der Tabelle befinden sich 4 Datensätze:

- 1, Arndt, Hoffmann, Stuttgart, Elstar, 15 kg
- 2, Corinna, Delphi, Hamburg, Janagold, 3 kg
- 3, Petra, Meyer, Wien, Gala, 5 kg
- 4, Peter, Schmidt, Berlin, Elstar, 25 kg

Anzeige der Beispieldatei "mysqli_fetch_array.php"

Zeichensatz mit der Funktion `mysqli_set_charset()` festlegen

Die `mysqli`-Erweiterung arbeitet standardmäßig mit dem Zeichensatz (`charset`) `latin1` und behandelt die eingelesenen Daten, als hätten sie die `collation latin1_swedish_ci`. Das führt bei Umlauten und dem ß in der Praxis regelmäßig zu Darstellungsproblemen. Die Sonderzeichen werden nicht korrekt erkannt, der Browser ersetzt Zeichen, die er nicht darstellen kann, mit einem Platzhalter, wie in der Abbildung zu sehen ist.

16, G♦nther, M♦ller, M♦nchen, Gala, 5 kg

Fehlerhafte Darstellung bei Verwendung des falschen Zeichensatzes

Um dem Problem zu begegnen, steht in PHP die Funktion `mysqli_set_charset()` zur Verfügung. Diese erwartet als erstes das Objekt der Datenbankbindung, als zweites den gewünschten Zeichensatz. Da `UTF8` derweil als Standardzeichensatz angesehen werden kann, alle beschriebenen Dateien in diesem Buch mit dem Zeichensatz `UTF8` gespeichert sind, die Datenbank mit dem Zeichensatz `UTF8` angelegt ist, setzen Sie für die Datenbankverbindung ebenfalls die `UTF8`-Einstellung. Es empfiehlt sich, die Zuweisung des Zeichensatzes direkt nach dem Aufbau der Verbindung zur Datenbank über `mysqli_connect()` durchzuführen.

Syntax der Funktion `mysqli_set_charset()`

```
mysqli_set_charset(Verbindungsvariable, Zeichensatz);
```

Beispiel: `mysqli_fetch_array.php` (Auszug)

	\$verbindung = mysqli_connect(\$server, \$user, \$pass, \$database) or die("Verbindung konnte nicht hergestellt werden.");
①	if (!mysqli_set_charset(\$verbindung, "utf8")) {
②	printf("Zeichensatz konnte nicht gesetzt werden: %s\n", mysqli_error(\$verbindung));
③	} else { printf("Aktuell verwendeter Zeichensatz: %s\n", mysqli_character_set_name(\$verbindung)); }

Auszug aus der Datei „`mysqli_fetch_array.php`“. Hier der Ausschnitt mit der Funktion `mysqli_set_charset()`.

- ① Über die Funktion `mysqli_set_charset()` setzen Sie den Zeichensatz für die Datenbankverbindung auf UTF8. Als ersten Parameter erwartet die Funktion das Objekt der Datenbankverbindung.
- ② Falls der Zeichensatz nicht gesetzt werden kann, gibt die Funktion FALSE zurück. Neben fehlender Verbindungskennung kann auch ein falsch geschriebener Zeichensatz die Fehlerursache sein. Das Skript meldet hier den Fehler.
- ③ Im Erfolgsfall wird eine entsprechende Meldung ausgegeben. Über die Funktion `mysqli_character_set_name()` können Sie zusätzlich den verwendeten Zeichensatz im Browser ausgeben.

13.8 Formulardaten in einer MySQL-Datenbank speichern

Die von Ihnen erstellten Tabellen der Datenbank können Sie von den Besuchern Ihrer Webseite automatisch füllen lassen. Dazu füllt der Benutzer ein Formular aus. Mithilfe von PHP werden dann die Formulardaten in die MySQL-Datenbank-Tabelle eingetragen.

Beispiel: *bestellformular_db.html*

The screenshot shows a web browser window with the title 'Daten in MySQL-Date...'. The URL bar shows 'http://localhost'. The page content is as follows:

Apfelkauf im Obstladen

Bitte geben Sie folgende Daten für Ihre Bestellung ein:

Vorname: _____

Nachname: _____

Wohnort: _____

Menge (in kg): _____

Apfelsorte: Jonagold Gala Elstar

Abschicken

Anzeige des Formulars "bestellformular_db.html"

```

<h1>Apfelkauf im Obstladen</h1>
<p>Bitte geben Sie folgende Daten für Ihre Bestellung ein:</p>
① <form action="bestellung_db.php" method="post">
②   <p>Vorname: <input type="text" name="vorname"></p>
   <p>Nachname: <input type="text" name="nachname"></p>
   <p>Wohnort: <input type="text" name="ort"></p>
   <p>Menge (in kg): <input type="text" size="5" name="menge"></p>
   <p>Apfelsorte:
      <input type="radio" name="sorte" value="Jonagold">Jonagold
      <input type="radio" name="sorte" value="Gala">Gala
      <input type="radio" name="sorte" value="Elstar">Elstar
    </p>
   <p><input type="submit" value="Abschicken"> </p>
</form>

```

- ① Die Daten des Formulars werden an das PHP-Skript *bestellung_db.php* übergeben.
- ② Die vom Besucher einzugebenden Daten sind: Vorname, Nachname, Wohnort, Menge und Sorte.

Beispiel: *bestellung_db.php* (Auszug)

Im Folgenden wird ein PHP-Skript erstellt, das die Daten des obigen HTML-Formulars auswertet. Die eingegebenen Daten sollen in die Tabelle **bestellung** der Datenbank **obstladen** gespeichert werden.

```

① $vorname = $_POST["vorname"];
$Nachname = $_POST["Nachname"];
$ort = $_POST["ort"];
$sorte = $_POST["sorte"];
$menge = $_POST["menge"];

② $verbindung = mysqli_connect($server, $user, $pass, $database)
    or die("Verbindung konnte nicht hergestellt werden.");

③ if (!mysqli_set_charset($verbindung, "utf8")) {
    printf("Error loading character set utf8: %s\n", mysqli_error($verbindung));
}

④ $sql = "INSERT INTO bestellung (vorname,nachname,ort,sorte,menge)";
$sql .= " VALUES ('$vorname', '$Nachname', '$ort', '$sorte', '$menge')";

⑤ $abfrage = mysqli_query($verbindung, $sql);
if ($abfrage) {
    echo "<p>Vielen Dank, Ihre Bestellung wurde gespeichert.</p>";
} else {
    echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
}

```

- ① Die Variablen `$vorname`, `$Nachname`, `$ort`, `$sorte` und `$menge` werden mit den Daten aus dem Formular gefüllt.
- ② Die Verbindung zum Datenbankserver wird hergestellt und die entsprechende Datenbank ausgewählt.
- ③ Damit die Daten im richtigen Zeichensatz an die Datenbank übergeben werden, setzen Sie den UTF8-Zeichensatz. Diese Einstellung ist notwendig, auch wenn die HTML-Datei mit dem UTF8-Zeichensatz abgespeichert wurde.
- ④ Die SQL-Anweisung zum Eintrag der übermittelten Werte in die Tabelle `bestellung` wird definiert. Die Felder müssen nach dem Tabellennamen angegeben werden, da in der Tabelle sechs Felder existieren, aber nur fünf Werte übergeben werden. Das Feld `id` ist vom Typ `auto_increment` und wird automatisch hochgezählt.
- ⑤ Die Abfrage wird ausgeführt. Anschließend wird eine Fehler- oder Erfolgsmeldung ausgegeben.

13.9 Übung

Einen Newsletter abonnieren

Übungsdatei: --

Ergebnisdateien: *newsletter.html*, *newsletter.php*

MySQL-Dump: *homepage-newsletter.sql*

Auf vielen Webseiten können Benutzer ihren Namen und E-Mail-Adresse hinterlassen, um einen Newsletter zu erhalten. Erstellen Sie eine Webseite, auf der die Besucher der Webseite Ihren Newsletter bestellen können.

- ① Erstellen Sie mithilfe von phpMyAdmin eine neue MySQL-Datenbank mit dem Namen `homepage`.

- ② Erzeugen Sie in dieser Datenbank die Tabelle *newsletter*. Die Tabelle soll folgende zwei Felder enthalten:

Feldname	Feldtyp	Null	Unique
UserName	varchar(50)	null	
UserMail	varchar(50)	not null	x

Alternativ können Sie die Import-Datei *homepage-newsletter.sql* verwenden, um Datenbank und Tabelle zu erstellen.

- ③ Die Anmeldung für den Newsletter geschieht über ein HTML-Formular, das Sie wie folgt gestalten können:

Über dieses Formular können Sie sich für den Newsletter anmelden:

Ihr Name: _____

Ihre E-Mail-Adresse: _____

Absenden

Anzeige der Formulardatei "newsletter.html"

- ④ Die Daten des Formulars sollen an das PHP-Skript *newsletter.php* übergeben werden.
- ⑤ Erstellen Sie das PHP-Skript *newsletter.php*, das die Daten des HTML-Formulars auswertet. Aufgabe des PHP-Skripts ist es, den Namen und die E-Mail-Adresse des Besuchers in die Tabelle *newsletter* der Datenbank *homepage* einzutragen.
- ⑥ Überprüfen Sie, ob der Benutzer eine E-Mail-Adresse angegeben hat. Wenn das Feld keinen Eintrag hat, soll der Benutzer eine Meldung und einen Link erhalten, mit dem er zum Formular zurück navigieren kann.
- ⑦ Der Versuch, eine E-Mail-Adresse mehrfach einzutragen, schlägt fehl, da das E-Mail-Feld als *unique* deklariert wurde. Lassen Sie für diesen Fall mithilfe der Funktion *mysqli_error()* eine entsprechende Fehlermeldung ausgeben.
- ⑧ Nachdem die Daten an die Tabelle *newsletter* übergeben wurden, soll eine Meldung erscheinen, z. B. *Die E-Mail-Adresse arndt@hoffmann.pc wurde gespeichert.*

Newsletter-Eintragung

Vielen Dank, die Mail-Adresse *max@mustermann.com* wurde gespeichert...
Sie erhalten ab sofort unsere Newsletter an die angegebene Adresse...

Bestätigungsseite nach erfolgreichem Eintrag

Installation und Konfiguration der Software

A.1 Installation und Konfiguration von XAMPP

Was ist XAMPP?

XAMPP ist ein Projekt der Apache Friends, das als Softwarebundle mit einer einfachen Installationsroutine auf einem "xbeliebigen" internetfähigen Betriebssystem den Apache-Webserver, MySQL, PHP und Perl lokal installiert. Vor allem für unerfahrene Benutzer ist es sehr zu empfehlen, sich nicht durch die teilweise komplexe und schwierige Installation und Konfiguration der Einzelkomponenten zu kämpfen. Aber auch fortgeschrittene Programmierer schätzen durchaus die fast automatische Installation innerhalb weniger Minuten.

Kostenloser Download des bereits konfigurierten Webservers:

<https://www.apachefriends.org/download.html>

Die für das Buch notwendige XAMPP-Version 5.6.3 mit der PHP-5.6.3-Version steht aktuell für drei Betriebssysteme zur Verfügung:

- ✓ für Windows
- ✓ für Linux-Systeme
- ✓ für Mac OS X

Download MAMP: <http://www.mamp.info/de/>

Alternativ zu XAMPP können Sie auch MAMP (My Apache – MySQL – PHP) 3.0 verwenden, welches von der appsolute GmbH angeboten wird. Dieser vorkonfigurierte und einfach zu installierende Webserver stand früher nur Mac-Benutzern zur Verfügung. Seit Anfang 2014 ist MAMP auch für Windows verfügbar. In der Mac-Version ist PHP 5.5.9 enthalten, PHP 5.6.2 kann separat heruntergeladen und eingebunden werden. Die Windows-Version wird mit PHP 5.6.0 ausgeliefert. MAMP steht zusätzlich in einer kostenpflichtigen Version (MAMP PRO) zur Verfügung.



Alle PHP-Skripte und Beispiele in diesem Buch sind auf dem XAMPP-Webserver umgesetzt und getestet worden. Es werden alle Komponenten (Apache, PHP und MySQL) bis auf die Sprache Perl verwendet. Die Beispiele sind in einer Windows-Umgebung entwickelt worden, daher wird auch für die Installation der Einsatz von XAMPP und Konfiguration Windows beispielhaft herangezogen.

Bei der Erstellung dieses Buchs wurde mit der XAMPP-Version 5.6.3 vom 19. November 2014 gearbeitet. Das Softwarebundle umfasst die für das Buch relevanten Komponenten in folgenden Versionen:

- ✓ PHP, Version 5.6.3
- ✓ Apache Webserver, Version 2.4.10
- ✓ MySQL, Version 5.6.21
- ✓ phpMyAdmin, Version 4.2.11

Installiert wurde das in der Abbildung gezeigte XAMPP 5.6.3 unter dem Betriebssystem Windows 7 Home Premium (Service Pack 1) und den Standardeinstellungen für alle Programme.

Download von XAMPP

Download

Hier klicken für weitere Versionen

XAMPP für Windows

v5.6.3 (PHP 5.6.3)

XAMPP für Linux

v5.6.3 (PHP 5.6.3)

XAMPP für OS X

v5.6.3 (PHP 5.6.3)

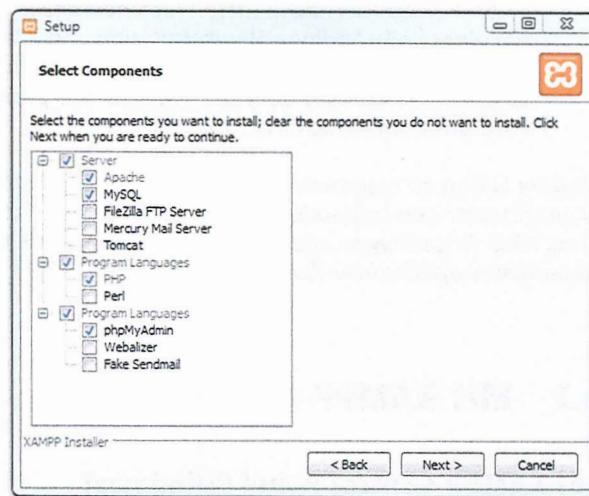
Auf der Startseite von <https://www.apachefriends.org> finden Sie Download-Links der Installer-Versionen für die unterschiedlichen Betriebssysteme. Weitere XAMPP-Versionen stehen auf der Download-Seite: <https://www.apachefriends.org/download.html> zur Verfügung. Dort finden Sie Versionen sowohl für Windows, Linux als auch für Mac OS X.

- Laden Sie von der angegebenen Internetadresse das aktuelle XAMPP-Paket herunter.

Installation von XAMPP

Der XAMPP-Installationsassistent ist so voreingestellt, dass Sie für eine Standardinstallation keine Einstellungen verändern müssen.

- ▶ Um XAMPP 5.6.3 zu installieren, folgen Sie den Schritten des Installationsassistenten.
- ▶ Falls Sie eine Firewall installiert oder die Windows Benutzerkontensteuerung aktiviert haben, erscheinen zu Beginn der Installation Warnhinweise. Bestätigen Sie diese mit *OK*. Falls Sie als Mac-Nutzer ein Passwort für den root-Nutzer verwenden, kann die Installation nur nach Eingabe des Passwortes gestartet werden.
- ▶ Klicken Sie in allen Dialogen die Schaltfläche *Next*, um die Installation durchzuführen.
- ▶ Wählen Sie die Module von XAMPP aus, die Sie installieren möchten (Windows). Wählen Sie mindestens *PHP*, *MySQL* und *phpMyAdmin* aus. Die anderen Module sind optional und für das Arbeiten mit diesem Buch nicht notwendig. Unter Mac OS ist die Auswahl reduziert, hier wählen Sie mindestens die *XAMPP Core Files* aus.
- ▶ Falls Sie XAMPP nicht in dem voreingestellten Verzeichnis *C:\xampp* installieren möchten, wählen Sie ein anderes Verzeichnis aus (gilt für Windows). Eine Auswahl des Installationspfads unter Mac OS ist nicht möglich.
- ▶ Nach Abschluss der Installation ist die Checkbox *Do you want to start the Control Panel now?* ausgewählt. Lassen Sie diese ausgewählt, klicken Sie *Finish*, um die Installation abzuschließen und gleichzeitig das Control Panel zu öffnen (vgl. nächsten Abschnitt).
- ▶ Mac-Nutzer erhalten einen anderen Dialog mit der Option *Launch XAMPP*. Über die Schaltfläche *Finish* schließen Sie die Installation ab und starten den Apache Webserver.



Wenn Sie bei der Installation die Standardeinstellungen beibehalten, werden folgende Einstellungen vorgenommen:

- ✓ Die Installation erfolgt unter Windows standardmäßig in das Verzeichnis *C:\xampp*.
- ✓ Im Startmenü (Windows) wird der Eintrag *XAMPP* sowie *XAMPP Control Panel* hinzugefügt.
- ✓ Unter Mac OS erfolgt die Installation im Verzeichnis */Applications/XAMPP*.

Sollten Sie mit einer anderen XAMPP-Version arbeiten, können die Schritte während der Installation, die Einstellungsmöglichkeiten und die Bezeichnungen der Schaltflächen von dieser Beschreibung abweichen. Bei allen Installationsassistenten in der Vergangenheit waren die Standardeinstellungen jedoch stets sinnvoll voreingestellt, so dass auch bei anderen Versionen keine speziellen Einstellungen vorgenommen werden müssen.

Die Installation des MAMP-Webservers ist vergleichbar. Folgen Sie den Anweisungen während der Installation oder lesen Sie das Benutzerhandbuch, welches unter <http://www.mamp.info/de/dokumentation/MAMP-3-Benutzerhandbuch.pdf> zum Download zur Verfügung steht.



Probleme bei der Installation von XAMPP

Durch die einfache und automatische Installation von XAMPP sind Probleme bei der Installation recht selten. Die häufigsten Probleme bereiten folgende Bereiche:

- ✓ Berechtigungen: Der Benutzer, der XAMPP installieren möchte, verfügt nicht über ausreichende Berechtigung für diese Softwareinstallation. Installieren Sie gegebenenfalls XAMPP als Administrator.
- ✓ Unter Windows kann die Benutzerkontensteuerung (User Account Control, UAC) die Installation verhindern. Eine Lösung für dieses Problem ist, die Benutzerkontensteuerung zu deaktivieren. Eine andere ist, den Standardinstallationspfad C:\xampp zu verändern und die XAMPP-Installation in einem eigenen Benutzeroberverzeichnis zu installieren, beispielweise in den eigenen Dokumenten.
- ✓ MySQL ist bereits installiert: Andere Software installiert mitunter einen eigenen MySQL-Server. Das kann bei der Installation von XAMPP zu Problemen zwischen den beiden vorhandenen Instanzen des Servers führen. In der Regel hilft die Deinstallation des ersten MySQL-Servers, damit XAMPP keinen bereits installierten MySQL-Server vorfindet.
- ✓ Installation einer alten XAMPP-Version: Das Passwort für den MySQL-SuperUser kann über die lokale XAMPP-Weboberfläche (<http://localhost/security/index.php>) in einem Cookie gespeichert werden. Auch bei einer vollständigen Neuinstallation von XAMPP bleibt das Cookie im Browser erhalten, was eine Zugriffsverweigerung von phpMyAdmin auf die Datenbank zur Folge haben kann. Hier hilft das Löschen des Cookies oder der Wechsel der phpMyAdmin-Authentifikation von cookie auf http. Damit wird das Cookie nicht mehr abgefragt.

Weitere Hilfen zu Fragen und Problemen rund um die Installation und den Betrieb von XAMPP finden Sie im Internet unter den folgenden Adressen: Windows FAQ: https://www.apachefriends.org/faq_windows.html, Linux FAQ: https://www.apachefriends.org/faq_linux.html, Mac OS X FAQ: https://www.apachefriends.org/faq_osx.html.

A.2 Mit XAMPP arbeiten

Das XAMPP Control Panel (Windows)

Das XAMPP Control Panel ist die Schaltzentrale zur Steuerung der einzelnen Komponenten. Standardmäßig wird beim Abschluss der Installation angeboten, das XAMPP Control Panel zu starten.

Verwenden Sie diese Applikation in erster Linie dazu, den Betrieb der einzelnen Komponenten zu steuern und zu überprüfen. Der Apache Webserver (einschließlich dem PHP-Parser) als auch der MySQL-Server müssen gestartet sein, damit PHP-Skripte überhaupt geparsert werden und Datenbankzugriffe möglich sind.

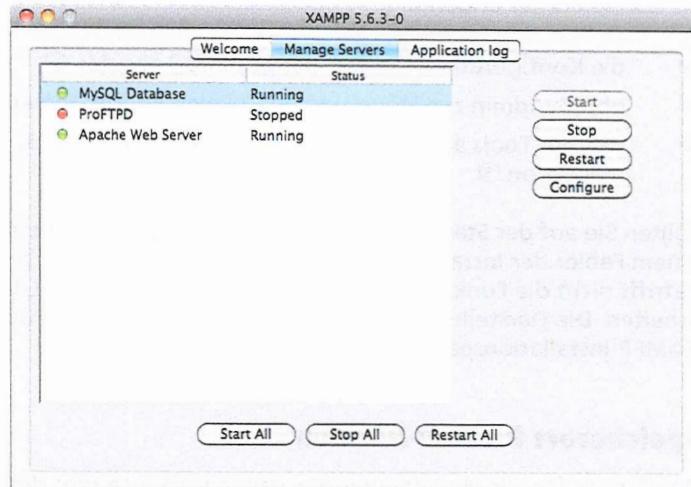
Welchen Status die Komponenten haben, erkennen Sie daran, ob die jeweilige Schaltfläche *Start* oder *Stop* anzeigt ①. Über diese Schaltflächen starten und stoppen Sie die Komponenten. Zusätzlich erkennen Sie die aktivierte Komponente an der grünen Hinterlegung des Modulnamens ②.



Sollen Apache-Webserver und MySQL-Server beim Hochfahren des Rechners automatisch gestartet werden, können Sie die Checkbox vor dem Modulnamen aktivieren ③. Dazu muss die jeweilige Komponente gestoppt werden, falls sie gerade aktiv ist. Mit der Auswahl der Checkbox wird die Komponente als Dienst installiert und startet dann automatisch beim Computerstart. Diese Einstellung ist sinnvoll, da Sie nicht jedes Mal die beiden Server manuell starten müssen.

manager-osx unter Mac OS

Vergleichbar zum XAMPP Control Panel für Windows steuern Sie den Apache Webserver sowie den MySQL-Datenbankserver über das manager-osx-Programm. Dieses finden Sie unter *Programme - XAMPP - xamppfiles - manager-osx*. Öffnen Sie das Programm und wechseln zum Register *Manage Servers*. Wählen Sie die einzelnen Einträge aus und bestätigen die Schaltfläche *Start*. Grüne Symbole vor den einzelnen Server-Einträgen zeigen an, dass der entsprechende Server gestartet ist. Falls bei der Installation die aktivierte Checkbox *Launch XAMPP* beibehalten wurde, läuft der Apache Webserver bereits. Sie müssen nur die MySQL-Datenbank starten. Gegebenenfalls müssen Sie den Rechner einmal neu starten, sollten die Server nicht zu starten sein.



Webserver testen

Jeder Webserver im Internet ist über einen eindeutigen Domain-Namen zu erreichen, beispielsweise <http://www.google.de>. Genauso verhält es sich mit dem lokalen Webserver.

Sie rufen den lokalen Webserver über die Adresse **<http://localhost>** oder alternativ über die IP-Adresse **<http://127.0.0.1>** auf. Sie können zudem auch den Namen des Rechners in der Netzwerkumgebung angeben.

- Starten Sie Ihren Browser.
- Geben Sie in der Adresszeile <http://127.0.0.1> oder <http://localhost> ein.

Bei erfolgreicher Installation und gestartetem Webserver erscheint die Startseite des installierten XAMPP. Beim ersten Start der Seite erscheint einmalig eine Abfrage, welche Sprache verwendet werden soll.

In der Navigation auf der Startseite von XAMPP sind die häufig genutzten Funktionen als Links angegeben. Sie können unter anderem ...

- ✓ die Konfiguration von PHP per `phpinfo()` abrufen ①,
- ✓ phpMyAdmin zur Steuerung des lokalen MySQL-Servers aufrufen ②,
- ✓ weitere Tools aufrufen, wie z. B. den Sicherheitscheck, den Status der Komponenten oder die Dokumentation ③.

 Sollten Sie auf der Startseite von XAMPP Fehler bei der Darstellung von Umlauten sehen, liegt das nicht an einem Fehler der Installation, sondern an einer falschen Einstellung der lokalen XAMPP-Seiten. Dieser Fehler betrifft nicht die Funktionalität der Webserver- oder MySQL-Installation, Sie können mit den Servern arbeiten. Die Darstellungsprobleme sind bekannt, es ist davon auszugehen, dass dieser in zukünftigen XAMPP-Installationspaketen behoben sein wird.

Speicherort für PHP-Dateien

Der Ordner für alle Webdokumente, das sogenannte *root directory*, lautet bei einer Standardinstallation unter Windows C:\xampp\htdocs, unter Mac OS finden Sie den entsprechenden Ordner unter *Programme - XAMPP - xamppfiles - htdocs*. Alle Dokumente, die sich in diesem Ordner befinden, können Sie im Browser bei gestartetem Apache-Webserver unter der Adresse <http://localhost/<Dateiname>> aufrufen.

Noch einfacher ist es, im Ordner C:\xampp\htdocs einen Unterordner zu erstellen, z. B. *myphp*, der all Ihre PHP-Dateien wie die Beispielskripte dieses Buches enthält. Dann können Sie mit der Adresse <http://localhost/myphp> den Ordnerinhalt auflisten und müssen die gewünschte Datei zur Anzeige nur noch anklicken.

A.3 Installation und Konfiguration von Notepad++

Arbeiten mit einem Texteditor

Um ein PHP-Skript zu schreiben, wie Sie es beispielsweise von HTML kennen, benötigen Sie einen Texteditor. Ein Editor ist eine Software zur Bearbeitung von ASCII-Texten und kann daher zur Eingabe von PHP-Code verwendet werden. Texteditoren sind auf allen Computer-Plattformen verfügbar und besitzen unterschiedliche Funktionsausstattungen. Ein Editor unterscheidet sich von einer Textverarbeitung oder einem Layout-Programm dadurch, dass er keine – normalerweise unsichtbaren – Formatierungsanweisungen in den Text einfügt, um ihn in Absätze, Listen, Tabellen usw. zu gliedern.

PHP-Editor einsetzen

Im Internet finden Sie auch spezielle PHP-Editoren, die eine Hervorhebung der Befehle und Hilfestellungen für verschiedene Sprachelemente bieten und Sie somit beim Erstellen Ihrer Webseiten visuell unterstützen.

Bei der Erstellung des vorliegenden Buchs wurde mit dem Texteditor **Notepad++**, Version 6.6.9, (<http://notepad-plus-plus.org/download/>) gearbeitet. Der Texteditor Notepad++ ist ein einfach zu bedienender Editor, der viele Programmiersprachen unter Windows unterstützt.

Download und Installation von Notepad++

- ▶ Laden Sie das Programm Notepad++ von der Webseite <http://notepad-plus-plus.org/download/> herunter.
- ▶ Um Notepad++ zu installieren, folgen Sie den Schritten des Installationsassistenten.
- ▶ Starten Sie nach erfolgreicher Installation das Programm.

Notepad++ wird bei einer 32-Bit-Windows-Version standardmäßig in den Ordner C:\Program Files\Notepad++ installiert, bei einer 64-Bit-Version in den Ordner C:\Program Files (x86)\Notepad++. Der Editor ist ohne weitere Konfiguration sofort für die Arbeit mit PHP-Skripten einsatzbereit.

The screenshot shows the Notepad++ interface with the following details:

- Title Bar:** C:\xampp\htdocs\herdt_php56_grundlagen\beispiele\kap12\fallbeispiel_kasse.php - Notepad++ [Administrator]
- Menu Bar:** Datei, Bearbeiten, Suchen, Ansicht, Kodierung, Sprachen, Einstellungen, Makro, Ausführen, Erweiterungen, Fenster, ?
- Toolbar:** Standard icons for file operations like Open, Save, Print, Find, Replace, etc.
- Code Editor:**

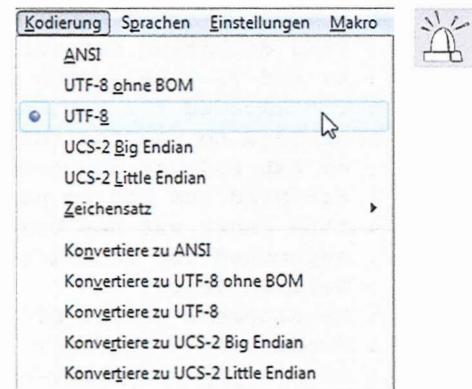
```

1 <?php
2 session_start();
3 include("fallbeispiel_artikel.inc.php");
4 ?<!DOCTYPE html>
5 <html>
6   <head>
7     <title>Kasse</title>
8   </head>
9   <body>
10    <h1>Fallbeispiel "Shop": Bestellung abschließen</h1>
11    <?php
12    if (isset($_POST["absenden"])) {
13      $vorname = $_POST["vorname"];
14      $nachname = $_POST["nachname"];
15      $ort = $_POST["ort"];
16      echo "<p>Sie haben folgende Bestellung übermittelt:</p>";
17      echo "<p><strong>$vorname $nachname aus $ort</strong></p>";
18      echo "<table border='1'><tr><th>Art.-Nr.</th><th>Artikel</th><th>Menge</th></tr>";
19      $bestellung = "<\n>nArt.-Nr.;Artikel;Menge<\n>";
20      while (list($key, $value) = each($_SESSION)) {
21        if (substr($key, 0, 1) == "s") {
22          echo "<tr><td>$key</td><td>$schoko_feld[$key]</td><td>$value</td></tr>";
23          $bestellung .= "$key;$schoko_feld[$key];$value<\n>";
24        }
25      }
26    }
  
```
- Status Bar:** PHP Hypertext Preprocessor file length: 2329 lines: 52 Ln: 37 Col: 33 Sel: 0 | 0 Dos\Windows UTF-8 INS

Notepad++: Ansicht mit geöffnetem PHP-Skript

Über das Menü *Einstellungen - Optionen* haben Sie die Möglichkeit, den Editor bei Bedarf Ihren persönlichen Vorstellungen anzupassen, z. B. können Sie PHP als Standardsprache einstellen oder das Farbschema zur Anzeige von PHP-Code ändern.

Seit PHP 5.4 ist der Standard-Zeichensatz auf UTF-8 gesetzt. Dies erleichtert den Umgang mit deutschen Sonderzeichen. UTF-8 unterstützt Umlaute und das ß, sodass diese im Quelltext nicht mehr als HTML-Entities angegeben werden müssen. Stellen Sie sicher, dass die Datei mit dem UTF-8-Zeichensatz gespeichert wird. Diese Einstellung finden Sie in Notepad++ unter dem Menüpunkt *Kodierung*.



Alternative für Mac-Nutzer

Notepad++ steht nur für Windows zur Verfügung. Für Mac-Nutzer ist der Editor **Sublime Text** verfügbar (<http://www.sublimetext.com/>). Beachten Sie die Installationshinweise auf der Webseite.

A.4 Mit den XAMPP-Konfigurationsdateien arbeiten

Auf die Konfigurationsdateien von XAMPP zugreifen

Im XAMPP werden alle Einstellungen klassisch über Konfigurationsdateien verändert. Standardmäßig werden die Dateien bei der Installation von XAMPP wie folgt abgelegt:

Konfigurationsdatei (Auswahl der wichtigsten Komponenten)	Verzeichnis (Windows)	Verzeichnis (Mac)
PHP: <i>php.ini</i>	C:\xampp\php	/Applications/XAMPP/xamppfiles/etc
Apache: <i>httpd.conf</i>	C:\xampp\apache\conf\	/Applications/XAMPP/xamppfiles/etc
MySQL: <i>my.ini</i> (<i>my.cnf</i> unter Mac)	C:\xampp\mysql\bin\	/Applications/XAMPP/xamppfiles/etc
phpMyAdmin: <i>config.inc.php</i>	C:\xampp\phpMyAdmin\	/Applications/XAMPP/xamppfiles/phpmyadmin

Wichtige Einträge in der *php.ini*

Änderungen an der Konfigurationsdatei *php.ini* sind Ihnen als Kunde eines Internetproviders in der Regel nicht möglich. Bei einer lokalen Installation haben Sie hingegen Zugriff auf diese Datei. Zum tieferen Verständnis der Konfigurationsmöglichkeiten ist die Kenntnis der Datei *php.ini* von Vorteil.

Den Pfad zur Konfigurationsdatei erfahren Sie mithilfe der PHP-Funktion *phpinfo()*. Dort finden Sie einen entsprechenden Eintrag unter *Loaded Configuration File*.

Der folgende Code zeigt einen Ausschnitt aus der *php.ini*. Zeilen, die mit einem Semikolon beginnen, sind Kommentarzeilen und werden beim Starten des Webservers ignoriert. In der *php.ini* selbst finden Sie Beschreibungen bzw. Erläuterungen der jeweiligen Einstellung auf Englisch. Durch das Löschen eines Semikolons können Sie die jeweilige Einstellung aktivieren.

```
;;;;;;;;;;
; Language Options ;
;;;;;;;;;

; Enable the PHP scripting language engine under Apache.
; http://php.net/engine
engine = On

; This directive determines whether or not PHP will recognize code between
; <? and ?> tags as PHP source which should be processed as such. It's been
; recommended for several years that you not use the short tag "short cut" and
; instead to use the full <?php and ?> tag combination. With the wide spread use
; of XML and use of these tags by other languages, the server can become easily
; confused and end up parsing the wrong code in the wrong context. But because
; this short cut has been a feature for such a long time, it's currently still
; supported for backwards compatibility, but we recommend you don't use them.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/short-open-tag
short_open_tag = Off
; XAMPP for Linux is currently old fashioned
;short_open_tag = On

; Allow ASP-style <% %> tags.
; http://php.net/asp-tags
asp_tags = Off
```

- ✓ Seit der XAMMP-1.8.0-Version ist der Wert der Variablen `short_open_tag` für die Nutzung der kurzen Tag-Angabe standardmäßig auf `Off` gesetzt. Bevorzugen Sie die kürzere Schreibweise der PHP-Einbindung (<? ?> statt <?php ?>), setzen Sie den Wert der Variablen auf `On`. Beachten Sie, dass diese Einstellungen nicht auf allen Servern, auf denen Sie arbeiten, identisch sein müssen. Richten Sie sich immer nach Standards bzw. verwenden Sie die allgemeingültige Schreibweise.
- ✓ Die ASP-Schreibweise ist normalerweise mit der Angabe von `asp_tags = Off` abgeschaltet. Möchten Sie diese Schreibweise verwenden, setzen Sie den Wert auf `On`.



Erst nach Neustart des Webservers liest der PHP-Interpreter die Einstellungen erneut aus und verwendet sie.

Weitere Einträge in der php.ini

Eintrag	Wirkung
display_errors = On/Off	Schaltet die Fehlermeldungen an/aus
allow_url_fopen = On/Off	Erlaubt/verbietet, dass eine URL wie eine Datei aufgerufen werden kann
session.save_path = "c:/xampp/tmp"	Legt den Speicherort für die Session-Datei fest
session.name = PHPSESSID	Gibt den Standardnamen der Session an

A.5 Zugriffsrechte von MySQL mit phpMyAdmin regeln

Das Berechtigungssystem von MySQL

Das MySQL-Berechtigungssystem regelt den Zugriff von Benutzern auf den MySQL-Server, MySQL-Datenbanken und die darin enthaltenen Tabellen und Felder.

Folgende Benutzerrechte können festgelegt werden:

- ✓ globale Zugriffsrechte, um den Zugriff auf den MySQL-Server und die Datenbanken auf diesem zu regeln;
- ✓ Zugriffsrechte auf Datenbankebene, um den Zugriff auf eine bestimmte Datenbank und deren Tabellen festzulegen;
- ✓ Tabellenzugriffsrechte zur Steuerung des Zugriffs auf eine Tabelle und deren Spalten;
- ✓ Spaltenzugriffsrechte, um den Zugriff auf eine bestimmte Spalte festzulegen.

Die verschiedenen Zugriffsrechte für Benutzer werden in der Datenbank *mysql* gespeichert.

Die Datenbank *mysql*

Standardmäßig werden mit der Datenbank *mysql* die Benutzerkonten und die Zugriffsrechte verwaltet.

Der automatisch bei der Installation des MySQL-Servers eingerichtete Benutzer *root* hat uneingeschränkten Zugriff auf den MySQL-Server und somit auch auf die Datenbank *mysql*. Vergeben Sie für den Benutzer *root* ein Kennwort, sodass über den Benutzer *root* nur berechtigte Personen Schreibzugriffe auf die Tabellen dieser Datenbank erhalten. **Verwenden Sie beim Einsatz im Internet einen anderen User mit einem sicheren Kennwort und angepassten Nutzerrechten.**



Benutzerverwaltung mit phpMyAdmin

Um mit phpMyAdmin globale Zugriffsrechte zu regeln, wie z. B. neue Benutzer anlegen, löschen und/oder Zugriffsrechte Benutzern zuweisen oder entziehen, gehen Sie folgendermaßen vor:

- ▶ Starten Sie phpMyAdmin über die Navigation auf der XAMPP-Startseite oder durch Aufruf von <http://localhost/phpmyadmin>
- ▶ Klicken Sie im Startbildschirm des phpMyAdmin auf den Eintrag *Benutzer* ① in der Navigation am oberen Bildschirmrand.

The screenshot shows the phpMyAdmin configuration interface. On the left, there's a sidebar with 'Neulich' and 'Favoriten' sections containing database names: 'cdcol', 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', 'test', and 'webauth'. The main area has tabs for 'Datenbanken', 'SQL', 'Status', 'Benutzer', 'Exportieren', 'Importieren', 'Einstellungen', and 'Mehr'. The 'Einstellungen' tab is active, displaying 'Allgemeine Einstellungen' (Character Set: utf8mb4_general_ci) and 'Anzeige-Einstellungen' (Language: Deutsch - German, Design: pmahomme, Font Size: 82%). To the right, there are two panels: 'Datenbank-Server' (Server: 127.0.0.1 via TCP/IP, Server-Typ: MySQL, Server Version: 5.6.21 - MySQL Community Server (GPL), Protokoll-Version: 10, Benutzer: root@localhost, Server Zeichensatz: UTF-8 Unicode (utf8)) and 'Webserver' (Apache/2.4.10 (Win32) OpenSSL/1.0.1 PHP/5.6.3, Datenbank-Client Version: libmysql - mysqld 5.0.11-dev - 20120503 - \$Id: f373ea5dd5536761406a8022a4b8a374418b240e \$, PHP-Erweiterung: mysqli).

Auf einer neuen Seite sehen Sie eine Übersicht über die vorhandenen Benutzer und deren Berechtigungen. Zusätzlich können Sie hier

- ✓ die Benutzerrechte von ausgewählten Benutzern ändern,
- ✓ einen neuen Benutzer anlegen,
- ✓ vorhandene Benutzer löschen.

The screenshot shows the 'Benutzerübersicht' (User Overview) page in phpMyAdmin. It lists users with their host, password, global rights, and group. The table includes rows for 'Jeder' (host %, password --, rights USAGE), 'root' (host 127.0.0.1, password Nein, rights ALL PRIVILEGES), and 'root' (host ::1, password Nein, rights ALL PRIVILEGES). Below the table, there are buttons for 'Alle auswählen' (Select All) and 'markierte:' (Selected). A confirmation dialog box is open, asking 'Die ausgewählten Benutzer löschen?' (Delete selected users?). It contains the note '(Den Benutzern alle Rechte entziehen und sie anschließend aus den Benutzertabellen löschen.)' (Revoke all rights from the users and delete them from the user tables.) and a checkbox 'Die gleichnamigen Datenbanken löschen.' (Delete the same-named databases.). At the bottom right of the dialog are buttons '(4)' (OK) and '(5)' (Cancel).

Neue Benutzer erstellen

Um einen neuen Benutzer zu erstellen, klicken Sie auf den Link *Neuen Benutzer hinzufügen* ①. Nun können Sie in den verschiedenen Eingabefeldern die entsprechenden Daten des neuen Benutzers eingeben. Abschließend betätigen Sie die Schaltfläche *Benutzer hinzufügen*, um den Benutzer zu erstellen.

Um in das Fenster mit der Benutzerübersicht zu gelangen, klicken Sie auf den Eintrag *Benutzer* in der Navigation. Der von Ihnen erstellte Benutzer wird in der Benutzerübersicht aufgelistet.

Neue Benutzer bearbeiten

Klicken Sie auf das Symbol bzw. den Link für *Rechte ändern* ② in der Zeile eines Benutzers. Ein Fenster wird geöffnet, in dem Sie folgende Werte ändern können:

- ✓ globale und datenbankspezifische Rechte des Benutzers,
- ✓ Anmelde-Information des Benutzers (Benutzername, Host und Passwort).

Benutzer löschen

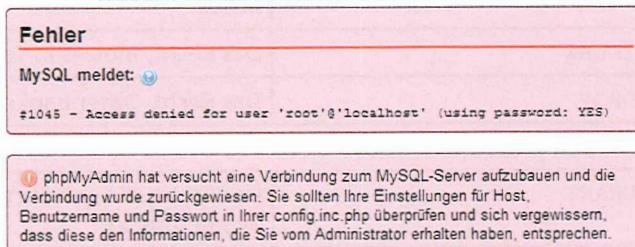
Durch Aktivierung der Checkbox vor dem Benutzernamen ③, Auswahl der Checkbox im Bereich *Die ausgewählten Benutzer löschen* ④ und anschließendes Betätigen der Schaltfläche *OK* ⑤ werden die ausgewählten Benutzer sowie eventuell angelegte Datenbanken dieser Benutzer gelöscht. Vor dem Löschen erscheint eine Sicherheitsabfrage, um ein versehentliches Löschen zu vermeiden.

A.6 Globale Zugriffsrechte des MySQL-Administrators root ändern

Der MySQL-Server startet ohne Passwort für den MySQL-Administrator `root`. Ein Passwort für `root` können Sie in phpMyAdmin wie folgt vergeben:

- ▶ Um die Zugriffsrechte des Benutzers `root` auf dem Host `localhost` zu ändern, klicken Sie in der Ansicht *Benutzerübersicht* auf den Hyperlink *Rechte ändern*. Mit dem Klick öffnet sich ein neues Fenster mit verschiedenen Formularen zur Rechteverwaltung.
- ▶ Aktivieren Sie im Formular *Passwort ändern* das Optionsfeld *Passwort* ① und vergeben Sie ein Kennwort.

Nachdem Sie das Passwort gespeichert haben und dann phpMyAdmin erneut aufrufen möchten, erhalten Sie eine Fehlermeldung. Das liegt daran, dass in der Konfiguration des phpMyAdmin das Passwort für den Nutzer `root` und den Datenbankzugriff hinterlegt ist, welches Sie gerade geändert haben. Aus dem Grund muss das neue Passwort für `root` in der Konfiguration des phpMyAdmin eingetragen werden, um wieder Zugriff zu erhalten. Dies geschieht über die Konfigurationsdatei, in der die nachfolgenden Änderungen vorzunehmen sind:



- ▶ Öffnen Sie die Konfigurationsdatei **C:\xampp\phpMyAdmin\config.inc.php** (Windows) bzw. unter **/Applications/XAMPP/xamppfiles/phpmyadmin/config.inc.php** (Mac OS).
- ▶ Editieren Sie die angegebenen Zeilen und ändern Sie sie wie folgt:

```
$cfg['Servers'][$i]['auth_type'] = 'http';
$cgi['Servers'][$i]['user'] = 'root';
$cgi['Servers'][$i]['password'] = ''; // hier wird das neue Passwort
eingetragen
```

Sobald Sie sich erneut über phpMyAdmin anmelden wollen, erhalten Sie eine Kennwortabfrage. Dieses Kennwort gilt auch, wenn Sie sich per PHP mit der Datenbank verbinden möchten.



Alternativ haben Sie die Möglichkeit, das MySQL-Administrator-Passwort über die Weboberfläche des XAMPP-Webservers zu verändern. Über <http://localhost/security/index.php> können Sie das Passwort ändern, zusätzlich können Sie über Radiobuttons die PhpMyAdmin-Authentifikation-Methode einstellen: *http* oder *cookie*. Dieser Weg erspart Ihnen vor allem das manuelle Bearbeiten der Konfigurationsdatei *config.inc.php*.

Sonstige globale Zugriffsrechte des Benutzers root verändern

- ▶ Bei Bedarf ändern Sie die Zugriffsrechte entsprechend der folgenden Tabelle:

Die Angaben gelten auch für alle anderen Benutzer. In der folgenden Tabelle finden Sie eine Beschreibung der Rechte für den Benutzer:

Zugriffsrecht	Beschreibung
Bereich Daten	
SELECT	Das Recht, die SELECT-Anweisung auszuführen, um ausgewählte Datenfelder aus Tabellen abzufragen
INSERT	Die SQL-Anweisung INSERT darf ausgeführt werden, um neue Datensätze in eine Tabelle einzufügen.
UPDATE	Die SQL-Anweisung UPDATE darf ausgeführt werden, um Daten zu bearbeiten und zu ändern.
DELETE	Diese Angabe gewährt das Recht zum Löschen von Daten.
FILE	Der Zugriff auf das lokale Dateisystem des Rechners mit dem MySQL-Server wird gestattet.
Bereich Struktur	
CREATE	Das Recht, Datenbanken und Tabellen zu erstellen
ALTER	Das Recht zum Ändern der Tabellenstruktur mit der SQL-Anweisung ALTER
INDEX	Das Recht, Indizes in Tabellen zu erstellen und zu löschen
DROP	Das Recht, Datenbanken und Tabellen zu löschen
Bereich Administration	
GRANT	Das Recht, die gewährten Zugriffsrechte an andere Benutzer weiterzugeben
PROCESS	Das Recht, die Prozessliste einzusehen und Prozesse zu löschen
RELOAD	Das Recht zum Ausführen der FLUSH-Anweisung bzw. der RELOAD-Anweisung des Programms Mysqldadmin, um damit die Berechtigungstabelle neu einzulesen
SHUTDOWN	Das Recht, den MySQL-Server mithilfe des Programms Mysqldadmin herunterzufahren

\$

<code>\$_POST</code>	152
<code>\$_REQUEST</code>	75
<code>\$_SERVER</code>	83
<code>\$_SESSION</code>	151
<code>\$_GLOBALS</code>	75
. 30	
.= 30	
?> 13	
<?php	12, 13

D

<code>date()</code>	138, 142
Datei einbinden	101
Dateien überschreiben	111
Dateinamenerweiterung php.....	12
Dateizeiger.....	107
Datenausgabe	16
Datenbank mysql	195
Datenbank, Zeichensatz	183
Datenbanken, Oracle.....	10
Datentyp	26
Datentyp, automatische Zuweisung.....	33
Datentyp, Zeichen.....	30
Datentypen, numerische.....	28
Datentypen, PHP	26
Datenübertragung bei Formularen	72
Datum	136
Datum formatieren.....	138, 141
Datum, Anpassung an Sprache....	139
Datum, deutsch.....	139
Datum, Formatanwei- sungen	138, 141
Datumsdifferenz	143
Datumsprüfung.....	146
<code>default</code>	47
<code>define()</code>	35
Dezimalpunkt.....	28
<code>die()</code>	177, 178
Division.....	29
<code>double</code>	26
<code>do-while-Schleife</code>	49, 50

E

<code>echo</code>	16, 17
Einbinden, Datei	101
<code>else</code>	42
<code>elseif</code>	44
<code>empty()</code>	81
Ergebnisdateien	5
<code>error_reporting()</code>	23
Escape-Sequenzen	17, 18
<code>explode()</code>	132
Exponent.....	29
Exponent, Basis	29
Exponentielle Rechnung	29
Externe Dateien lesen.....	106
Externe Dateien nutzen.....	106
Externe Dateien öffnen	106
Externe Dateien schließen....	106, 108
Externe Dateien schreiben.....	106

F

Fallauswahl	47
<code>FALSE</code>	38
<code>fatal error</code>	21
<code>fclose()</code>	108
Fehlerarten, Benachrichtigung....	21
Fehler, Warnung	21
Fehlerarten in PHP	21
Fehlersuche	23
Felder, Datentyp	26

Feldvariablen

<code>feof()</code>	108
<code>fgets()</code>	107, 108
<code>file()</code>	109, 111
<code>file_exists()</code>	111
<code>file_get_contents()</code>	110
<code>file_put_contents()</code>	113
Fließkommazahl	26, 28
<code>float</code>	26
<code>flock()</code>	115
<code>floor()</code>	143
<code>fopen()</code>	106
<code>for</code>	51
<code>foreach</code>	63, 67
foreach-Schleife.....	62
Formatieren, Zeichenketten.....	119
Formularauswertung.....	72
Formulardaten.....	75
Formulare auswerten	74, 75,
.....	79, 82, 83
Formulare und PHP.....	72
Formulare, Auswertung in derselben PHP-Datei.....	83
Formulare, Checkboxen und Radiobuttons	79
Formulare, Daten eingeben	75, 80
Formulare, Eingabefelder	75
Formulare, mehrere Absende- Schaltflächen	82
<code>for-Schleife</code>	49, 51
<code>for-Schleife, Operatoren</code>	52
<code>fputs()</code>	111, 112
<code>fseek()</code>	115
<code>func_get_args()</code>	95
<code>function</code>	89
<code>function()</code>	89
Funktion	88
Funktion aufrufen	90
Funktion erstellen	89
Funktion, benutzerdefinierte.....	88
Funktion, call-by-reference	93
Funktion, call-by-value	93
Funktion, mehrere Rückgabewerte	97
Funktion, optionale Parameter.....	92
Funktion, Parameter.....	89, 91
<code>funktion, return</code>	89
Funktion, Rückgabewert	89, 96
Funktion, vordefinierte	88
Funktionen, Variadisch.....	95

G

Ganze Zahlen.....	26
Ganzzahl.....	26
<code>GET</code>	72
<code>getdate()</code>	136
GET-Methode	73
<code>gettype()</code>	34
Gleichheitsoperator	39
Gleitkommazahl	28
Gleitkommazahl, Datentyp	26
Globale Variable	98
Greenwich-Zeit	142
Gültigkeitsbereich, Variable.....	98

H

- HTML, PHP-Code in 7
- `htmlentities()` 20
- HTML-Tags in PHP-Anweisungen 16
- HTTP 72, 148
- httpd.conf* 194
- Hypertext Transfer Protocol 72
- Hypertext-Übertragungsprotokoll 72

I

- Identisch-Operator 39
- `if` 39, 44, 45
- `if, else` 42
- `if, elseif` 44
- `if, verschachtelt` 45
- `implode()` 110, 111, 132
- `include()` 101
- `include_once()` 101
- Index** 57
- Installation 188
- `integer` 26
- Internet-Provider 9
- `isset()` 81

K

- `key` 57, 58
- Keywords 27
- Kommentare 15
- Kommentare, einzeilige 15
- Kommentare, mehrzeilige 15
- Konfigurationsdateien 193
- Konkatenation 30
- Konstante skalare Ausdrücke 35
- Konstanten 34
- Kontrollstrukturen 38

L

- Lesen, externe Dateien 107
- `list()` 67, 68, 97, 154
- `localhost` 13, 191
- Logischer Operator 43
- Lokale Variablen 98
- `ltrim()` 129

M

- `manager-osx` 191
- Mehrdimensionale Arrays 64
- `mktime()` 143
- Modulo 29
- Multiplikation 29
- `my.ini` 194
- MySQL 4, 166, 188
- MySQL, Benutzerverwaltung 195
- MySQL, Berechtigungssystem 195
- MySQL, Daten exportieren 173
- MySQL, Daten importieren 173
- MySQL, Datentypen 169
- MySQL, SQL-Dump 173
- MySQL, Zugriffsrechte 195

- MySQL-Datenbanken 9, 166, 195
- MySQL-Datenbank erzeugen 168
- `mysqli_close()` 176
- `mysqli_connect()` 176
- `mysqli_error()` 180
- `mysqli_fetch_array` 182
- `mysqli_num_rows()` 181
- `mysqli_query()` 179
- `mysqli_select_db()` 178
- `mysqli_set_charset()` 183
- MySQL-Server 166
- MySQL-Tabelle erzeugen 169
- MySQL-User verwalten 167

N

- Newsletter 185
- `nl2br()` 110
- Notepad, installieren 192
- Notepad+ 4, 192
- `notice` 22
- `number_format()` 121
- Numerische Datentypen 28

O

- `object` 26
- Open-Source-Datenbanksystem ... 167
- Operator, arithmetischer 28
- Operator, logischer 43
- Operator, ternärer 43
- Operatoren, relationale 38
- OR 43
- Oracle 10

P

- `parse error` 21
- PHP, Anweisungen 14
- PHP, Befehle einfügen 12
- PHP, Definition 6
- PHP, Entwicklung 9
- PHP, Interpreter 6
- PHP, Kommentare 15
- PHP, Merkmale 8
- PHP, öffnender Tag 7
- PHP, schließendes Tag 8
- PHP, Versionen 9
- `php.ini` 194
- PHP-Anweisungen trennen 14
- PHP-Code, in HTML einfügen 7
- PHP-Editor 192
- `phpinfo()` 13
- PHP-Informationsdatei 13
- phpMyAdmin 166, 188
- phpMyAdmin, Benutzer erstellen 197
- phpMyAdmin, Benutzer löschen 197
- phpMyAdmin, Benutzerdaten ändern 197
- phpMyAdmin, Benutzerverwaltung 195
- phpMyAdmin, Tabelle erstellen 169
- phpMyAdmin, User 'root' 197
- `PHPSESSID` 150
- POST 72

- Postdekrement 29
- Postinkrement 29
- POST-Methode 73
- Potenz 29
- Potenz-Operator 29
- Prädekrement 29
- Präinkrement 29
- Primärschlüssel 170
- `print_r()` 76, 77, 137, 152
- `printf()` 118

R

- `readfile()` 109
- Rechenregeln, mathematische 28
- Relationale Operatoren 38
- `require()` 100, 101
- `require_once()` 101
- Reservierte Wörter, PHP 27, 89
- `resource` 26
- `return` 89, 96
- `root` 197
- `rtrim()` 129

S

- Schleife, for** 51
- Schleife, foreach** 62
- Schleife, while** 49
- Schleifen verwenden 49
- Schlüssel 58, 59
- Schlüssel, Array 57
- Schlüsselwörter, PHP 27
- Schreibweise, CamelCase 27
- Selektion, switch-Anweisung 47
- Session 148
- Session fortsetzen 149
- Session konfigurieren 149
- Session löschen 155
- Session starten 149
- Session, Daten lesen 154
- Session, Daten löschen 155
- Session, Daten speichern 151
- Session, Name 150
- `session_destroy()` 155, 156
- `session_id()` 149
- `session_name()` 150
- `session_start()` 149, 150
- Session-Datei 151
- Session-Datei anzeigen 153
- Session-ID 149
- `setlocale()` 140
- Sitzungen 148
- Skalare Ausdrücke 35
- Splat-Operator 11, 95
- SQL 167
- SQL-Abfrage senden 179
- SQL-Dump 173
- `sqrt()` 92
- `str_repeat()` 128
- `str_replace()` 130, 131
- `strcasecmp()` 128
- `strcmp()` 128
- `strcmp()` 128
- `strchr()` 124
- `strftime()` 141, 142

<i>string</i>	26
<i>stristr</i>	122
<i>strlen</i> ()	126
<i>strpos</i> ()	125
<i> strrchr</i> ()	124
<i> strrpos</i> ()	125
<i> strstr</i> ()	122
<i> strtolower</i> ()	129
<i> strtotime</i> ()	145
<i> strtoupper</i> ()	129
<i> strtr</i> ()	130
<i> substr</i> ()	125, 161
<i> substr_count</i> ()	127
Subtraktion	28
Suchen, Zeichen	125
Superglobale Variablen	98
switch	46, 47
switch, break	47
switch, case	47
switch, default	47
switch-case, erweiterte Notation	48

T

Ternärer Operator	43
Texteditor	192
time()	142, 143
Tipps zum Codieren	14
trim()	129
TRUE	38

U

Übungsdateien	5
ucfirst()	129
ucwords()	129
UNIX-Timestamp	142
unset()	61, 155, 156
UTF-8	19
utf8_encode()	21

V

Value	57
var_dump()	76, 77
Variable, globale	98
Variable, Gültigkeitsbereich	98
Variable, lokale	98
Variable, superglobale	98
Variablen	26
Variablen, Ausgabe von	31
Variablen, Wertzuweisung	27
Variadiische Funktionen	95
Vergleichsoperatoren	38
Verkettungsoperator	30
Verschachtelte Auswahl	45
Vorgabewert	93

W

Wahrheitswert, boolscher Wert	26
warning	22
Webcode	5
Webserver	188
Webserver testen	191
Webserver, PHP-Unterstützung	8
Wertzuweisung	27
while	49
while-Schleife	49

X

XAMPP	4, 7, 166, 167, 188
XAMPP Control Panel	190
XAMPP Installationsassistent	189
XAMPP, Arbeit mit	190
XAMPP, document root-Verzeichnis	13
XAMPP, Download	188
XAMPP, Installationsprobleme	189
XAMPP, konfigurieren	193
XAMPP, root directory	192

XML-Schreibweise

12, 13

XOR.....

43

Z

Zahlen formatieren	121
Zeichenkette modifizieren	128
Zeichenkette vergleichen	122
Zeichenkette wiederholen	128
Zeichenkette, Länge bestimmen	126
Zeichenkette, Position ermitteln	125
Zeichenkette, Teilsting bestimmen	125
Zeichenkette, Umwandeln in Feld	132
Zeichenkette, Zeichen austauschen	130
Zeichenkette, Zeichenfolge austauschen	130
Zeichenkette, Länge	126
Zeichenketten	26
Zeichenketten ausgeben	118
Zeichenketten formatieren	118
Zeichenketten suchen	122, 123
Zeichenketten vergleichen	128
Zeichenketten, Datentyp	26
Zeichenketten, Konkatenation	30
Zeichenketten, Vergrößern von	30
Zeichenkettenoperator	30
Zeichensatz	183
Zeichensatz, UTF-8	19
Zeit	136, 142
Zeitspannen berechnen	144
Zeitstempel	142, 144
Zend Engine	9
Zend Technologies Ltd.	9
Zugriffszähler	114
Zuweisungsoperator =	27
Zuweisungsoperatoren	29

Impressum

Matchcode: GPHP56

Autor: Stephan Heller

Redaktion: Andrea Weikert

Produziert im HERDT-Digitaldruck

1. Ausgabe, März 2015

HERDT-Verlag für Bildungsmedien GmbH
Am Kümmerling 21-25
55294 Bodenheim
Internet: www.herdt.com
E-Mail: info@herdt.com

© HERDT-Verlag für Bildungsmedien GmbH, Bodenheim

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlags reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Wenn nicht explizit an anderer Stelle des Werkes aufgeführt, liegen die Copyrights an allen Screenshots beim HERDT-Verlag. Sollte es trotz intensiver Recherche nicht gelungen sein, alle weiteren Rechteinhaber der verwendeten Quellen und Abbildungen zu finden, bitten wir um kurze Nachricht an die Redaktion.

Die in diesem Buch und in den abgebildeten bzw. zum Download angebotenen Dateien genannten Personen und Organisationen, Adress- und Telekommunikationsangaben, Bankverbindungen etc. sind frei erfunden. Eventuelle Übereinstimmungen oder Ähnlichkeiten sind unbeabsichtigt und rein zufällig.

Die Bildungsmedien des HERDT-Verlags enthalten Verweise auf Webseiten Dritter. Diese Webseiten unterliegen der Haftung der jeweiligen Betreiber, wir haben keinerlei Einfluss auf die Gestaltung und die Inhalte dieser Webseiten. Bei der Bucherstellung haben wir die fremden Inhalte daraufhin überprüft, ob etwaige Rechtsverstöße bestehen. Zu diesem Zeitpunkt waren keine Rechtsverstöße ersichtlich. Wir werden bei Kenntnis von Rechtsverstößen jedoch umgehend die entsprechenden Internetadressen aus dem Buch entfernen.

Die in den Bildungsmedien des HERDT-Verlags vorhandenen Internetadressen, Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen waren zum Zeitpunkt der Erstellung der jeweiligen Produkte aktuell und gültig. Sollten Sie die Webseiten nicht mehr unter den angegebenen Adressen finden, sind diese eventuell inzwischen komplett aus dem Internet genommen worden oder unter einer neuen Adresse zu finden. Sollten im vorliegenden Produkt vorhandene Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen nicht mehr der beschriebenen Software entsprechen, hat der Hersteller der jeweiligen Software nach Drucklegung Änderungen vorgenommen oder vorhandene Funktionen geändert oder entfernt.



Bénédict-Schule Zürich

Militärstrasse 106
8004 Zürich

Tel. 044 242 12 60
Fax 044 291 07 90
E-Mail: info.zh@benedict.ch

Bénédict-Schule St. Gallen

Neumarkt 1/St.-Leonhardstrasse 35
9000 St. Gallen

Tel. 071 226 55 55
Fax 071 226 55 44
E-Mail: info.sg@benedict.ch

Bénédict-Schule Luzern

Inseliquai 12b
Lakefront Center
6005 Luzern

Tel. 041 227 01 01
Fax 041 227 01 02
E-Mail: info.lu@benedict.ch

Bénédict-Schule Bern

Aarbergergasse 5
3011 Bern

Tel. 031 310 28 28
Fax 031 310 28 29
E-Mail: info.be@benedict.ch